

Sparse LDL, LU, and Inverse Butterfly Factorization via Tree Decomposition

Edgar Solomonik

Siebel School of Computing and Data Sciences
University of Illinois Urbana-Champaign

Saint-Girons Conference

July 2, 2026

Complexity of Sparse Direct Solvers

Prior (most recent) state of the art in complexity of sparse direct solvers

Fast simulation of planar Clifford circuits

David Gosset^{1,2,3}, Daniel Grier^{1,4,5}, Alex Kerzner^{1,2}, and Luke Schaeffer^{1,2,6}

¹Institute for Quantum Computing, University of Waterloo, Canada

²Department of Combinatorics and Optimization, University of Waterloo, Canada

³Perimeter Institute for Theoretical Physics, Waterloo, Canada

⁴Cheriton School of Computer Science, University of Waterloo, Canada

⁵Department of Computer Science and Engineering and Department of Mathematics, University of California, San Diego, US

⁶Joint Center for Quantum Information and Computer Science, College Park, Maryland, US

D Linear System Solving

Theorem 26. *There exists an algorithm to find a uniformly random solution to $Ay = b$ (or report that no solution exists) in time $O(\|T\|_\omega)$ given a graph G with adjacency matrix $A \in \mathbb{F}_2^{n \times n}$, a vector $b \in \mathbb{F}_2^n$, and a nice tree decomposition T for G .*

Proof. Construct the circuit $\mathcal{C}$ associated with G and its tree decomposition (in time linear in $\|T\|_1$). Claim 21 says that $\mathcal{C}|+\rangle^{\otimes n}$ is a graph state $|G'\rangle$ for $G' = (V', E')$ which

Treewidth

A tree decomposition of G specifies a set of bags (subsets of vertices of G) and a tree over these bags, such that

- ① each vertex is assigned to at least one bag
- ② the set of bags to which any vertex is assigned is connected in the tree
- ③ the end-points of any edge in G are contained within some bag

The treewidth of a graph G is

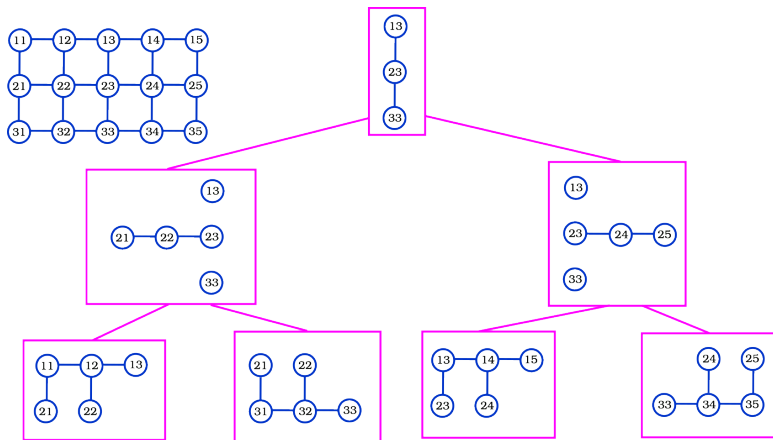
$$\text{tw}(G) = \min_{\text{tree decompositions of } G} \max_{\text{bags in tree decomposition}} \text{bag size}$$

Equivalently, if A is a generic SPD matrix associated with graph G

$$\text{tw}(G) = \min_{\text{elimination orders}} \max_{\text{columns}} \text{number of nonzeros in column of } L$$

To ensure $\text{tw}(G) = 1$ if G is a tree, the conventional definition subtracts 1.

Tree Decomposition by Nested Dissection



A Tree Decomposition Canonical Form

For any graph G , with $\text{tw}(G) = \tau$, there exists a tree decomposition that

- is a complete binary tree
- has height $O(\log(n/\tau + 1))$
- has width (max bag size) $O(\tau)$

See previously mentioned paper by Grier et al. and Hans L. Bodlaender and Torben Hagerup, *Parallel Algorithms with Optimal Speedup for Bounded Treewidth*, SIAM Journal on Computing 1998.

Complexity of Cholesky Based on Treewidth

It is straightforward to implement Cholesky given a tree decomposition

- Eliminate vertices assigned only to leaf bags (fill limited to pairs of vertices contained in bag)
- Discard the leaf bags and recurse

Complexity is

- for a general tree decomposition T ,

$$\|T\|_{\omega}^{\omega} = \sum_{B \in T} |B|^{\omega}$$

where ω is the matrix multiplication exponent

- using the canonical complete binary tree form,

$$O(n\tau^{\omega-1})$$

where n is the matrix dimension and τ is treewidth

What About Indefinite Systems?

- The LDL decomposition of a symmetric matrix $A \in \mathbb{F}^{n \times n}$ is

$$P^T A P = L D L^T$$

where L is lower triangular and unit-diagonal, D is block diagonal with blocks that are either 1×1 or 2×2 and antidiagonal.

- If $A = A^T$, we have $D = D^T$, while for $A^T = -A$, we would require $D^T = -D$.
- The LDL decomposition can hence preserve a variety of symmetries
 - a complex Hermitian or skew-Hermitian matrix
 - a real symmetric or skew-symmetric matrix
 - a symmetric matrix over $\mathbb{F}_2$

LDL Factorization from Elementary Eliminations

- **Vertex elimination**, if $a_{11} \neq 0$ compute

$$A - a_{:,1} a_{11}^{-1} a_{:,1}^T$$

- **Edge elimination**, if $a_{21} \neq 0$, and $a_{11} = 0$ or $a_{22} = 0$, eliminate $(1, 2)$,

$$A - \begin{bmatrix} a_{:,1} & a_{:,2} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} \begin{bmatrix} a_{:,1} & a_{:,2} \end{bmatrix}^T$$

Dense LDL Factorization

Consider a naive recursive LDL algorithm for

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

- Recursively factorize $P_1 A_{11} P_1^T = L_1 D_1 L_1^T$
- The recursive subproblem can eliminate $\text{rank}(A_{11})$ vertices
- Update the full A to obtain

$$\begin{bmatrix} 0 & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix}$$

where $\tilde{A}_{12}$ has $\text{rank}(A_{11})$ fewer rows than A_{12}

Null-Space Method for Saddle-Point Systems

Given

$$M = \begin{bmatrix} A & B \\ B^T & 0 \end{bmatrix}$$

- Perform pivoted (rank-revealing) LU of B
- Each elimination step of LU of B is an LDL edge elimination on A and preserves the saddle point structure of M
- Given LU of B , can construct partial LDL of M
- Appropriate efficient dense LU algorithms exist (Ibarra, Moran, and Hui, 1981), (Jeannerod 2006)

Sparse LDL Algorithm

Given a tree decomposition of A with treewidth τ

- can eliminate vertices in a leaf bag using an edge elimination, but if other endpoint is not contained only in leaf bag, fill is not controlled
- so, we apply vertex elimination or edge eliminations so long as both endpoints are in the bag, and pass the rest to the parent bag / recursive subproblem
- to keep the number of passed vertices bounded by τ , peel linearly dependent rows/columns in off-diagonal saddle-point block

Yields implicit LDL factorization and explicit factorization of leading (after permutation) $\text{rank}(A) \times \text{rank}(A)$ block with complexity

$$O(\|T\|_{\omega}^{\omega}) \quad \text{or} \quad O(n\tau^{\omega-1})$$

Complementary Low Rank of Matrix Inverse

- A symmetric matrix has complementary rank r if we can arrange the vertices in a tree, such that the off-diagonal block separating a subtree from the remainder of the tree has rank r
- Any bag $\mathcal{B}$ in a tree decomposition of G is a vertex separator of G
- Choose $\mathcal{B}$ as the root of the tree decomposition, then the LDL factorization gives

$$A^{-1} = \begin{bmatrix} L_{11}^{-T} D_1^{-1} L_{11}^{-1} & 0 & 0 \\ 0 & L_{22}^{-T} D_2^{-1} L_{22}^{-1} & 0 \\ 0 & 0 & 0 \end{bmatrix} + U^T D_3^{-1} U,$$
$$U = \begin{bmatrix} -L_{33}^{-1} L_{31} L_{11}^{-1} & -L_{33}^{-1} L_{32} L_{22}^{-1} & L_{33}^{-1} \end{bmatrix}.$$

- Hence, if the treewidth of the graph of A is τ , the complementary rank of A^{-1} is $O(\tau)$

Butterfly Decomposition of Matrix Inverse

- A butterfly decomposition of rank r of a matrix M of dimension n is

$$M = B_1 \cdots B_k$$

where $k = O(\log n)$ and each B_i contains $O(nr)$ nonzeros (block sparse with $r \times r$ blocks)

- Consider the LDL of A as on the prior slide, and pick $\mathcal{B}$ as the root of a complete binary tree decomposition, then the L factor satisfies,

$$L^{-1} = \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ 0 & 0 & L_{33}^{-1} \end{bmatrix} \begin{bmatrix} I & 0 & 0 \\ 0 & I & 0 \\ -L_{31} & -L_{32} & I \end{bmatrix} \begin{bmatrix} L_{11}^{-1} & 0 & 0 \\ 0 & L_{22}^{-1} & 0 \\ 0 & 0 & I \end{bmatrix}$$

- Expand L_{11}^{-1} and L_{22}^{-1} the same way recursively and jointly to get rank $O(\tau)$ butterfly decomposition of A^{-1}

Conclusions and Open Questions

- Obtained relatively simple sparse LDL algorithm with complexity $O(n\tau^{\omega-1})$ for treewidth τ
- For low-rank matrices, it is unclear how to obtain this complexity, but can get implicit form
- Results extend to LU by off-diagonal embedding, $\begin{bmatrix} 0 & B \\ B^T & 0 \end{bmatrix}$
- Tree decomposition and LDL enables butterfly factorization of inverse
- Stability? HSS rank of inverse? High-performance implementation?

Sparse LDL, LU, and Inverse Butterfly Factorization via Tree Decomposition, arXiv:2504.20305

See also, Clifford circuit simulation via solving linear system

Pang, Yuchen, and E.S. *Simulating Clifford Circuits with Gaussian Elimination*. arXiv:2511.06127