Structure prediction, elimination process on the graphs and fill-reducing ordering methods

Fanny Dufossé & Bora Uçar

Inria, Grenoble Rhône-Alpes & CNRS and ENS Lyon, France

CR12: Combinatorial scientific computing, September 2018
http://perso.ens-lyon.fr/bora.ucar/CR12/
Outline

1. Sparse triangular system solutions

2. Predicting the structure of factorization
   - Elimination process on the graphs
Solve $Lx = b$

We are going to solve $Lx = b$ with forward substitution.

At index $i$, we need to solve

$$\sum_{j<i} \ell_{ij}x_j + \ell_{ii}x_i = b_i.$$ 

When do we subtract $\sum_{j<i} \ell_{ij}x_j$ from $b_i$?
Solve $Lx = b$

Row oriented

for $i = 1$ to $n$ do
  $x(i) \leftarrow b(i)$
  for $j = 1$ to $i - 1$ do
    $x(i) \leftarrow x(i) - L(i,j) \times x(j)$
  $x(i) \leftarrow x(i)/L(i,i)$

Pull based: to compute $x_i$ pull other, known $x$.

$L$ is accessed by rows.
Solve \( Lx = b \)

**Column oriented**

- \( x(1 : n) \leftarrow b(1 : n) \)
- \( \text{for } j = 1 \text{ to } n \text{ do} \)
  - \( x(j) \leftarrow x(j)/L(j, j) \)
  - \( x(j + 1 : n) \leftarrow x(j + 1 : n) - L(j + 1 : n, j) \times x(j) \)

Push based: when \( x_j \) is computed, inform others.

\( L \) is accessed by columns.
**Solve** $Lx = b$

**Row oriented**

```
for i = 1 to n do
    x(i) ← b(i)
    for j = 1 to i - 1 do
        x(i) ← x(i) - L(i, j) × x(j)
    x(i) ← x(i)/L(i, i)
```

Pull based: to compute $x_i$ pull other, known $x$.
$L$ is accessed by rows.

**Column oriented**

```
x(1 : n) ← b(1 : n)
for j = 1 to n do
    x(j) ← x(j)/L(j, j)
    x(j + 1 : n) ← x(j + 1 : n) -
                   L(j + 1 : n, j) × x(j)
```

Push based: when $x_i$ is computed, inform others.$L$ is accessed by columns.

- Both versions run in $O(nnz(L))$ time.
- For dense $b$ and $x$, it is ok.
- How one can exploit sparsity of $b$ and $x$?

(some parts of the following four slides are from John Gilbert)
The directed graph $G(A)$ of a matrix $A$ is such that $a_{i,j} \neq 0$ iff $(i,j)$ is an edge in $G(A)$.

The directed graph of a triangular matrix is directed acyclic.
The structure of the solution vector $x$ of $Lx = b$ is given by the set of vertices reachable from vertices of $b$ by paths in the dag of $G(L^T)$.

**Symbolic:** predict the structure of $x$ with depth-first-search from nonzeros of $b$.

**Numeric:** compute the values of $x$ in topological order.
Column oriented algorithm

DFS in $G(L^T)$ to predict nonzeros of $x$

\[
x(1 : n) \leftarrow b(1 : n)
\]

for $j =$ nonzero indices of $x$ in topological order do

\[
x(j) \leftarrow x(j)/L(j,j)
\]

for each $i$ in $L(j + 1 : n, j)$ do

\[
x(i) \leftarrow x(i) - L(i,j) \times x(j)
\]

Run time?

\[\langle\langle\langle \text{Context (on the board): Left-looking LU à la Gilbert & Peierls}\rangle\rangle\rangle\]
DFS visits only the necessary edges and vertices. Not $O(n + \tau)$. In other words not $O(|V| + |E|)$.

Accesses only to the necessary components of $x$ (and of course that of $L$ as before) and performs only necessary operations (no zeros).

Therefore, a lower (or an upper) triangular system $Lx = b$ can be solved in $O(flops(Lx))$ time, if $L$ is stored by columns.

---

Question (Class): Is there a similar algorithm if $L$ is stored by rows? (J. Gilbert)

No $O(L)$ additional storage is allowed. Where is the difficulty?

We can assume reasonable things: that the graph of $L + L^T$ is chordal, i.e., for $i < j < k$, if $\ell_{ji}$ and $\ell_{ki}$ are nonzero, so is $\ell_{kj}$; or an $O(n)$ initialization for $n$ solves that will follow.
Outline

1. Sparse triangular system solutions

2. Predicting the structure of factorization
   - Elimination process on the graphs
From: Gilbert, ’94 [Predicting structure]

$f$ be a function from one or more matrices/vectors to a matrix/vector.

We want to determine the structure of $f(A)$ using $A$.

$A$ is not always enough: sum of two vectors is full, but $(1, 1)^T + (1, -1)^T$ is not.

We ignore zeros created by coincidence in the numerical values of $A$ and determine the smallest structure that is big enough for the result of $f$ with any given input of the given structure. Given $f$ and $\text{struct}(A)$, determine

$$\bigcup_{B} \{ \text{struct}(f(B)) : \text{struct}(B) \subseteq \text{struct}(A) \}$$
Predicting structure helps

- in reducing the memory requirements,
- in achieving high performance,
- in simplifying the algorithms.

We will consider the Cholesky factorization $A = LL^T$. In this case, structural and numerical aspects are neatly separated. We will also touch $A = LU$ without pivoting.

For general case (e.g., LU factorization) pivoting is necessary and depends on the actual numerical values. There are specialized combinatorial tools for these. The methods that are used for $LL^T$ are viable in this case too, hence our focus.

Structure prediction algorithms should run, preferably, faster than the numerical computations that will follow.
Symmetric matrices and graphs

- Assumptions: \( A \) symmetric (positive definite; diagonals are \( > 0 \) even during elimination \( > 0 \))

- Structure of \( A \) (symmetric) is represented by the graph \( G = (V, E) \)
  - Vertices are associated to columns: \( V = \{1, \ldots, n\} \)
  - Edges \( E \) are defined by: \( (i, j) \in E \iff a_{ij} \neq 0 \)
  - \( G \) undirected (symmetry of \( A \))
Number of nonzeros in column $j$ is \( d(j) = |\text{adj}_G(j)| \)
The elimination graph model for symmetric matrices

- Let $A$ be a symmetric positive define matrix of order $n$
- The $LL^T$ factorization can be described by the equation:

$$A = A_0 = H_0 = \begin{pmatrix} d_1 & v_1^T \\ v_1 & H_1 \end{pmatrix}$$

$$= \begin{pmatrix} \sqrt{d_1} & 0 \\ \frac{v_1}{\sqrt{d_1}} & I_{n-1} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & H_1 \end{pmatrix} \begin{pmatrix} \sqrt{d_1} & \frac{v_1^T}{\sqrt{d_1}} \\ 0 & I_{n-1} \end{pmatrix}$$

$$= L_1 A_1 L_1^T$$, where

$$H_1 = H_1 - \frac{v_1 v_1^T}{d_1}$$

- The basic step is applied on $H_1 H_2 \cdots$ to obtain:

$$A = (L_1 L_2 \cdots L_{n-1}) I_n \left( L_{n-1}^T \cdots L_2^T L_1^T \right) = LL^T$$
The basic step: \[ H_1 = \overline{H_1} - \frac{v_1v_1^T}{d_1} \]

What is \( v_1v_1^T \) in terms of structure?

\( v_1 \) is a column of \( A \), hence the neighbors of the corresponding vertex.

\( v_1v_1^T \) results in a dense sub-block in \( H_1 \).

If any of the nonzeros in dense submatrix are not in \( A \), then we have fill-ins.
The elimination process in the graphs

\[ G_U(V, E) \leftarrow \text{undirected graph of } A \]

\for k = 1 : n - 1 \do
  \[ V \leftarrow V - \{k\} \triangleright \text{remove vertex } k \]
  \[ E \leftarrow E - \{(k, \ell) : \ell \in \text{adj}(k)\} \cup \{(x, y) : x \in \text{adj}(k) \text{ and } y \in \text{adj}(k)\} \]
  \[ G_k \leftarrow (V, E) \triangleright \text{for definition} \]

\[ G_k \] are the so-called \textbf{elimination graphs} (Parter, ’61).
A sequence of elimination graphs

\begin{align*}
G_0 & \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \\
G_1 & \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \\
G_2 & \quad 4 \quad 3 \quad 5 \quad 6 \\
G_3 & \quad 3 \quad 4 \quad 5 \quad 6 \\
H_0 & \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \\
H_1 & \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \\
H_2 & \quad 3 \quad 4 \quad 5 \quad 6 \\
H_3 & \quad 4 \quad 5 \quad 6 
\end{align*}
Fill-in and operation count

In the light of the elimination process,

**Question (Class):** what is the total number of nonzeros in $L$?

**Question (Class):** what is the operation count incurred while forming $L$?
In the light of the elimination process,

**Question (Class)**: what is the total number of nonzeros in $L$?

**Question (Class)**: what is the operation count incurred while forming $L$?

express them using the degree of the vertices: $\sum d(i)$ and $\sum d(i)^2$
For an undirected graph $G = (V, E)$ with $|V| = n$, an ordering of $V$ is a bijection $\alpha : \{1, \ldots, n\} \leftrightarrow V$. $\alpha^{-1}(v)$ gives the order of $v$.

$G_\alpha = (V, E, \alpha)$ is an ordered graph.

Deficiency of a vertex: $D(v)$ is the set of edges defined by
$D(v) = \{(x, y) : x \in \text{adj}(v) \text{ and } y \in \text{adj}(v) \text{ and } y \notin \text{adj}(x) \text{ and } x \neq y\}$

At this point, we do not define $D(v)$ according to the ordering.
In most of the cases, we will though.

$v$-elimination graph: Apply the elimination process to the vertex $v$ of $G$ to obtain $G_v = (V - \{v\}, E(V - \{v\}) \cup D(v))$.

For an ordered graph $G_\alpha = (V, E, \alpha)$, the elimination process $P(G_\alpha) = [G = G_0, G_1, G_2, \ldots, G_{n-1}]$ is the sequence of elimination graphs defined by $G_0 = G$, $G_i = (G_{i-1})_{\alpha(i)}$. 
For a directed graph $G = (V, E)$ with $|V| = n$, an ordering of $V$ is a bijection $\alpha : \{1, \ldots, n\} \leftrightarrow V$. $\alpha^{-1}(v)$ gives the order of $v$.

$G_\alpha = (V, E, \alpha)$ is an ordered graph.

**Deficiency of a vertex:** $D(v)$ is the set of edges defined by $D(v) = \{(x, y) : x \rightarrow v$ and $v \rightarrow y$ and $x \not\rightarrow y$ and $x \neq y\}$

**$v$-elimination graph:** Apply the elimination process to the vertex $v$ of $G$ to obtain $G_v = (V - \{v\}, E(V - \{v\}) \cup D(v))$.

For an ordered graph $G_\alpha = (V, E, \alpha)$, the elimination process $P(G_\alpha) = [G = G_0, G_1, G_2, \ldots, G_{n-1}]$ is the sequence of elimination graphs defined by $G_0 = G$, $G_i = (G_{i-1})_{\alpha(i)}$. 

Elimination process: Formal definitions

\[ P(G_{\alpha}) = [G = G_0, G_1, G_2, \ldots, G_{n-1}] \] is the sequence of elimination graphs defined by \( G_0 = G, G_i = (G_{i-1})_{\alpha(i)} \). Let \( G_i = (V_i, E_i) \) for \( i = 0, 1, \ldots, n - 1 \). The fill-in \( F(G_{\alpha}) \) is defined by

\[ F(G_{\alpha}) = \bigcup_{i=1}^{n-1} \tau_i \]

where \( \tau_i = D(\alpha(i)) \) in \( G_{i-1} \), and the elimination graph is defined by

\[ G^*_\alpha = (V, E \cup F(G_{\alpha})) \]

For a matrix \( A \), \( \tau_i \) corresponds to the new nonzeros elements, the fill-ins, created during \( i \) the step of elimination.

Observe that \( (v, w) \in G^*_\alpha \), if \( (v, w) \in E \) or there is a vertex \( u \), \( \alpha^{-1}(u) < \min(\alpha^{-1}(v), \alpha^{-1}(w)) \) and both \( (u, v) \) and \( (u, w) \) are in \( G^*_\alpha \).
Continuing from the previous example, we have the filled-graph $G^*(A)$.
Elimination process: Formal definitions

Given a graph \( G = (V, E) \), an ordering \( \alpha \) of \( V \) is a perfect elimination ordering of \( G \) if \( F(G_\alpha) = \emptyset \).

The ordering \( \alpha \) is a perfect elimination ordering if \( w \in \text{adj}(v) \), \( x \in \text{adj}(v) \), and \( \alpha^{-1}(v) < \min\{\alpha^{-1}(w), \alpha^{-1}(x)\} \) in \( G_\alpha \), imply either \((w, x) \in E \) or \( w = x \). In other words, when \( v \) is to be eliminated (both \( w \) and \( x \) are not eliminated yet), there is an edge \((w, x)\).

A graph which has a perfect elimination ordering is a perfect elimination graph. Any elimination graph \( G^*_\alpha \) is a perfect elimination graph, since \( \alpha \) is a perfect ordering.

Similar statements for the directed graph case.
A graph $G$ is called triangulated if for every cycle $\mu = [v_1, v_2, \ldots, v_\ell]$ of length $\ell > 3$, there is an edge of $G$ joining two nonconsecutive vertices of $\mu$ (different names: chordal, monotone transitive, and rigid circuit).
Elimination process: Perfect elimination

A graph $G$ is called **triangulated** if for every cycle $\mu = [v_1, v_2, \ldots, v_\ell]$ of length $\ell > 3$, there is an edge of $G$ joining two nonconsecutive vertices of $\mu$.

**Simplicial vertex**: is a vertex whose adjacency is a clique. In a triangulated graph, if not a clique, there are at least two nonadjacent simplicial vertices. (**Shown in class**)

**Theorem**: $G$ is triangulated iff it has a perfect elimination ordering (**Proof (Class)**).

- $G$ triangulated $\Rightarrow$ pe: use the property of the simplicial vertices.
- PEO$\Rightarrow$ $G$ triangulated: Take a cycle of length $> 3$, and show the chord.

Any simplicial vertex can be the first vertex of a perfect elimination ordering. (**Proof (Class)**).
Elimination process: Perfect elimination

A graph $G$ is called *triangulated* if for every cycle $\mu = [v_1, v_2, \ldots, v_\ell]$ of length $\ell > 3$, there is an edge of $G$ joining two nonconsecutive vertices of $\mu$.

**Simplicial vertex:** is a vertex whose adjacency is a clique. In a triangulated graph, if not a clique, there are at least two nonadjacent simplicial vertices.

**Lemma:** Let $\alpha$ be a perfect elimination ordering of a triangulated graph $G = (V, E)$ and let $x \in V$. Then, $\alpha$ is a perfect elimination ordering for $G' = (V, E \cup D(x))$. (Proof (Class)?).

**Corollary:** If $G = (V, E)$ is triangulated and $x$ any vertex, the elimination graph $G_x = (V - \{x\}, E(V - \{x\}) \cup D(x))$ is triangulated.
Elimination process: Perfect elimination

One can recognize triangulated graphs by a slightly modified breadth-first search algorithm called **lexicographic breadth-first search** (LexBFS).

assign the label $\emptyset$ to each vertex
\[
\text{for } i = n : -1 : 1 \text{ do}
\]
pick an unnumbered vertex $v$ with the largest label (lexicographic)
\[\alpha(i) \leftarrow v\]
\[
\text{for each unnumbered vertex } w \in \text{adj}(v) \text{ do}
\]
\[\text{label}(w) \leftarrow \text{label}(w) \oplus i\]

We obtain an ordering $\alpha$. Graph is triangulated iff $\alpha$ is a perfect elimination ordering. The essential idea for such a test is to check at each step if the neighbors of a vertex (in the elimination graph) are equal to its original neighbors.

LexBFS runs in $O(|V| + |E|)$ time to generate the ordering, and the ordering can be tested in $O(|V| + |E'|)$ time to see if it is perfect (if so, then $E' = E$). (Question: A pseudocode for these algorithms).
**Elimination process: LexBFS**

```
<table>
<thead>
<tr>
<th>label</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>5</td>
</tr>
<tr>
<td>c</td>
<td>4</td>
</tr>
<tr>
<td>d</td>
<td>4</td>
</tr>
<tr>
<td>e</td>
<td>54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>label</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>43</td>
</tr>
<tr>
<td>d</td>
<td>54</td>
</tr>
<tr>
<td>e</td>
<td>54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>label</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>42</td>
</tr>
<tr>
<td>c</td>
<td>43</td>
</tr>
<tr>
<td>d</td>
<td>54</td>
</tr>
<tr>
<td>e</td>
<td>54</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>label</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>4</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
</tr>
</tbody>
</table>
```
Elimination process: Maximum cardinality search

Much simpler algorithm.

number the vertices from \( n \) to 1 in decreasing order.

As the next vertex to number, select the vertex adjacent to the largest number of previously numbered vertices; break the ties arbitrarily.

**Question (Class):** What is the running time complexity? Is searching the next \( j \) a problem?

---

**Maximum cardinality search.**

```plaintext
local \( j, v; \\
for \ i \in [0, n-1] \rightarrow \text{set (} i \text{):= } \emptyset \text{ rof}; \\
for \ v \in \text{vertices} \rightarrow \text{size (} v \text{):= 0; add } v \text{ to set (0) rof}; \\
i := n; j := 0; \\
do \ i \geq 1 \rightarrow \\
\quad v := \text{delete any from set (} j \text{);} \\
\quad \alpha (v) := i; \quad \alpha^{-1}(i) := v; \quad \text{size (} v \text{):= -1;} \\
\quad \text{for } \{v, w\} \in E \text{ such that size (} w \text{) } \geq 0 \rightarrow \\
\quad \quad \text{delete } w \text{ from set (size (} w \text{));} \\
\quad \quad \text{size (} w \text{):= size (} w \text{)+1;} \\
\quad \quad \text{add } w \text{ to set (size (} w \text{))} \\
\text{rof; } \\
i := i - 1; \\
j := j + 1; \\
do \ j \geq 0 \text{ and set (} j \text{) } = \emptyset \rightarrow j := j - 1 \text{ od} \\
of;
```

---
Elimination process: What if the graph is not triangulated?

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

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

In the graph of $L + L^T$: $l_{ij} \neq 0$ iff there is a path in $G(A, \alpha)$ with only vertices that come earlier than $i$ and $j$.

(Proof (Class)?)
Elimination process: What if the directed graph do not have peo?

Fill-path theorem [Rose, Tarjan,’78]

Let $G = (V, E, \alpha)$ be an ordered directed 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

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

In the directed graph of $L + U$: $\ell_{ij} \neq 0$ iff there is a path in $G(A, \alpha)$ with only vertices that come earlier than $i$ and $j$ (and $j$ comes earlier than $i$). Similar for the upper triangular part.
Fill-in

How significant is this?

× × × × ×
× ×
× ×
× × ×
× ×
Big question

For undirected graphs, we can recognize in $O(n + \tau)$ time if the given graph has a peo.

**Question (Class):** What about the directed graphs?

Refs for the question will be posted in the web.
Prove or disprove:

Let \( A \) be a nonsingular matrix with \( A = LU \) and irreducible.

Then every column in \( L \) has a nonzero below the diagonal.

Then every row in \( U \) has a nonzero to the right of the diagonal.
Question: LU and reducibility

Prove or disprove:

Let \( A \) be a nonsingular matrix with \( A = LU \) and irreducible.

Then every column in \( L \) has a nonzero below the diagonal.

Then every row in \( U \) has a nonzero to the right of the diagonal.

**Hint:** Use the directed graph version of the fill-path theorem.
(Class): When is the inverse of a sparse matrix sparse/full?
Question

(Class): When is the inverse of a sparse matrix sparse/full?

Hint: assume $A = LU$ is an LU factorization and consider obtaining the inverse with solves.
Question

(Class): When is the inverse of a sparse matrix sparse/full?
reducible, irreducible...
Suppose $W$ is a connected subgraph of $G$ and $\alpha$ is an elimination ordering.

Let

$$Z = \{ v : v \in \text{adj}(w) \text{ for some } w \in W \text{ and } \alpha^{-1}(v) > \max_{u \in W} \alpha^{-1}(u) \}$$

Then $Z \cup u$ where $u \in W$, and $\alpha^{-1}(u)$ is the largest in $W$ is a clique at the time we eliminate $u$. 