Elimination tree

Jean-Yves L’Excellent and Bora Uçar

GRAAL, LIP, ENS Lyon, France

CR-07: Sparse Matrix Computations, October 2010
http://graal.ens-lyon.fr/~bucar/CR07/
1 The elimination tree
   • Definition and properties
   • Determining row and column counts
   • Determination
   • Some other properties
The role of elimination tree

There are many uses of elimination trees for sparse factorization. We will be discussing its use in Cholesky factorization $LL^T$ of a given symmetric positive matrix $A$.

Nonzeros in $L$ can be characterized in terms of paths in the elimination tree.

In particular, the row and column structures (or the number of nonzeros in each row and column of $L$) can be expressed in terms of the elimination tree. Predicting the structure of $L$.

Recall that one can build the graph of $L$ using the elimination process. This gives a running time proportional to the size of the filled-in graph $G^+ (A)$ or the number of nonzeros in $L$. Not bad, but we want more.
Set-up

Reminder

- A spanning tree of a connected graph \( G = (V, E) \) is a tree \( T = (V, F) \), such that \( F \subseteq E \).
- A topological ordering of a rooted tree is an ordering that numbers children vertices before their parent.
- A postorder is a topological ordering which numbers the vertices in any subtree consecutively.

Let \( A \) be an \( n \times n \) symmetric positive-definite and irreducible matrix, \( A = LL^T \) its Cholesky factorization, and \( G^+(A) \) its filled graph (graph of \( F = L + L^T \)).

We assume an ordering (at least partial) on \( A \).
A first definition

Since $A$ is irreducible, each of the first $n - 1$ columns of $L$ has at least one off-diagonal nonzero (prove?).

For each column $j < n$ of $L$, remove all the nonzeros in the column $j$ except the first one below the diagonal.

Let $L_t$ denote the remaining structure and consider the matrix $F_t = L_t + L_t^T$. The graph $G(F_t)$ is a tree called the elimination tree.
A first definition

The elimination tree of $A$ is a spanning tree of $G^+(A)$ satisfying the relation $PARENT[j] = \min\{i > j : \ell_{ij} \neq 0\}$. 

![Diagram showing elimination tree and related matrices and graphs.](image)
A second definition: Represents column dependencies

- Dependency between columns of $\mathbf{L}$:
  - Column $i > j$ depends on column $j$ iff $\ell_{ij} \neq 0$
  - Use a directed graph to express this dependency (edge from $j$ to $i$, if column $i$ depends on column $j$)
  - Simplify redundant dependencies (transitive reduction)

- The transitive reduction of the directed filled graph gives the elimination tree structure. Remove a directed edge $(j, i)$ if there is a path of length greater than one from $j$ to $i$. 
Directed filled graph and its transitive reduction

Directed filled graph

Transitive reduction

T(A)
The elimination tree

A third definition: DFS tree

Theorem

The elimination tree $T(A)$ of a connected graph $G(A)$ is a depth-first search tree of the filled graph $G^+(A)$.

Proof.

Let $x_1, x_2, \ldots, x_n$ be the node ordering of $G^+(A)$. Consider the depth-first search subject to the following tie-breaking rule: when there is a choice of more than one node to explore next, always pick the one with largest subscript. With this additional rule the depth-first search will construct $T(A)$. 

Any DFS on an undirected graph produces only Tree and Back edges.
Path characterization of filled edges

**Nonzeros of $L$**

If $\ell_{ij} \neq 0$, then node $x_i$ is an ancestor of $x_j$ in $T(A)$.

**Some zeros of $L$**

Let $T[x_i]$ and $T[x_j]$ be two disjoint subtrees of $T(A)$. Then $\ell_{st} = 0$ for any $x_s \in T[x_i]$ and $x_t \in T[x_j]$.

- Simply because there is no cross edge in the DFS tree of an undirected graph.
- The corollary also shows that the elimination tree expresses potential parallelism (resulting from the sparsity).
Fill-in entries

**Fill-path theorem** [Rose, Tarjan, Lueker’76]

Let $G = (V, E, \alpha)$ be an ordered graph. Then $(v, w)$ is an edge of $G^*_\alpha = (V, E \cup F(G_{\alpha}))$ iff there exists a path $\mu = [v = v_1, v_2, \ldots, v_{k+1} = w]$ in $G$ such that

$$\alpha^{-1}(v_i) < \min\{\alpha^{-1}(v), \alpha^{-1}(w)\}, \quad 2 \leq i \leq k$$

**Restating using the elimination tree**

Let $i > j$. Then $\ell_{ij} \neq 0$ iff there exists a path $x_i, x_{p_1}, \ldots, x_{p_k}, x_j$ in the graph of $A$ such that $\{x_{p_1}, \ldots, x_{p_k}\} \subseteq T[x_j]$. Here $T[x_j]$ is the set of nodes in the subtree rooted at $x_j$. 
Row structure theorem

Let \( i > j > k \). We have \( l_{ij} \neq 0 \) iff node \( x_j \) is an ancestor of some node \( x_k \) in the elimination tree with \( a_{ik} \neq 0 \).

Define \( T_r[x_i] \) to be the structure of the \( i \)th row of \( L \), i.e.,
\[
T_r[x_i] = \{ x_j : l_{ij} \neq 0, j \leq i \}. \]
We have \( T_r[x_i] \subseteq T[x_i] \).

A previous theorem (\( l_{ij} \neq 0 \) iff \( x_i \) is an ancestor of \( x_j \)) states \( \subseteq T[x_i] \). In other words, \( T_r[x_i] \) is obtained from \( T[x_i] \) by pruning: Find the leaves of \( T_r[x_i] \) and prune \( T[x_i] \) at those leaves.
**Corollary**

The node $x_j$ is a leaf node in the row subtree $T_{r}[x_i]$ iff $a_{ij} \neq 0$ and for every proper descendant $x_k$ of $x_j$, we have $a_{ik} = 0$. 

$$F = \begin{pmatrix} a & \bullet & \bullet & \bullet & \bullet \\ b & c & d & \bullet & \bullet \\ \bullet & e & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & i \\ \bullet & \bullet & \bullet & \bullet & j \end{pmatrix}$$
Remark
Each leaf $x_j$ of row subtree $T_r[x_i]$ corresponds to an edge $\{x_i, x_j\}$ in the original graph of $A$.

The row subtree $T_r[x_i]$

$marker[x_i] \leftarrow i$
for $k < i$ and $a_{ik} \neq 0$ do
  $j \leftarrow k$
while $marker[x_j] \neq i$ do
  $marker[x_j] \leftarrow i$
  $j \leftarrow parent(j)$
Column structure of the Cholesky factor

Column structure theorem

The structure of column $j$ of $L$, i.e., $\{x_i : \ell_{ij} \neq 0, i \geq j\}$ is given by

$$\text{Adj}_{G(A)}(T[x_j]) \cup \{x_j\}$$

Here, $\text{Adj}_{G(A)}(S) = \{x \notin S : x \in \text{Adj}_{G(A)}(v) \text{ for some } v \in S\}$

Remark: Since $i \in L_{\ast j}$ iff $j \in L_{i\ast}$, the column $j$ is included in row subtrees $T_r[i]$ that contain $j$. We can traverse all row subtrees $T_r[i]$ and determine the pattern of $L_{\ast j}$ by noting each $i$ such that $T_r[i]$ contains $j$.

$$\text{Adj}_{G(A)}(T[c]) \cup \{c\} = \{c, g, h\}$$
Column structure of the Cholesky factor

Column structure theorem

\[ \{ x_i : \ell_{ij} \neq 0, i \geq j \} = \text{Adj}_{G(A)}(T[x_j]) \cup \{ x_j \} \]

Why? First, since \( \ell_{jj} \neq 0 \), we have to have \( x_j \) in the structure of \( L_{*j} \).

Reminder: Row structure theorem

Let \( i > j > k \). We have \( \ell_{ij} \neq 0 \) iff node \( x_j \) is an ancestor of some node \( x_k \) in the elimination tree with \( a_{ik} \neq 0 \).

Second, if \( \ell_{ij} \neq 0 \) with \( i > j \), by the theorem on the left, the node \( x_j \) is an ancestor of some node \( x_k \) with \( a_{ik} \neq 0 \).

In other words, \( x_i \in \text{Adj}_{G(A)}(x_k) \), where \( x_k \in T[x_j] \). Equivalently, \( x_i \in \text{Adj}_{G(A)}(\ T[x_j]) \).
The row and column counts: Slow version

Traverse each row subtree to count the nonzeros in rows and columns.

\[
cc(j) \leftarrow 1 \text{ for } j = 1, \ldots, n \\
\text{for } i = 1, \ldots, n \text{ do} \\
\quad rc(i) \leftarrow 1 \\
\quad marker[x_i] \leftarrow i \\
\quad \text{for } k < i \text{ and } a_{ik} \neq 0 \text{ do} \\
\quad \quad j \leftarrow k \triangleright \text{traverse and mark the nodes in row subtree } T_r[x_i] \\
\quad \quad \text{while } marker[x_j] \neq i \text{ do} \\
\quad \quad \quad rc(i) \leftarrow rc(i) + 1 \\
\quad \quad \quad cc(j) \leftarrow cc(j) + 1 \\
\quad \quad \quad marker[x_j] \leftarrow i \\
\quad \quad \quad j \leftarrow \text{parent}(j)
\]

Runs in time proportional to the number of nonzeros in \( L \).
The row and column counts

Gilbert, Ng, and Peyton,’94 compute row and column counts in time proportional to $O(m \alpha(m, n))$, where $m = \text{nnz}(A)$ and $\alpha(m, n)$ is the inverse of the Ackermann’s function (which is $\leq 4$ for sensible $m$ and $n$).

The essential ideas are as follows (to highlight the machinery used):

Row counts: postorder the elimination tree, and count the number of edges in a row subtree without building it. We know the leaves, sort them according to the postorder. Find the level of the least common ancestor of two consecutively numbered nodes and subtract it from the first of those two nodes; difference in levels give the length of the path. Add the resulting path lengths.

Column counts: A judicious definition of the column count for a node using the weights of descendants of the node, where the weight of a node are computed as contributions from each row subtree. Again, the postorder and the least common ancestors of two consecutive nodes are used.
Determination of elimination trees

start with an empty forest (no vertices)
for $i = 1, \ldots, n$ do
  add vertex $i$ to the forest
  for $k < i$ and $a_{ik} \neq 0$ do
    make $i$ the parent of the root of the tree containing $k$

For efficiency use disjoint set operations to find the root of the tree containing $j$. This way a complexity of $O(m\alpha(m,n))$ can be achieved.

makeset($x$): create a new singleton set with element $x$.

find($x$): return the representative of the set containing $x$.

link($x$, $y$): form the union of the two sets containing $x$ and $y$, and return the representative of the new set.

Simple yet effective techniques (path compression and balancing) should be used for $O(m\alpha(m,n))$ time — necessitates keeping two structures, in practice, a little costlier version is used.
Determination of elimination trees

for $i = 1, \ldots, n$ do
  makeset($i$)
  parent[$i$] ← 0
  for $k < i$ and $a_{ik} \neq 0$ do
    $u$ ← find($k$)
    if parent[$u$] = 0 and $u \neq i$ then
      parent[$u$] ← $i$
      link($i$, $u$)

$F = \begin{pmatrix}
  a & b & \cdot & \cdot & \cdot \\
  b & c & \cdot & \cdot & \cdot \\
  \cdot & c & d & \cdot & \cdot \\
  \cdot & d & e & \cdot & \cdot \\
  \cdot & \cdot & \cdot & f & \cdot \\
  \cdot & \cdot & \cdot & g & \cdot \\
  \cdot & \cdot & \cdot & \cdot & h \\
  \cdot & \cdot & \cdot & \cdot & i \\
  \cdot & \cdot & \cdot & \cdot & j
\end{pmatrix}$
Determination of elimination trees

for $i = 1, \ldots, n$ do
  makeset($i$)
  parent[$i$] ← 0
  for $k < i$ and $a_{ik} \neq 0$ do
    $u$ ← find($k$)
    if parent[$u$] = 0 and $u \neq i$ then
      parent[$u$] ← $i$
      link($i, u$)
Determination of elimination trees

for \( i = 1, \ldots, n \) do
    \( \text{makeset}(i) \)
    \( \text{parent}[i] \leftarrow 0 \)
    for \( k < i \) and \( a_{ik} \neq 0 \) do
        \( u \leftarrow \text{find}(k) \)
        if \( \text{parent}[u] = 0 \) and \( u \neq i \) then
            \( \text{parent}[u] \leftarrow i \)
            \( \text{link}(i, u) \)

\[
F = \begin{pmatrix}
    a & \bullet & \bullet & \bullet \\
    b & \bullet & \bullet & \bullet \\
    c & d & \bullet & \bullet \\
    \bullet & e & \bullet & \bullet \\
    \bullet & \bullet & \bullet & \bullet \\
    \bullet & \bullet & \bullet & \bullet \\
    \bullet & \bullet & \bullet & \bullet \\
    \bullet & \bullet & \bullet & \bullet \\
    \bullet & \bullet & \bullet & \bullet \\
    \bullet & \bullet & \bullet & \bullet \\
\end{pmatrix}
\]
Determination of elimination trees

```
for i = 1, ..., n do 
    makeset(i)
    parent[i] ← 0
    for k < i and a_{ik} ≠ 0 do
        u ← find(k)
        if parent[u] = 0 and u ≠ i then
            parent[u] ← i
            link(i, u)
```

\[ F = \begin{pmatrix}
    a & \bullet & \bullet & \bullet \\
    b & c & \bullet & \bullet \\
    & d & e & \bullet \\
    & & f & \bullet \\
    & & & g \\
    & & & h \\
    & & & i \\
    & & & j
\end{pmatrix} \]
Determination of elimination trees

for \( i = 1, \ldots, n \) do
  \( \text{makeset}(i) \)
  \( \text{parent}[i] \leftarrow 0 \)
  for \( k < i \) and \( a_{ik} \neq 0 \) do
    \( u \leftarrow \text{find}(k) \)
    if \( \text{parent}[u] = 0 \) and \( u \neq i \) then
      \( \text{parent}[u] \leftarrow i \)
      \( \text{link}(i, u) \)

\[ F = \begin{pmatrix}
  a & \cdot & \cdot & \cdot \\
  b & c & \cdot & \cdot \\
  \cdot & d & e & \cdot \\
  \cdot & \cdot & f & \cdot \\
  \cdot & \cdot & \cdot & g \\
  \cdot & \cdot & \cdot & h \\
  \cdot & \cdot & \cdot & i \\
  \cdot & \cdot & \cdot & j
\end{pmatrix} \]
Determination of elimination trees

\[
\text{for } i = 1, \ldots, n \text{ do}
\]
\[
\text{makeset}(i)
\]
\[
\text{parent}[i] \leftarrow 0
\]
\[
\text{for } k < i \text{ and } a_{ik} \neq 0 \text{ do}
\]
\[
u \leftarrow \text{find}(k)
\]
\[
\text{if } \text{parent}[u] = 0 \text{ and } u \neq i \text{ then}
\]
\[
\text{parent}[u] \leftarrow i
\]
\[
\text{link}(i, u)
\]

\[
F = \begin{pmatrix}
    a & \bullet & \bullet & \bullet \\
    b & \bullet & \bullet & \bullet \\
    c & \bullet & \bullet & \bullet \\
    d & \bullet & \bullet & \bullet \\
    e & \bullet & \bullet & \bullet \\
    f & \bullet & \bullet & \bullet \\
    g & \bullet & \bullet & \bullet \\
    h & \bullet & \bullet & \bullet \\
    i & \bullet & \bullet & \bullet \\
    j & \bullet & \bullet & \bullet \\
\end{pmatrix}
\]

\[\text{FIG. 2.1. An example of matrix structures.}\]

\[\text{FIG. 2.2. Graph structures of the example in Fig. 2.1.}\]
Determination of elimination trees

\[ \text{for } i = 1, \ldots, n \text{ do} \]
\[ \quad \text{makeset}(i) \]
\[ \quad \text{parent}[i] \leftarrow 0 \]
\[ \quad \text{for } k < i \text{ and } a_{ik} \neq 0 \text{ do} \]
\[ \quad \quad u \leftarrow \text{find}(k) \]
\[ \quad \quad \text{if } \text{parent}[u] = 0 \text{ and } u \neq i \text{ then} \]
\[ \quad \quad \quad \text{parent}[u] \leftarrow i \]
\[ \quad \quad \text{link}(i, u) \]

\[ F = \begin{pmatrix}
  a & \bullet & \bullet & \bullet \\
  b & c & \bullet & \bullet \\
  \bullet & d & e & \bullet \\
  \bullet & \bullet & f & \bullet \\
  \bullet & \bullet & g & \bullet \\
  \bullet & \bullet & \bullet & h \\
  \bullet & \bullet & \bullet & i \\
  \bullet & \bullet & \bullet & j
\end{pmatrix} \]
Determination of elimination trees

for $i = 1, \ldots, n$ do
  makeset($i$)
  parent[$i$] ← 0
  for $k < i$ and $a_{ik} \neq 0$ do
    $u$ ← find($k$)
    if parent[$u$] = 0 and $u \neq i$ then
      parent[$u$] ← $i$
      link($i$, $u$)

\[
F = \begin{pmatrix}
  a & \bullet & \bullet & \bullet \\
  b & c & \bullet & \bullet \\
  \bullet & d & e & \bullet \\
  \bullet & \bullet & f & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
  \bullet & \bullet & \bullet & \bullet \\
\end{pmatrix}
\]
Determination of elimination trees

for $i = 1, \ldots, n$ do
  makeset($i$)
  parent[$i$] ← 0
  for $k < i$ and $a_{ik} \neq 0$ do
    $u \leftarrow$ find($k$)
    if parent[$u$] = 0 and $u \neq i$ then
      parent[$u$] ← $i$
      link($i, u$)

$F =$
\[
\begin{pmatrix}
  a & \bullet & \bullet & \bullet & \bullet \\
  b & & & & \\
  c & & & \bullet & \bullet \\
  d & & \bullet & & \\
  e & \bullet & \bullet & \bullet & \bullet \\
  f & & & & \\
  g & \bullet & & \bullet & \bullet \\
  h & \bullet & \bullet & \bullet & \bullet \\
  i & \bullet & \bullet & \bullet & \bullet \\
  j & \bullet & \bullet & \bullet & \bullet \\
\end{pmatrix}
\]
Equivalent reorderings

Definition

Two orderings $P$ and $Q$ are equivalent if the structures of the filled graphs of $PAP^T$ and $QAQ^T$ are the same (that is they are isomorphic).

Equivalent orderings result in the same amount of fill-in and computation during factorization. To ease the notation, we discuss only one ordering wrt $A$, i.e., $P$ is an equivalent ordering of $A$ if the filled graph of $A$ and that of $PAP^T$ are isomorphic.
Equivalent reorderings

Any topological ordering on $T(A)$ are equivalent

Let $P$ be the permutation matrix corresponding to a topological ordering of $T(A)$. Then, $G^+(PAP^T)$ and $G^+(A)$ are isomorphic.

A quoi bon? We can process the children in any order (in a parallel implementation this would help).

There are some other equivalent orderings than the postordering of a given elimination tree. For some other aspects of factorization, one may prefer one ordering to some other. See papers by Liu on the web.
Questions

- Let $L_*^i$ and $L_*^j$ be two columns of $L$ and suppose $i > j$.

  When these columns have the same pattern at indices $k > i$? What if $i$ is the father of $j$?

- Given an ordered matrix and its elimination tree. How can we say if the given ordering is perfect?