Chordal graphs and sparse semidefinite optimization

Lieven Vandenberghe

Electrical Engineering Department

University of California, Los Angeles

Linköping University

August 29–31, 2016

Semidefinite program (SDP)

$$
\begin{array}{ll}
\text{minimize} & \mathbf{tr}(CX) \\
\text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\
& X \succeq 0
\end{array}
$$

variable X is $n \times n$ symmetric matrix; $X \succeq 0$ means X is positive semidefinite

- matrix inequalities arise naturally in many areas (*e.g.*, control, statistics)
- used in convex modeling systems (CVX, YALMIP, CVXPY, ...)
- relaxations of nonconvex quadratic and polynomial optimization

- in many applications the coefficients A_i, C are sparse
- optimal X is typically dense, even for sparse A_i, C

Power flow optimization

an optimization problem with non-convex quadratic constraints

Variables

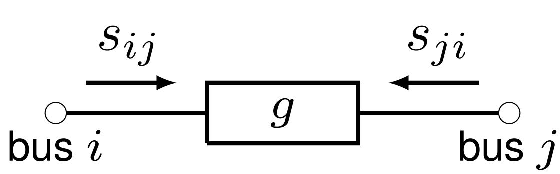
- complex voltages v_i at each node (bus) of the network
- complex power flow s_{ij} entering the link (line) from node i to node j

Non-convex constraints

- (lower) bounds on voltage magnitudes

$$v_{\min} \leq |v_i| \leq v_{\max}$$

- flow balance equations:



$$s_{ij} + s_{ji} = \bar{g}_{ij}|v_i - v_j|^2$$

g_{ij} is admittance of line from node i to j

Semidefinite relaxation of optimal power flow problem

- introduce matrix variable $X = \mathrm{Re}(vv^H)$, *i.e.*, with elements $X_{ij} = \mathrm{Re}\,(v_i \bar{v}_j)$
- voltage bounds and flow balance equations are convex in X:

$$v_{\min} \leq |v_i| \leq v_{\max} \quad \longrightarrow \quad v_{\min}^2 \leq X_{ii} \leq v_{\max}^2$$

$$s_{ij} + s_{ji} = \bar{g}_{ij}|v_i - v_j|^2 \quad \longrightarrow \quad s_{ij} + s_{ji} = \bar{g}_{ij}(X_{ii} + X_{jj} - 2X_{ij})$$

- replace constraint $X = \mathrm{Re}(vv^H)$ with weaker constraint $X \succeq 0$
- relaxation is exact if optimal X happens to have rank two

Sparsity in relaxation:

off-diagonal X_{ij} appears in constraints only if there is a line between buses i and j

(Jabr 2006, Bai et al. 2008, Lavaei and Low 2012, ...)

Modeling software

Convex modeling systems (CVX, YALMIP, CVXPY, ...)

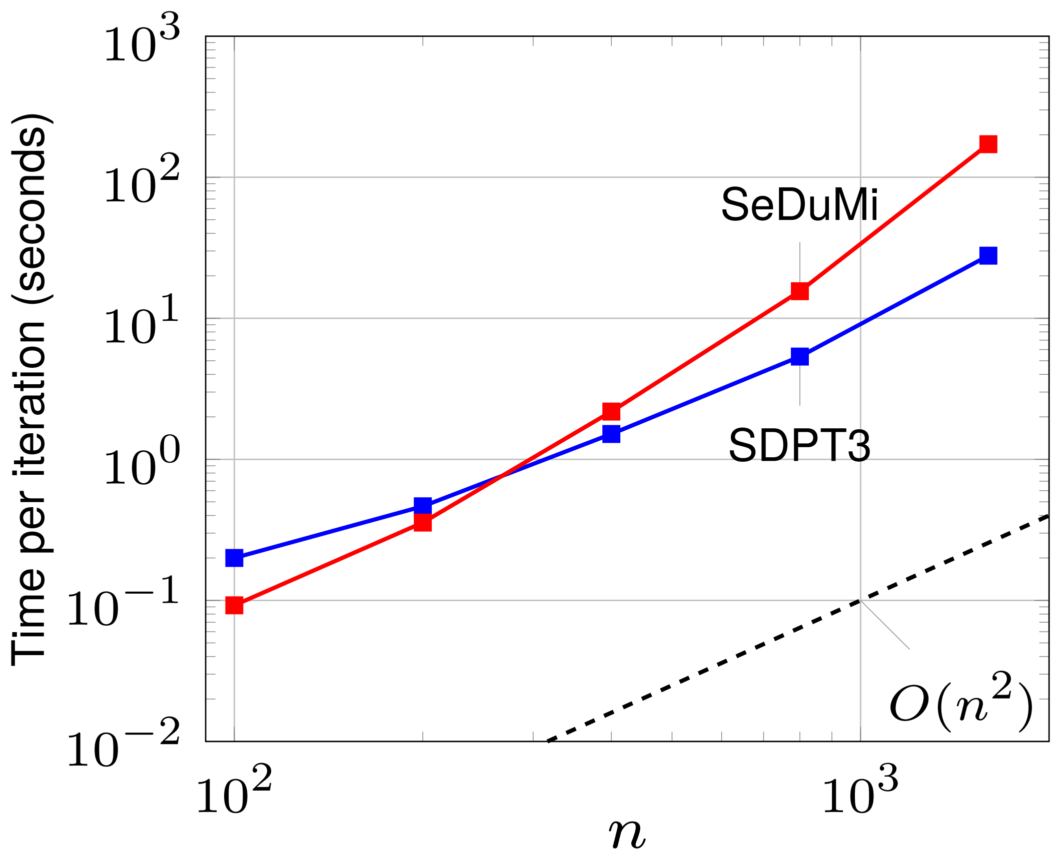
- convert problems stated in standard mathematical notation to conic LPs
- choice of cones is limited by available algorithms and solvers

General-purpose solvers (SDPT3, Sedumi, SDPA, CSDP, DSDP, ...)

- handle three symmetric cones (linear, quadratic, semidefinite)
- sufficiently general for most convex problems encountered in practice
- reformulation often leads to large, sparse SDPs
- large differences in (linear algebra) complexity between three cones

SDP with band structure

cost of solving SDP with banded matrices (half bandwidth $w = 5$, 100 constraints)

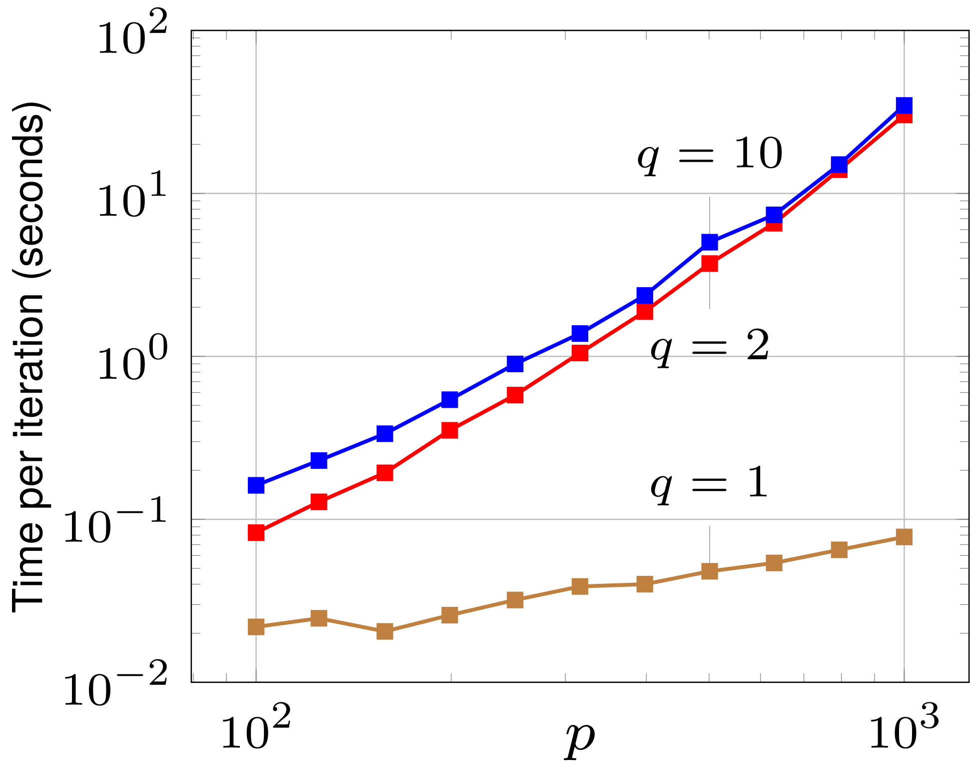


- for $w = 0$ (linear program), cost/iteration is linear in n
- for $w > 0$, cost grows as n^2 or faster

(from Andersen, Dahl, Vandenberghe 2010)

Matrix norm minimization

$$\begin{array}{ll} \text{minimize} & \|F_0 + x_1F_1 + \cdots + x_mF_m\|_2 + c^Tx \\ \text{subject to} & 0 \preceq x \preceq \mathbf{1} \end{array}$$



CVX/SeDuMi

matrices F_i of size $p \times q$

- $q = 1$: solved as second-order cone program
- $q > 1$: semidefinite program with $(p+q) \times (p+q)$ 'block-arrow' sparsity

Trace norm minimization

$$\begin{array}{ll} \text{minimize} & \|Y\|_* \\ \text{subject to} & \text{convex constraints on } Y \end{array}$$

- $\|Y\|_*$ is sum of singular values (trace norm or nuclear norm)
- popular as a convex optimization method for finding low rank solutions

SDP formulation

$$\begin{array}{ll} \text{minimize} & (\mathbf{tr}\, U + \mathbf{tr}\, V)/2 \\ \text{subject to} & \left[\begin{array}{cc} U & Y \\ Y^T & V \end{array} \right] \succeq 0 \\ & \text{convex constraints on } Y \end{array}$$

- for larger Y, expensive to solve using general-purpose SDP solvers
- except for matrix inequality, only diagonal entries of U, V are needed

Exploiting sparsity

1. **Symmetric primal-dual interior-point methods**

exploit sparsity when forming 'Schur complement' equations

2. **Non-symmetric interior-point methods** (matrix completion methods)

(Fukuda et al. 2000, Burer 2003, Srijuntongsiri et al. 2004, Andersen et al. 2010)

3. **Decomposition** (combined with interior-point or first-order methods)

(Fukuda et al. 2000, Nakata et al. 2003, Kim et al. 2011, Sun et al. 2014, . . .)

we will discuss approaches 2 and 3

Chordal graphs

chordal graphs have been studied in many disciplines since the 1960s

- linear algebra (sparse factorization, matrix completion problems)
- combinatorial optimization (a class of 'perfect' graphs)
- machine learning (graphical models, Euclidean distance matrices)
- nonlinear optimization (partial separability)
- computer science (database theory)

first used in semidefinite optimization by Fujisawa, Kojima, Nakata (1997)

References

The course material is from the survey paper

> L. Vandenberghe, M. S. Andersen, *Chordal Graphs and Semidefinite Optimization*, Foundations and Trends in Optimization, 2015.
>
> `www.seas.ucla.edu/~vandenbe/publications/chordalsdp.pdf`

Software is available at

`github.com/cvxopt/chompack`

Outline

I. Graph theory

- chordal graphs
- tree representations
- graph elimination

II. Sparse matrices

- sparse positive semidefinite matrices
- positive semidefinite completion
- Euclidean distance matrices

III. Optimization

- partial separability
- decomposition
- sparse semidefinite optimization

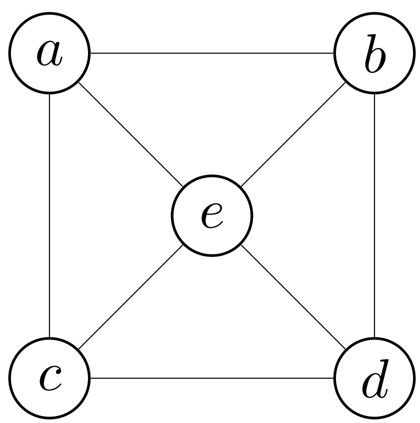
I. Chordal graphs

- undirected graphs
- origins
- definition
- clique trees
- perfect elimination
- elimination trees
- supernodes
- graph elimination

Undirected graph

$$G = (V, E)$$

- V is a finite set of **vertices**
- $E \subseteq \{\{v, w\} \mid v, w \in V\}$ is the set of **edges**
- vertices v and w are **adjacent** if $\{v, w\} \in E$
- the **neighborhood** $\mathrm{adj}(v)$ of vertex v is the set of vertices adjacent to v



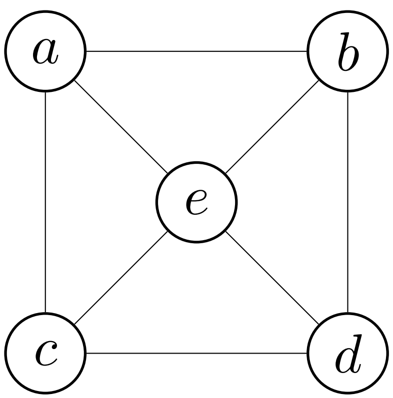
- vertices: $V = \{a, b, c, d, e\}$
- edges: $E = \{\{a, b\}, \{a, c\}, \{a, e\}, \ldots\}$
- neighborhood of a: $\mathrm{adj}(a) = \{b, c, e\}$

Subgraphs and cliques

the **subgraph** (induced by) $W \subset V$ is

$$G(W) = (W, E(W)), \qquad E(W) = \{\{v,w\} \in E \mid v,w \in W\}$$

- a subgraph W is **complete** if $E(W) = \{\{v,w\} \mid v,w \in W\}$
- we will use the term **clique** to mean *maximal* complete subgraph



- subgraph (induced by) $W = \{a,b,c,d\}$:

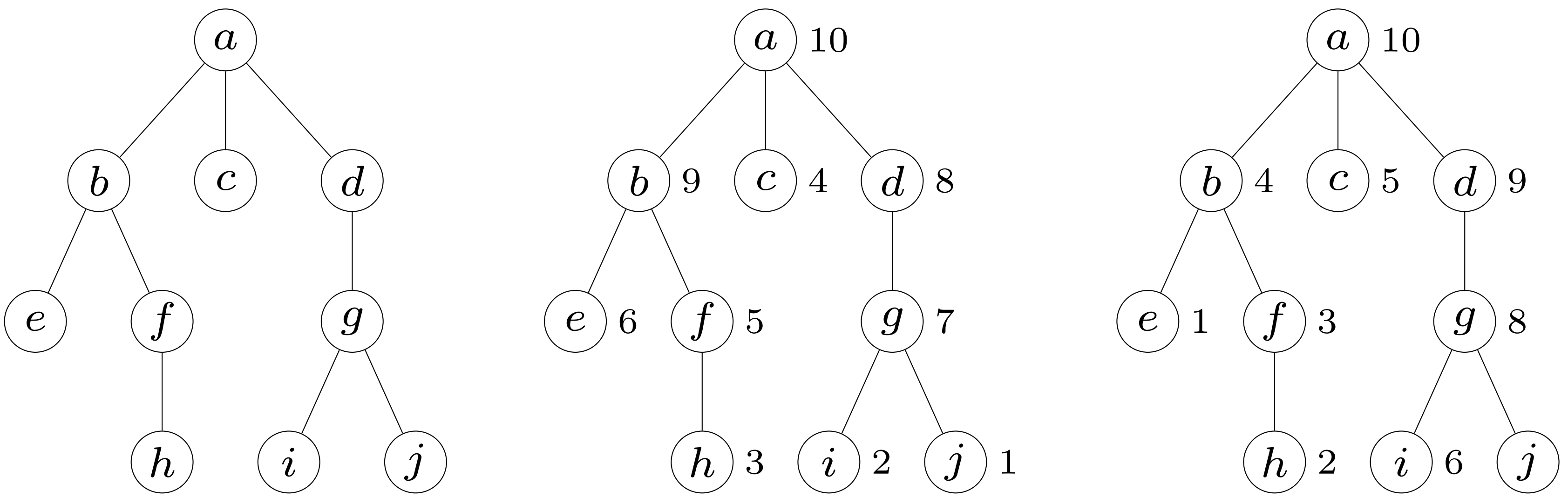
$$E(W) = \{\{a,b\},\{b,d\},\{c,d\},\{a,c\}\}$$

- $W = \{a,b,e\}$ is a clique
- $W = \{a,b\}$ is complete but not a clique

Rooted tree

connected, acyclic graph with one vertex designated as root

- parent of vertex v is denoted $p(v)$
- ancestors are denoted $p^k(v)$: $p^1(v) = p(v)$, $p^2(v) = p(p(v))$, ...
- topological ordering: parent follows its children
- postordering: topological, descendants of each vertex numbered consecutively



rooted tree | a topological ordering | a postordering

Symmetric sparsity pattern

undirected graphs will be used to represent symmetric sparsity patterns

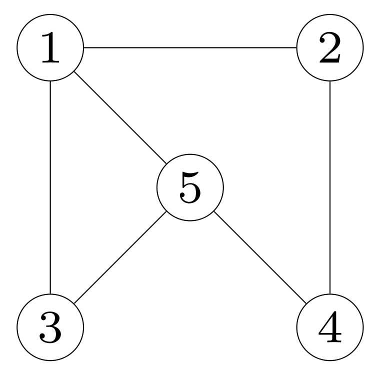
- $n \times n$ pattern is represented by graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$
- symmetric matrix A of order n has the sparsity pattern E if

$$i \neq j, \quad \{i, j\} \notin E \qquad \Longrightarrow \qquad A_{ij} = A_{ji} = 0$$

  entries A_{ij} with $i = j$ or $\{i, j\} \in E$ may or may not be zero

- E is not unique (unless all off-diagonal entries of A are nonzero)
- cliques of G correspond to maximal 'dense' principal submatrices

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & A_{15} \\ A_{21} & A_{22} & 0 & A_{24} & 0 \\ A_{31} & 0 & A_{33} & 0 & A_{35} \\ 0 & A_{42} & 0 & A_{44} & A_{45} \\ A_{51} & 0 & A_{53} & A_{54} & A_{55} \end{bmatrix}$$



I. Chordal graphs

- undirected graphs
- **origins**
- definition
- clique trees
- perfect elimination
- elimination trees
- supernodes
- graph elimination

Combinatorial properties of graphs

Clique number $\omega(G)$: size of largest clique

Clique cover number $\bar{\chi}(G)$: minimum number of cliques needed to cover V

Stable set number $\alpha(G)$

- a subset $W \subseteq V$ is a stable (independent) set if no vertices in W are adjacent
- stable sets of G are complete subgraphs of the complementary graph
- stable set number $\alpha(G)$ is the size of the largest stable set
- upper bounded by clique cover number: $\alpha(G) \leq \bar{\chi}(G)$

Coloring number $\chi(G)$

- a vertex coloring is a partitioning of V in stable sets
- coloring number $\chi(G)$: minimum number of stable sets in a vertex coloring
- lower bounded by clique number: $\chi(G) \geq \omega(G)$

Shannon zero-error capacity of a communication channel

- interpret vertices of $G = (V, E)$ as symbols
- edges E connect symbols that can be confused during transmission
- define graph $G^k = (V^k, E^k)$: vertices are words of k symbols from V
- edges E^k connect words that can be confused:

$$\{v_1 v_2 \cdots v_k,\ w_1 w_2 \cdots w_k\} \in E^k \qquad \Longleftrightarrow \qquad \forall i : v_i = w_i \text{ or } \{v_i, w_i\} \in E$$

- a stable set of G^k is a set of words of length k that cannot be confused

Zero-error capacity (Shannon 1956)

$$\Theta(G) = \sup_k \alpha(G^k)^{1/k}$$

$\alpha(G^k)$ is stable set number of G^k

Shannon capacity and chordal graphs

Bounds on Shannon capacity:

$$\alpha(G) \le \Theta(G) \le \bar{\chi}(G)$$

Perfect graphs (Berge 1963, Lovász 1972)

- graph and all subgraphs satisfy $\alpha(G) = \bar{\chi}(G)$ (as well as $\omega(G) = \chi(G)$)
- definition was inspired by Shannon's paper

Chordal graphs

- an important class of perfect graphs
- simple greedy algorithms compute $\alpha(G)$, $\bar{\chi}(G)$, $\omega(G)$, $\chi(G)$ (Gavril 1972)
- for general graphs, computing any of these quantities is NP-complete

Shannon capacity and semidefinite optimization

Lovász bound on Shannon capacity (Lovász 1979)

$$\begin{array}{ll} \text{minimize} & \lambda_{\max}(S) \\ \text{subject to} & S_{ii} = 1, \quad i = 1, \ldots, n \\ & S_{ij} = S_{ji} = 1, \quad \{i,j\} \notin E \end{array}$$

- optimal value is upper bound on $\Theta(G)$
- an early application of semidefinite relaxation
- can be expressed as a sparse SDP:

$$\begin{array}{ll} \text{minimize} & 1 + (1/n)\,\mathbf{tr}\,X \\ \text{subject to} & X_{11} = X_{22} = \cdots = X_{nn} \\ & X_{ij} = X_{ji} = -1, \quad \{i,j\} \notin E \\ & X \succeq 0 \end{array}$$

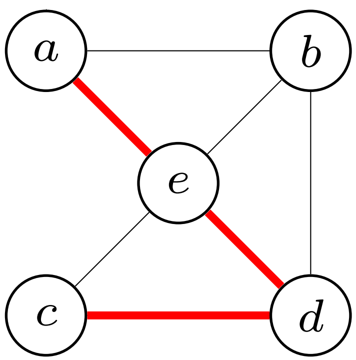
I. Chordal graphs

- undirected graphs
- origins
- **definition**
- clique trees
- perfect elimination
- elimination trees
- supernodes
- graph elimination

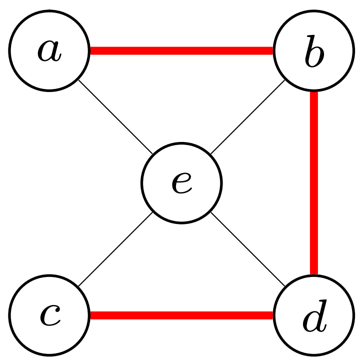
Chorded paths and cycles

a **chord** is an edge between non-consecutive vertices in a path or cycle

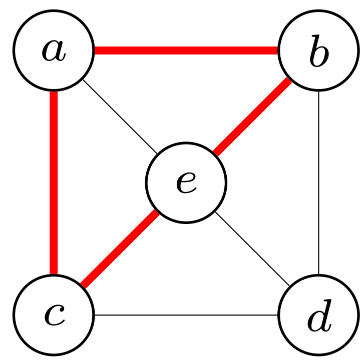
- a one-edge 'shortcut' in a path or cycle
- all shortest paths are chordless



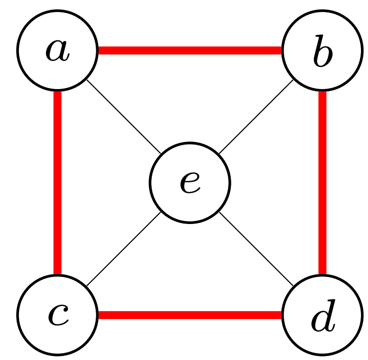
chorded path



chordless path



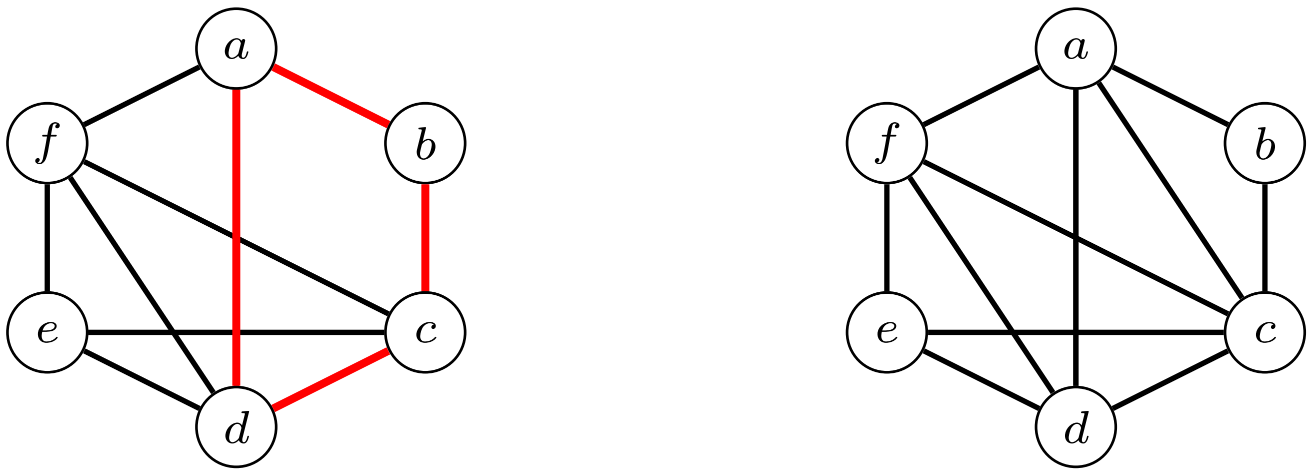
chorded cycle



chordless cycle

Chordal graph

Chordal graph: every cycle of length greater than three has a chord



not chordal chordal

- using chords to take 'shortcuts', all cycles can be reduced to triangles
- subgraphs of chordal graphs are chordal

also known as rigid circuit graphs, triangulated graphs, decomposable graphs, ...

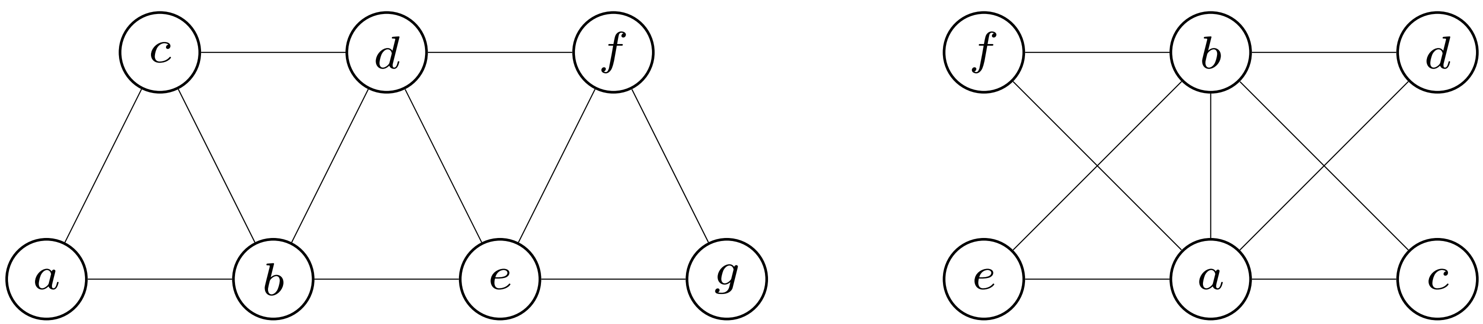
Examples

Trivial: complete graphs, trees, cactus graphs (no cycles of length > 3)

k**-trees:** constructed recursively

- k-tree with k vertices is complete graph
- to construct k-tree with $n + 1$ vertices from k-tree with n vertices:

  make new vertex adjacent to a complete subgraph of k vertices

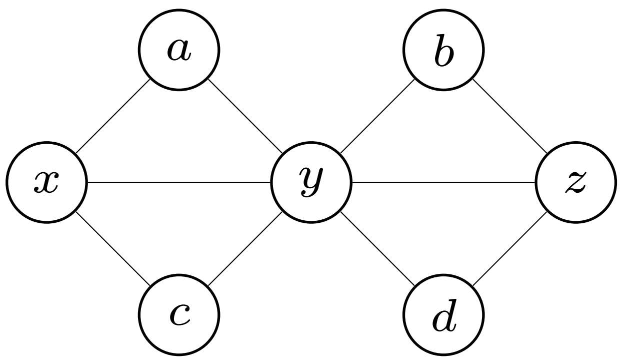


two 2-trees

Minimal vertex separator

Definition: $S \subset V$ is a **minimal** vw**-separator** if

- v and w are in different connected components of $G(V \setminus S)$
- no strict subset of S is a vw-separator



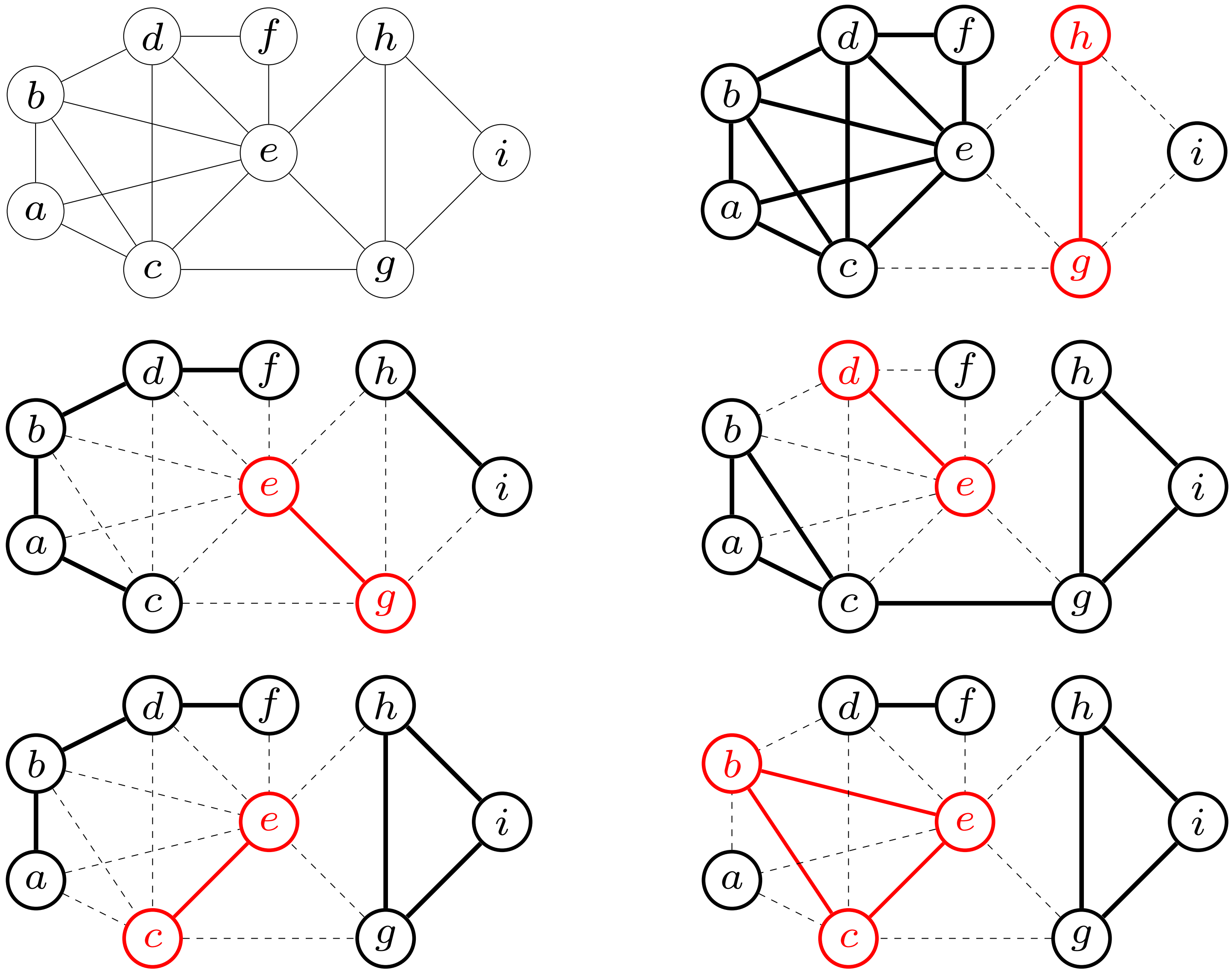
- $\{x, y\}$ is a minimal ac-separator
- $\{y\}$ is a minimal ad-separator

Chordal graphs (Dirac 1961, Buneman 1974)

- a graph is chordal if and only if all minimal vertex separators are complete
- every minimal vertex separator is a subset of at least two cliques

Example

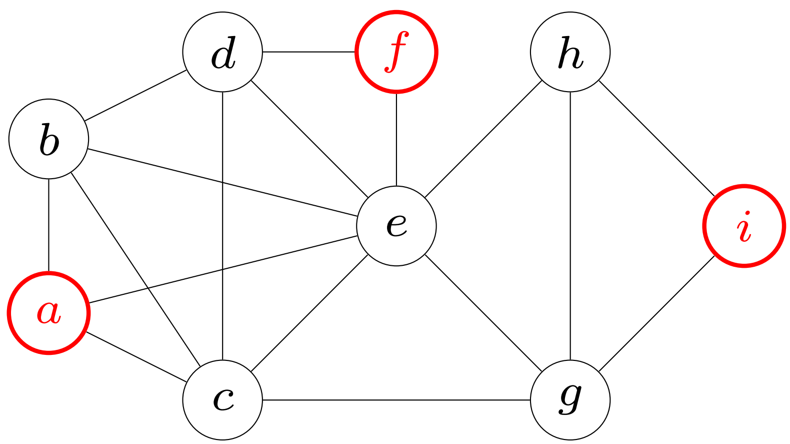
a chordal graph and all its minimal vertex separators



Simplicial vertices

Definition: a vertex v is **simplicial** if $\mathrm{adj}(v)$ is complete

- closed neighborhood $\{v\} \cup \mathrm{adj}(v)$ is a clique
- $\{v\} \cup \mathrm{adj}(v)$ is the only clique that contains v



three simplicial vertices

Chordal graphs (Dirac 1961)

a non-complete chordal graph has at least two non-adjacent simplicial vertices

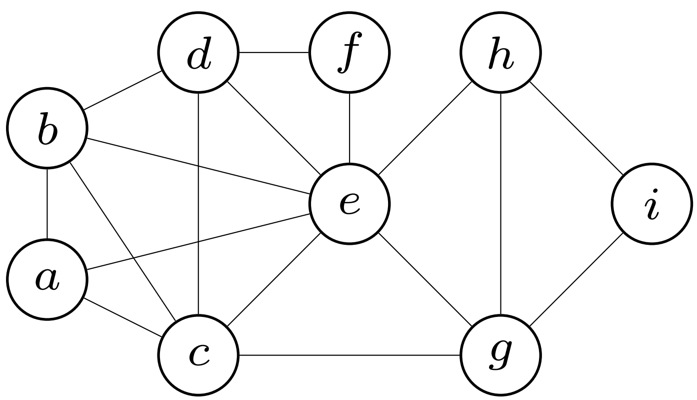
I. Chordal graphs

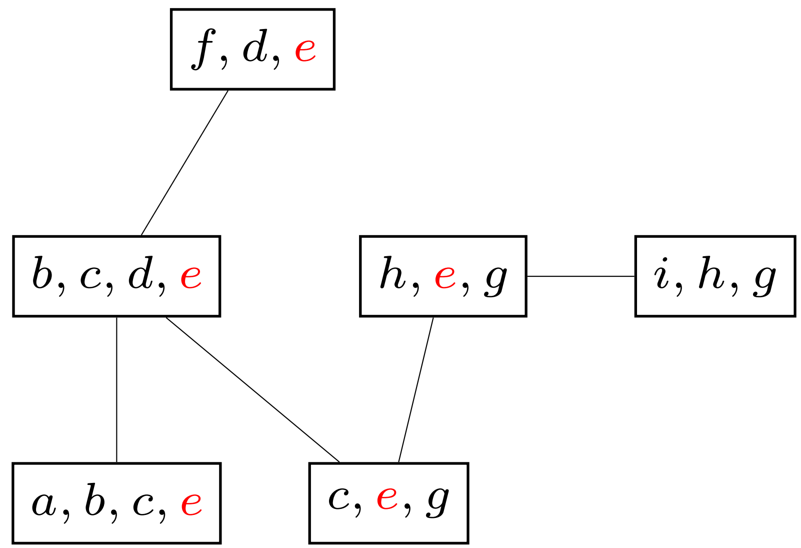
- undirected graphs
- origins
- definition
- **clique trees**
- perfect elimination
- elimination trees
- supernodes
- graph elimination

Clique tree

Definition: clique tree with the **induced subtree property** for $G = (V, E)$

- vertices of clique tree are the cliques of G
- for every $v \in V$, the cliques that contain v form a subtree of the clique tree





Chordal graphs (Buneman 1974, Gavril 1974)

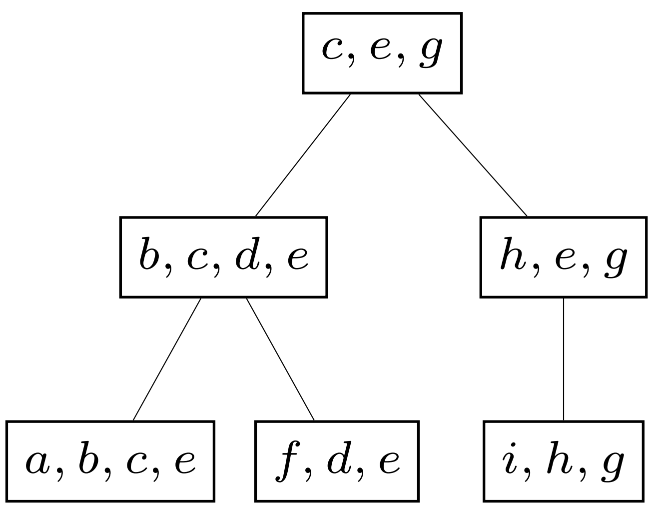
G is chordal if and only it has a clique tree with induced subtree property

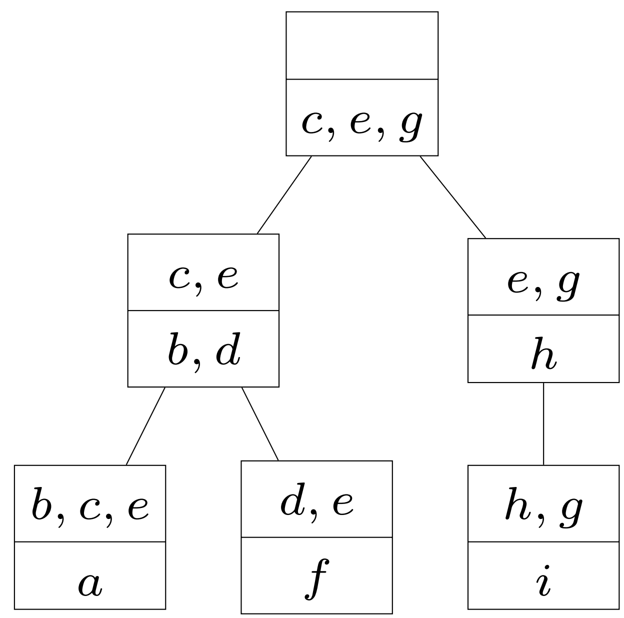
Clique separators and residuals

- choose any clique as root of the clique tree; denote parent function as $p_c(W)$
- clique **separator** and **residual** of non-root clique W are defined as

$$\mathrm{sep}(W) = W \cap p_c(W), \qquad \mathrm{res}(W) = W \setminus \mathrm{sep}(W)$$

for the root clique, $\mathrm{sep}(W) = \emptyset$ and $\mathrm{res}(W) = W$

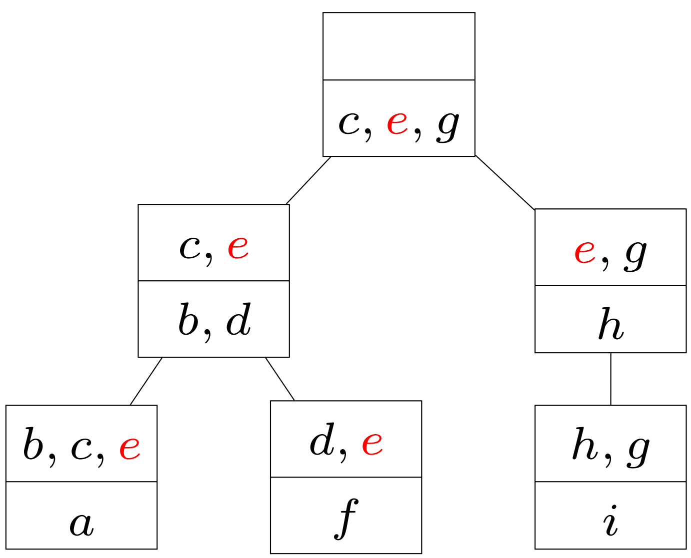


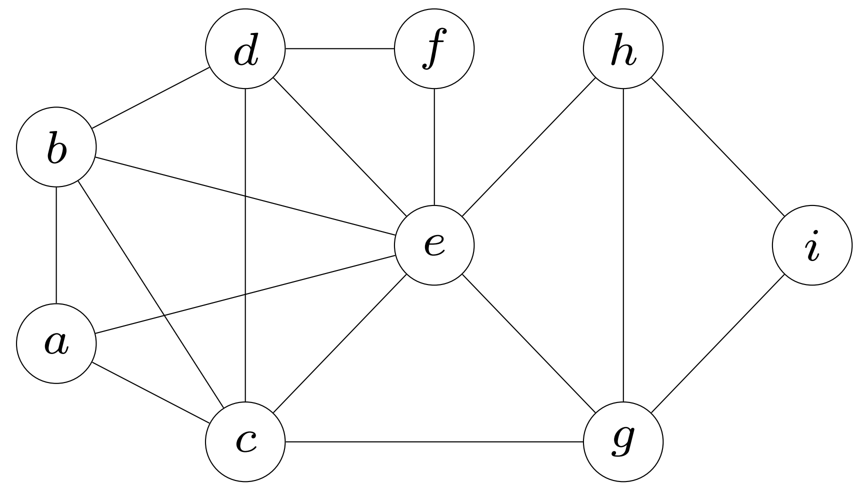


$$W = \{b, c, d, e\}, \qquad \mathrm{res}(W) = \{b, d\}, \qquad \mathrm{sep}(W) = \{c, e\}$$

Graph structure from rooted clique tree

- every vertex v belongs to exactly one clique residual $\mathrm{res}(W)$
- if $v \in \mathrm{res}(W)$ then W is the root of the subtree of cliques that contain v
- the clique separators $\mathrm{sep}(W)$ are the minimal vertex separators of the graph
- a vertex is simplicial if it does not belong to any clique separator





a chordal graph has at most $n = |V|$ cliques, $n - 1$ minimal vertex separators

Tree intersection graphs

Definition: given a family of subtrees $\{R_v \mid v \in V\}$ of a tree T

- tree intersection graph $G = (V, E)$ has vertex set V
- $\{v, w\} \in E$ if and only if R_v and R_w intersect

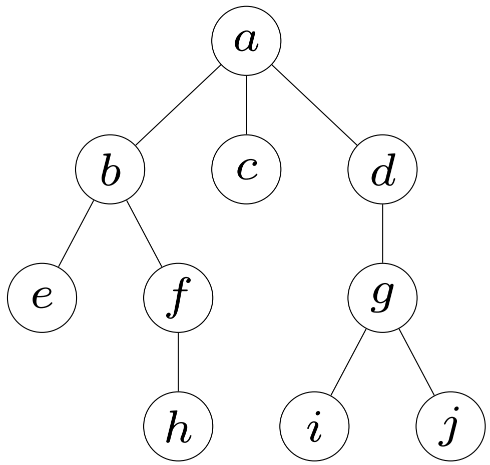
Chordality (Gavril 1974, Buneman 1974)

- a tree intersection graph is chordal
- every chordal graph can be represented as a tree intersection graph

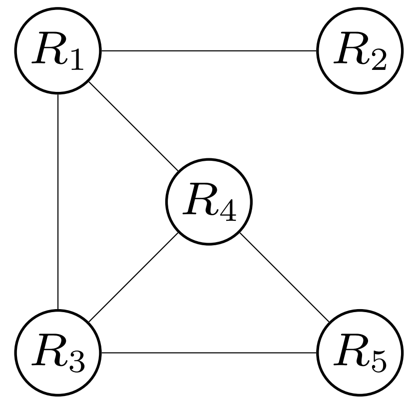
  (for example, T is the clique tree, R_v subtree of cliques that contain v)

Example

tree T

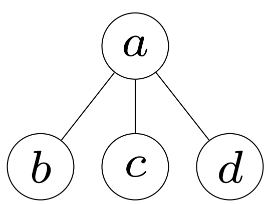


tree intersection graph

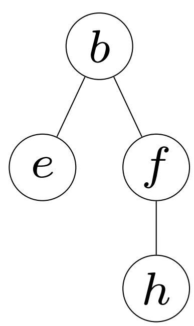


five subtrees of T

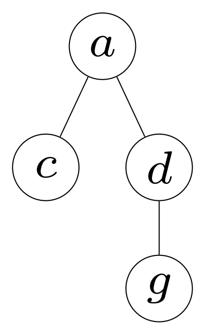
R_1



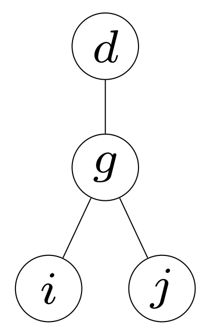
R_2



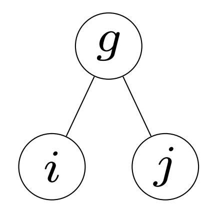
R_3



R_4



R_5



Representing chordal graphs as tree intersection graph

Clique trees

- vertices of T are (maximal) cliques
- R_v is subtree of cliques that contain v

Junction tree (join tree)

- used in machine learning and artificial intelligence
- vertices of T are complete subgraphs (not necessarily maximal)

Elimination tree

- used in sparse matrix algorithms
- discussed later

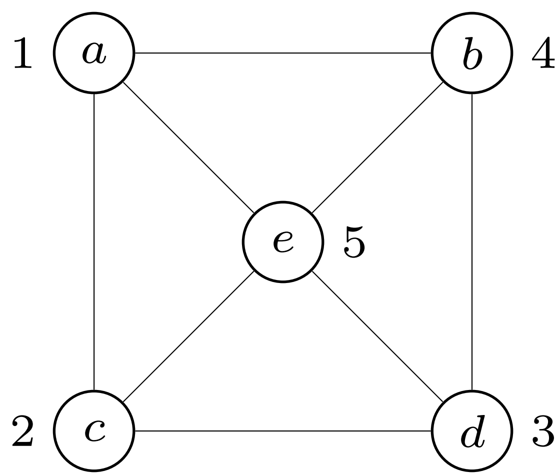
I. Chordal graphs

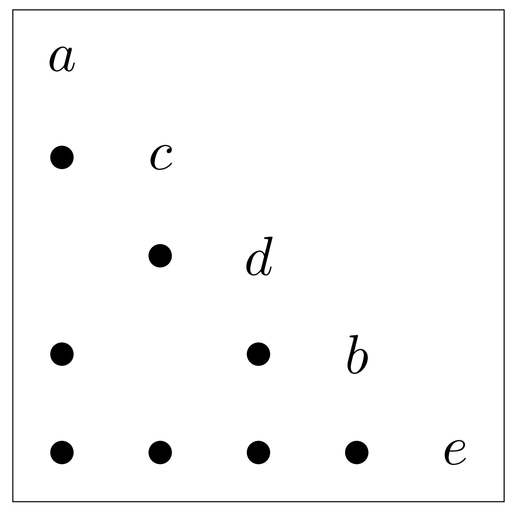
- undirected graphs
- origins
- definition
- clique trees
- **perfect elimination**
- elimination trees
- supernodes
- graph elimination

Ordered undirected graphs

$$G_\sigma = (V, E, \sigma)$$

- σ is a bijection from $\{1, 2, \ldots, |V|\}$ to V
- ordering notation: $v \prec w$ means $\sigma^{-1}(v) < \sigma^{-1}(w)$





can be represented as annotated graph or triangular array

Monotone neighborhoods

- higher and lower (monotone) neighborhoods

$$\mathrm{adj}^+(v) = \mathrm{adj}(v) \cap \{w \mid w \succ v\}, \qquad \mathrm{adj}^-(v) = \mathrm{adj}(v) \cap \{w \mid w \prec v\}$$

- the sizes of these sets are called higher and lower degrees:

$$\deg^+(v) = |\mathrm{adj}^+(v)|, \qquad \deg^-(v) = |\mathrm{adj}^-(v)|$$

- closed higher and lower neighborhoods

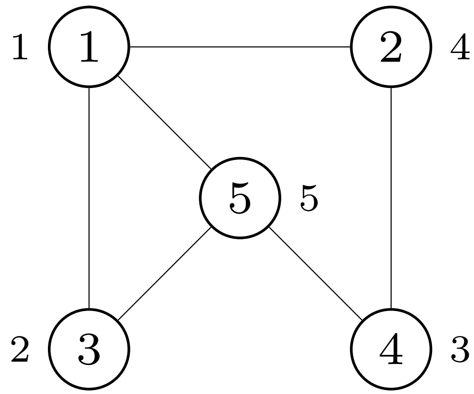
$$\mathrm{col}(v) = \{v\} \cup \mathrm{adj}^+(v), \qquad \mathrm{row}(v) = \{v\} \cup \mathrm{adj}^-(v)$$

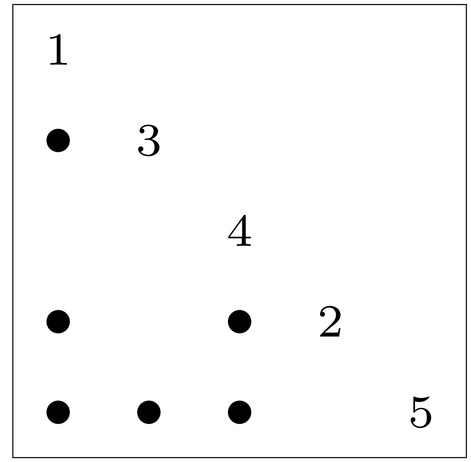
$$\boxed{\begin{array}{ccccc} a & & & & \\ \bullet & c & & & \\ & \bullet & d & & \\ \bullet & & \bullet & b & \\ \bullet & \bullet & \bullet & \bullet & e \end{array}}$$

$$\mathrm{adj}^+(c) = \{d, e\}, \quad \mathrm{adj}^-(c) = \{a\}$$
$$\deg^+(c) = 2, \quad \deg^-(c) = 1$$
$$\mathrm{col}(c) = \{c, d, e\}, \quad \mathrm{row}(c) = \{c, a\}$$

Example: ordered symmetric sparsity pattern

- ordered sparsity pattern (V, E, σ) of order 5 with $\sigma = (1, 3, 4, 2, 5)$





- represents symmetric reordering (P_σ is permutation matrix defined by σ)

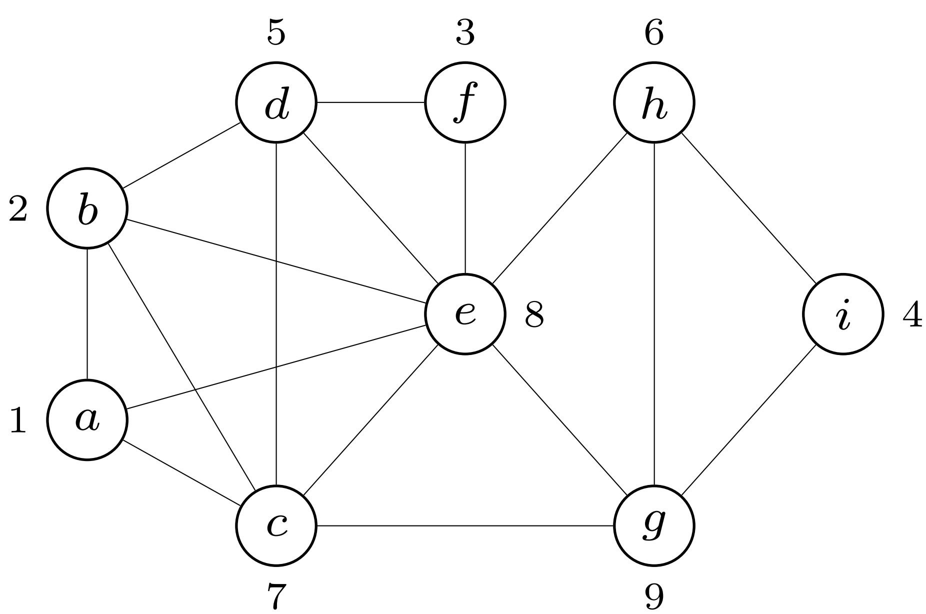
$$P_\sigma A P_\sigma^T = \left[\begin{array}{ccccc} A_{11} & A_{31} & 0 & A_{21} & A_{51} \\ A_{31} & A_{33} & 0 & 0 & A_{53} \\ 0 & 0 & A_{44} & A_{42} & A_{54} \\ A_{21} & 0 & A_{42} & A_{22} & 0 \\ A_{51} & A_{53} & A_{54} & 0 & A_{55} \end{array}\right]$$

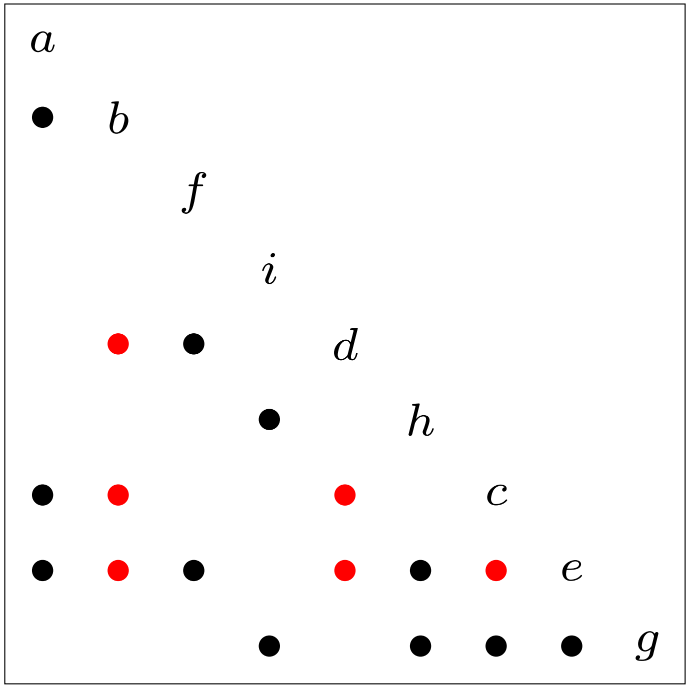
Filled graph

an ordered undirected graph is **filled** or **monotone transitive** if

$$w, z \in \mathrm{adj}^+(v) \qquad \Longrightarrow \qquad \{w, z\} \in E$$

the higher neighborhood of every vertex is complete





Perfect elimination ordering

σ is a **perfect elimination ordering** for (V, E) if (V, E, σ) is filled

Chordal graphs (Fulkerson and Gross 1965)

a graph is chordal if and only if it has a perfect elimination ordering

Simplicial elimination: to construct a perfect elimination,

- find a simplicial vertex v and take $\sigma(1) = v$
- choose for $\sigma(2), \ldots, \sigma(n)$ a perfect elimination ordering of $G(V \setminus \{v\})$

Practical algorithms

- algorithms exist that find perfect elimination ordering in $O(|V| + |E|)$ time
- best known algorithm is *Maximum Cardinality Search* (Tarjan and Yannakakis 1984)
- can be used to test chordality

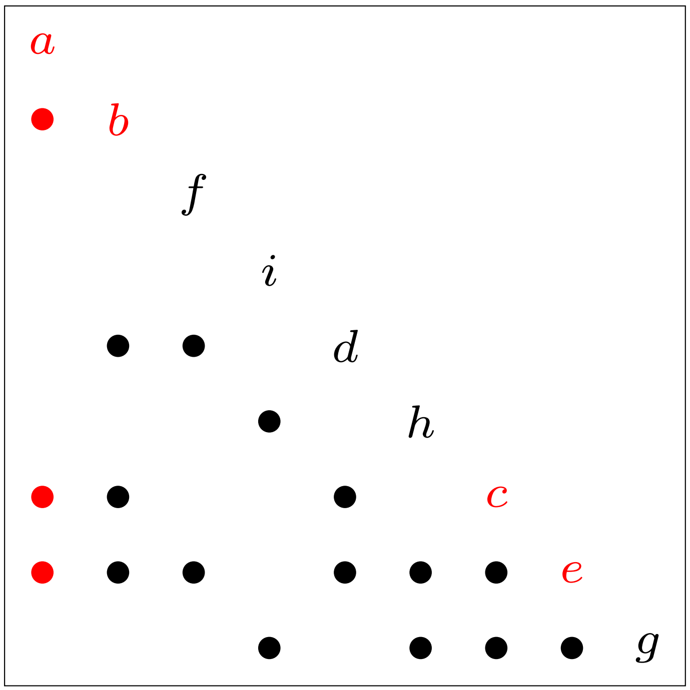
I. Chordal graphs

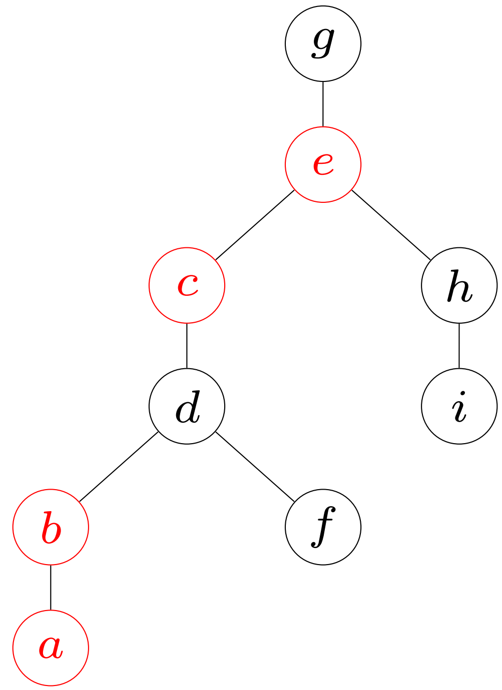
- undirected graphs
- origins
- definition
- clique trees
- perfect elimination
- **elimination trees**
- supernodes
- graph elimination

Elimination tree for filled graph

Elimination tree (etree) of filled graph $G = (V, E, \sigma)$

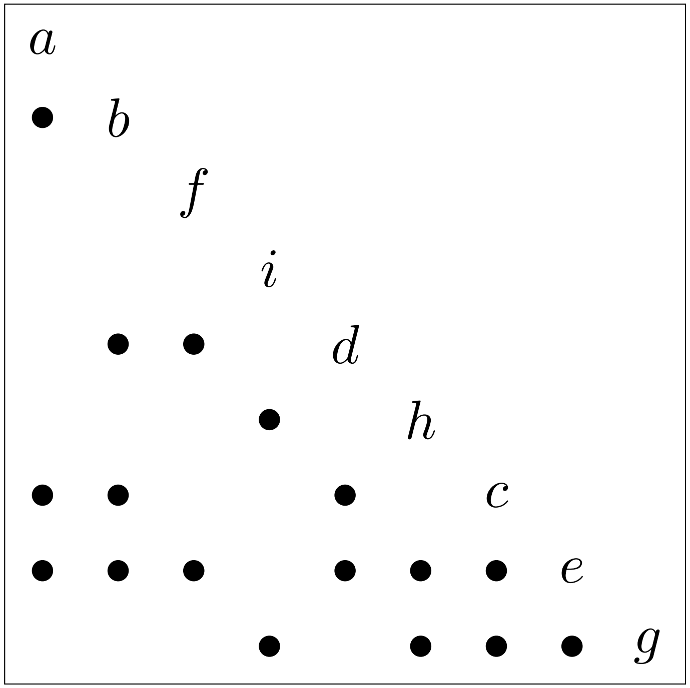
- vertices of elimination tree are V
- parent $p(v)$ of vertex v in elimination tree is first vertex in $\mathrm{adj}^+(v)$

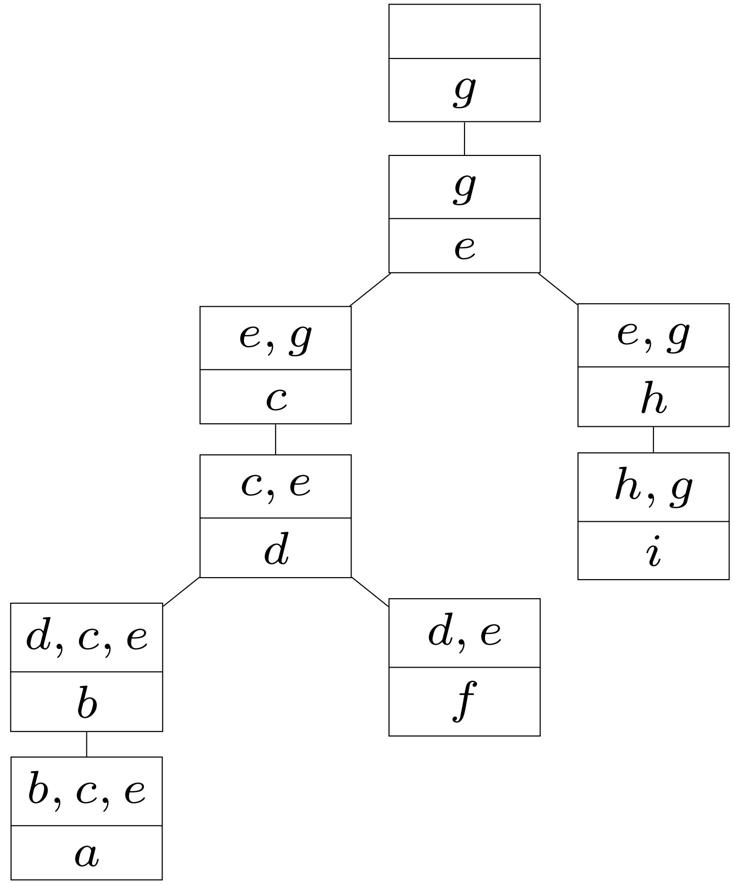




- complete pattern cannot be determined from elimination tree
- some useful information, for example, elements of $\mathrm{adj}^+(v)$ are ancestors of v

Expanded elimination tree

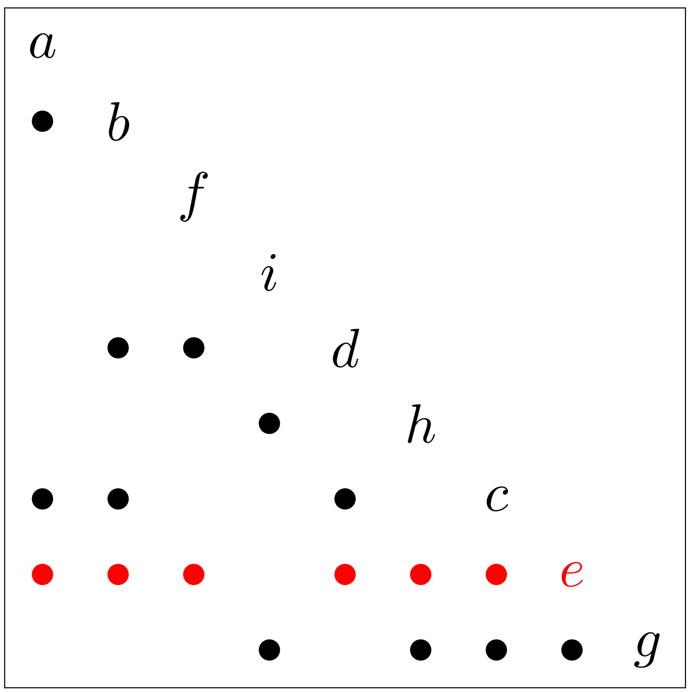


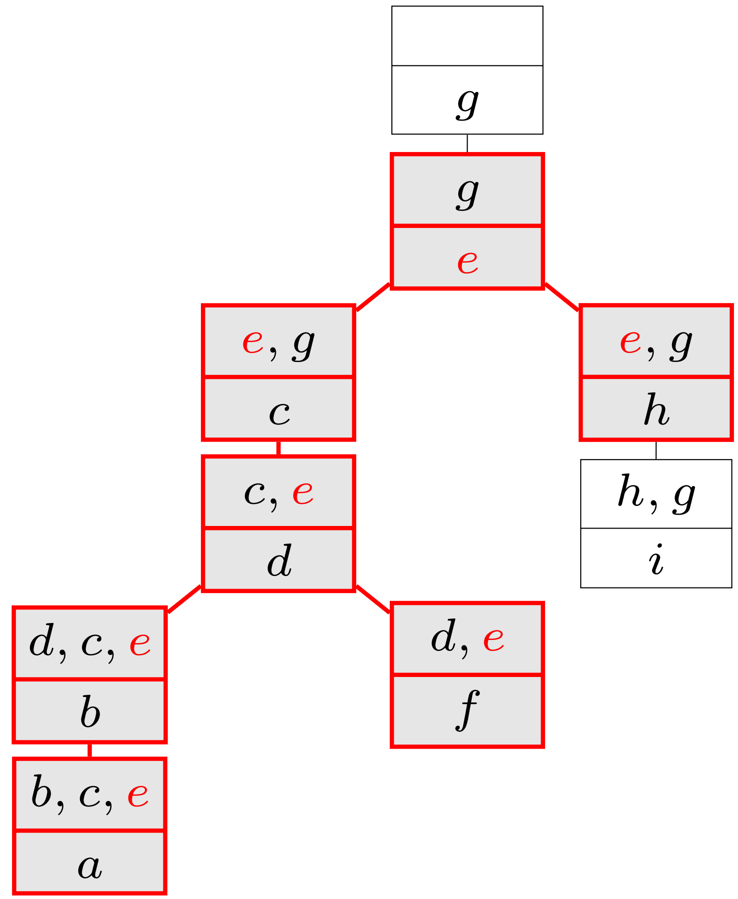


- bottom row in each block is a vertex v, top row is $\mathrm{adj}^+(v)$
- monotone transitivity means that each set $\mathrm{col}(v) = \{v\} \cup \mathrm{adj}^+(v)$ is complete
- therefore $\mathrm{adj}^+(v) \subseteq \mathrm{col}(p(v))$ for every (non-root) v

Induced subtree property

vertices in $\mathrm{row}(v) = \{w \mid v \in \mathrm{col}(w)\}$ form a subtree of elimination tree





$$\mathrm{row}(e) = \{a, b, f, d, h, c, e\}$$

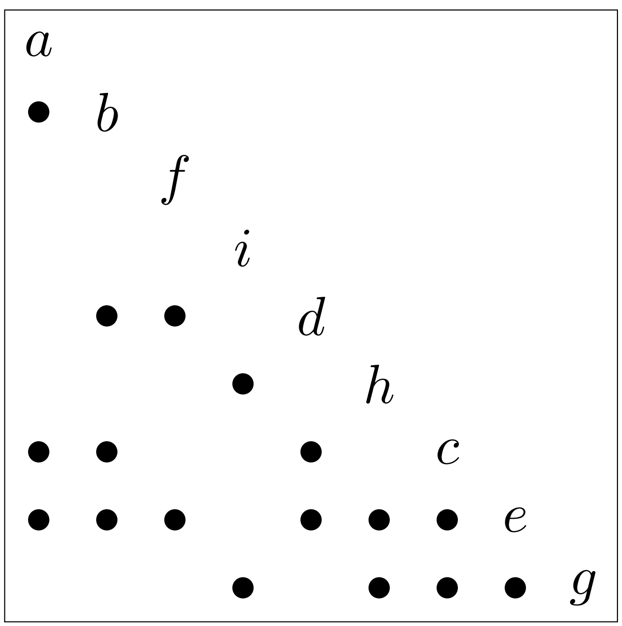
gives another representation of chordal graph as tree intersection graph

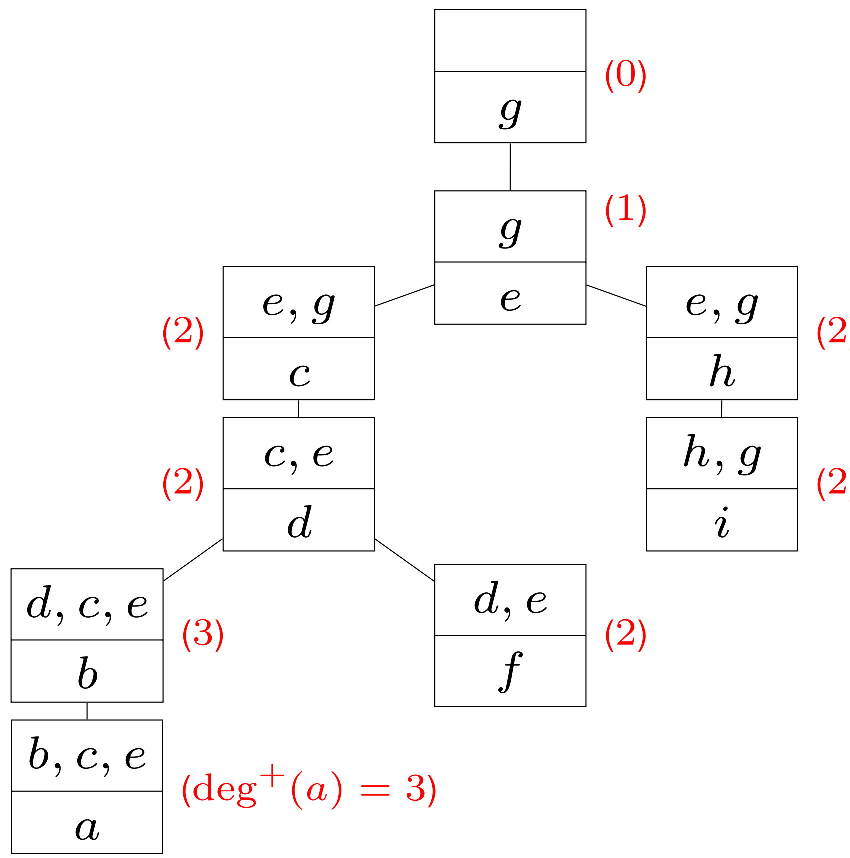
Higher degrees

since $\mathrm{adj}^+(v) \subseteq \mathrm{col}(p(v))$, the higher degrees satisfy

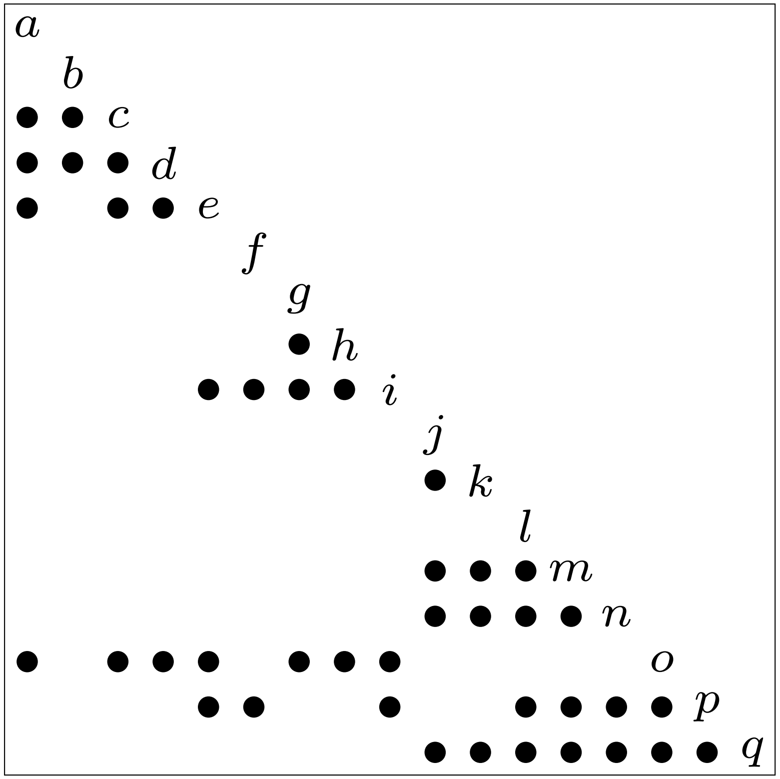
$$\deg^+(v) \leq \deg^+(p(v)) + 1$$

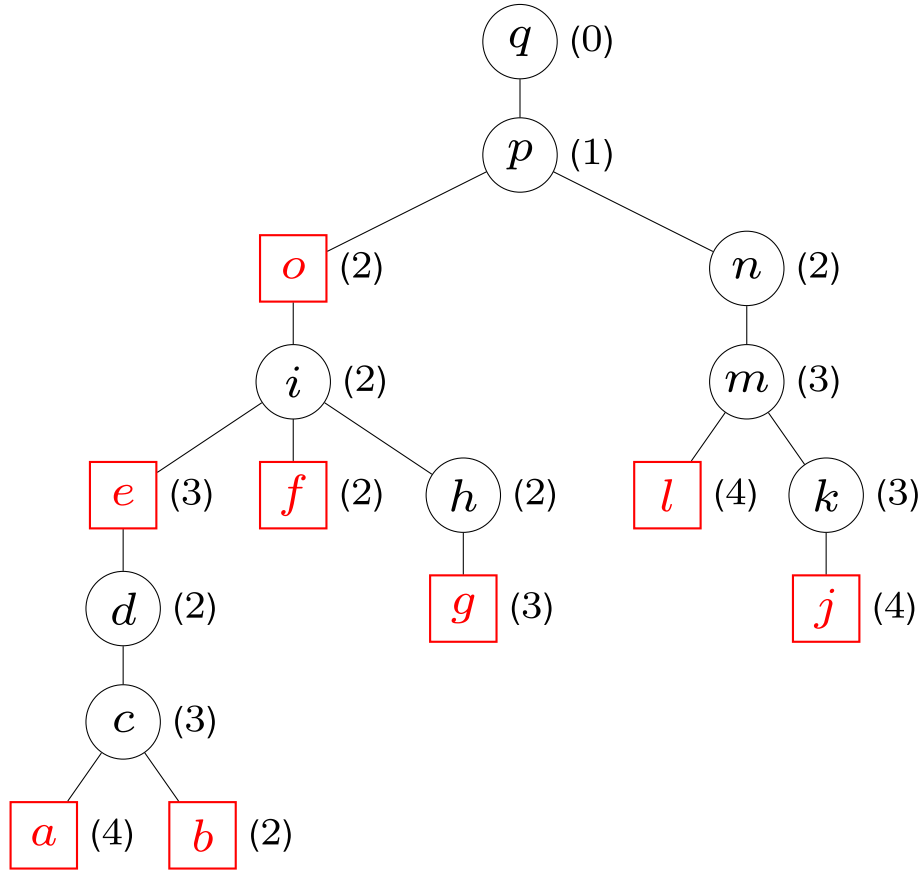
with equality if $\mathrm{adj}^+(v) = \mathrm{col}(p(v))$





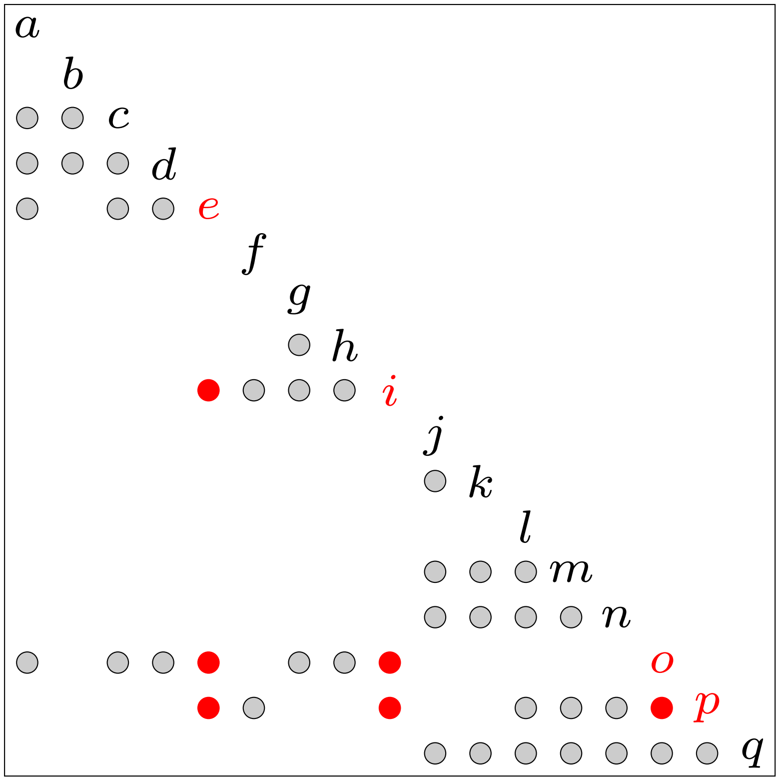
Cliques from elimination tree and higher degrees

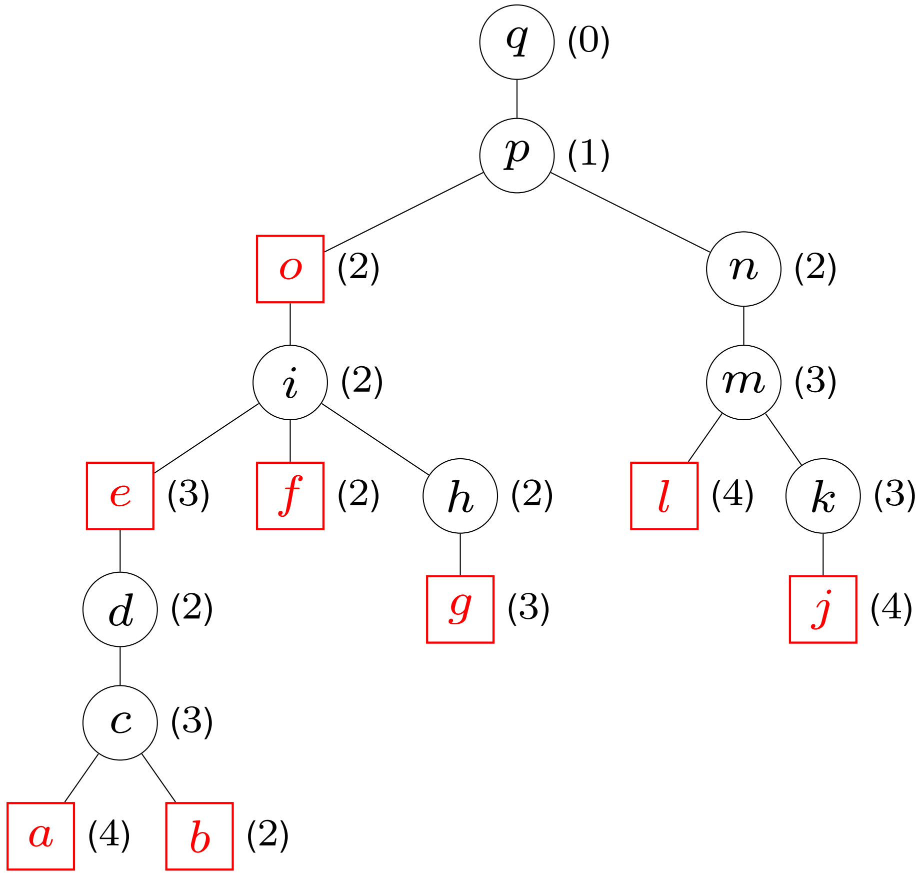




- $\mathrm{col}(v)$ is a clique if $\deg^+(w) < \deg^+(v) + 1$ for all children w of v
- if $\mathrm{col}(v)$ is a clique, we call v the **representative vertex** of the clique

Cliques from elimination tree and higher degrees





- test only needs elimination tree and higher degrees, not the entire graph
- implies that a chordal graph has at most n cliques

I. Chordal graphs

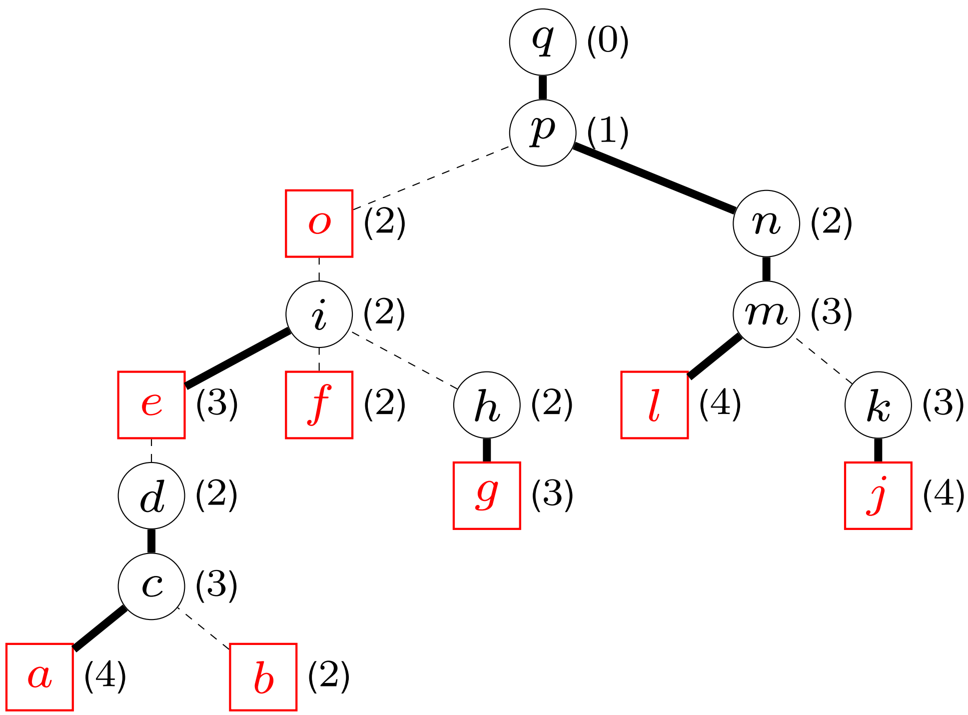
- undirected graphs
- origins
- definition
- clique trees
- perfect elimination
- elimination trees
- **supernodes**
- graph elimination

Maximal supernode partition

partition V in **maximal supernodes:** sets of the form

$$\operatorname{snd}(v) = \{v, p(v), p^2(v), \ldots, p^{n_v}(v)\}$$

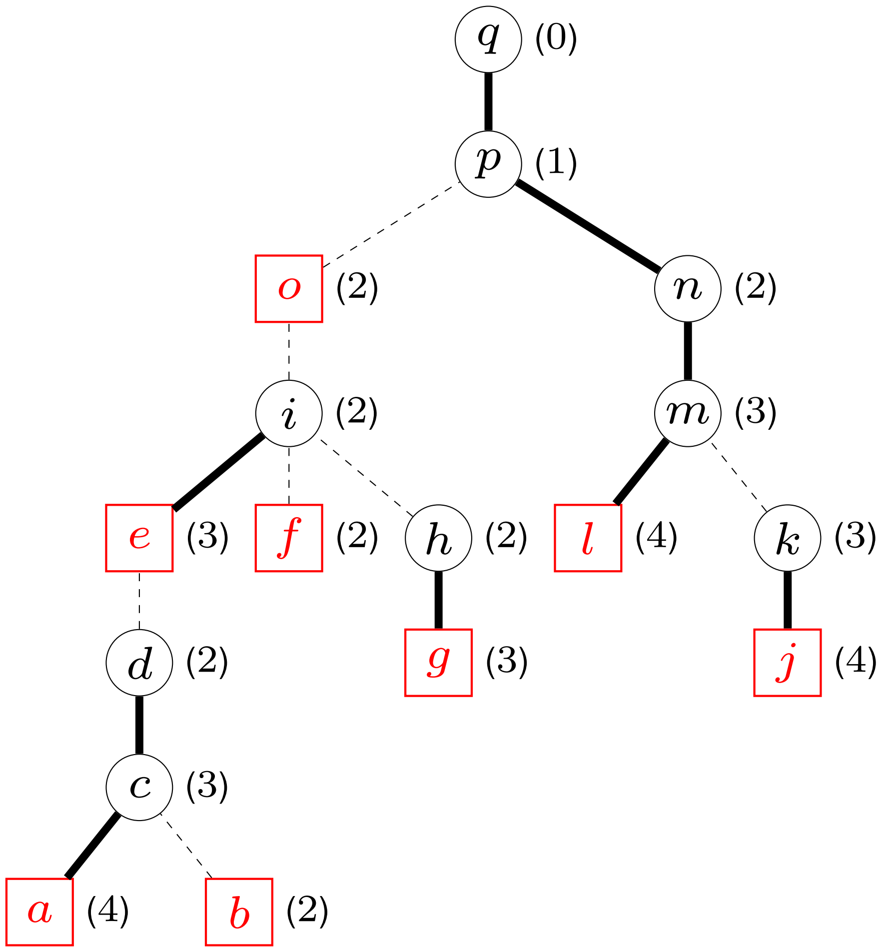
- first vertex v is a clique representative vertex ($\operatorname{col}(v)$ is a clique)
- $\deg^+(p^k(v)) = \deg^+(p^{k-1}(v)) - 1$ for $k = 1, \ldots, n_v$

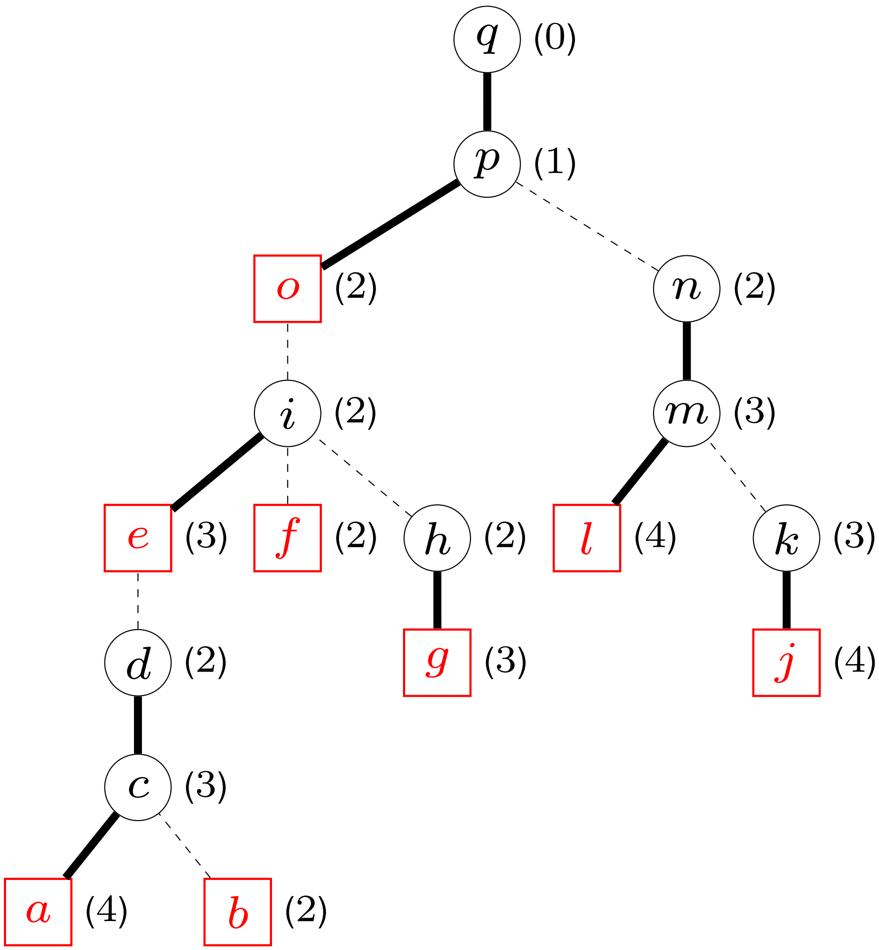


$$\begin{aligned}
\operatorname{snd}(a) &= \{a, c, d\} \\
\operatorname{snd}(b) &= \{b\} \\
\operatorname{snd}(e) &= \{e, i\} \\
\operatorname{snd}(f) &= \{f\} \\
\operatorname{snd}(g) &= \{g, h\} \\
\operatorname{snd}(j) &= \{j, k\} \\
\operatorname{snd}(l) &= \{l, m, n, p, q\} \\
\operatorname{snd}(o) &= \{o\}
\end{aligned}$$

(Lewis, Peyton, Pothen 1998, Pothen and Sun 1990)

Nonuniqueness of maximal supernode partition



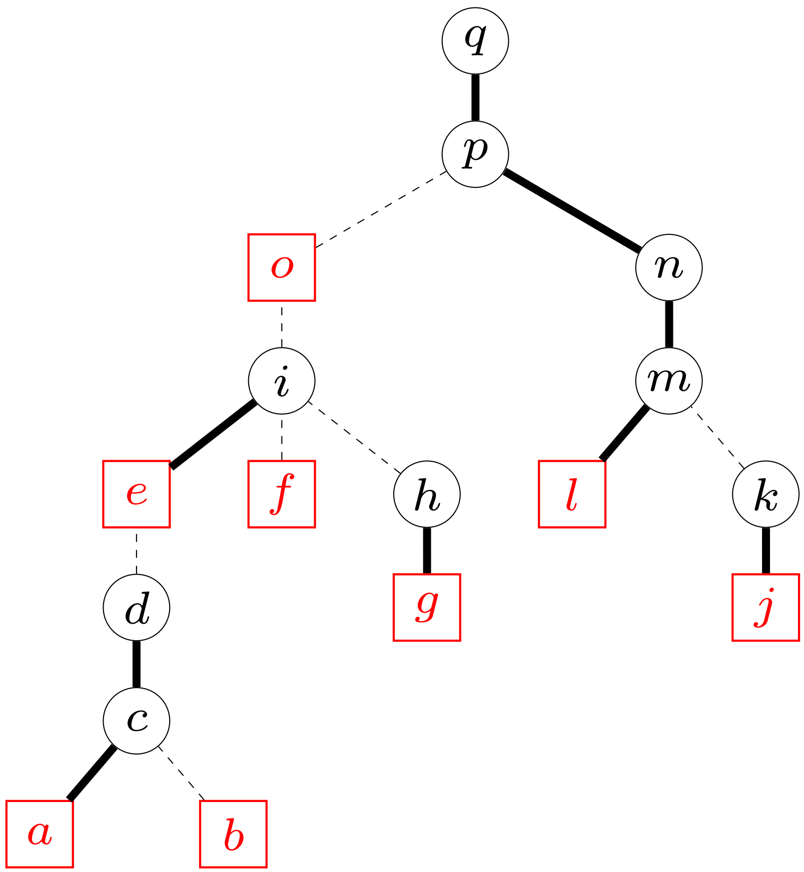


$$\begin{aligned} \mathrm{snd}(o) &= \{o\} \\ \mathrm{snd}(l) &= \{l, m, n, p, q\} \end{aligned}$$

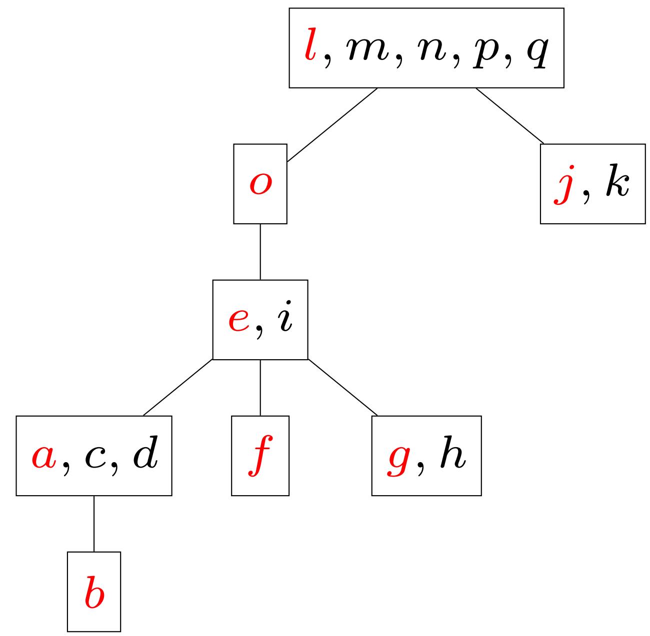
$$\begin{aligned} \mathrm{snd}(o) &= \{o, p, q\} \\ \mathrm{snd}(l) &= \{l, m, n\} \end{aligned}$$

Supernodal elimination tree

elimination tree



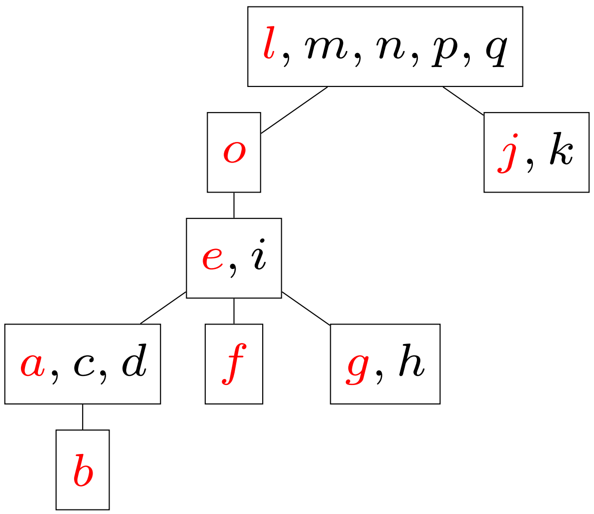
supernodal elimination tree

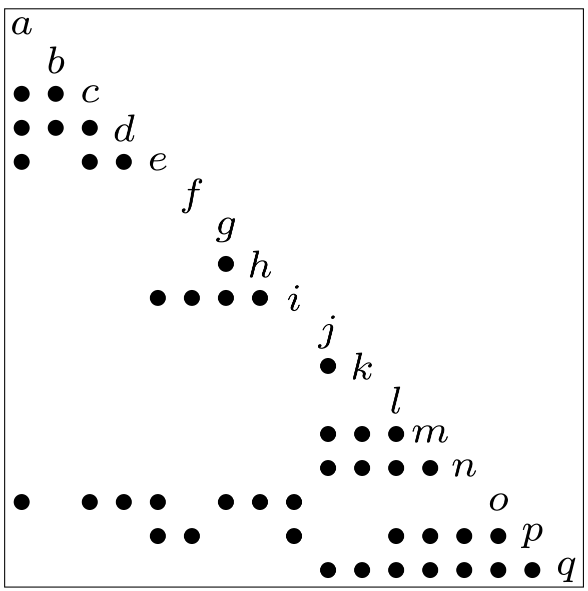


- vertices are maximal supernodes
- parent of $\mathrm{snd}(v)$: supernode that contains parent (in etree) of last element of $\mathrm{snd}(v)$

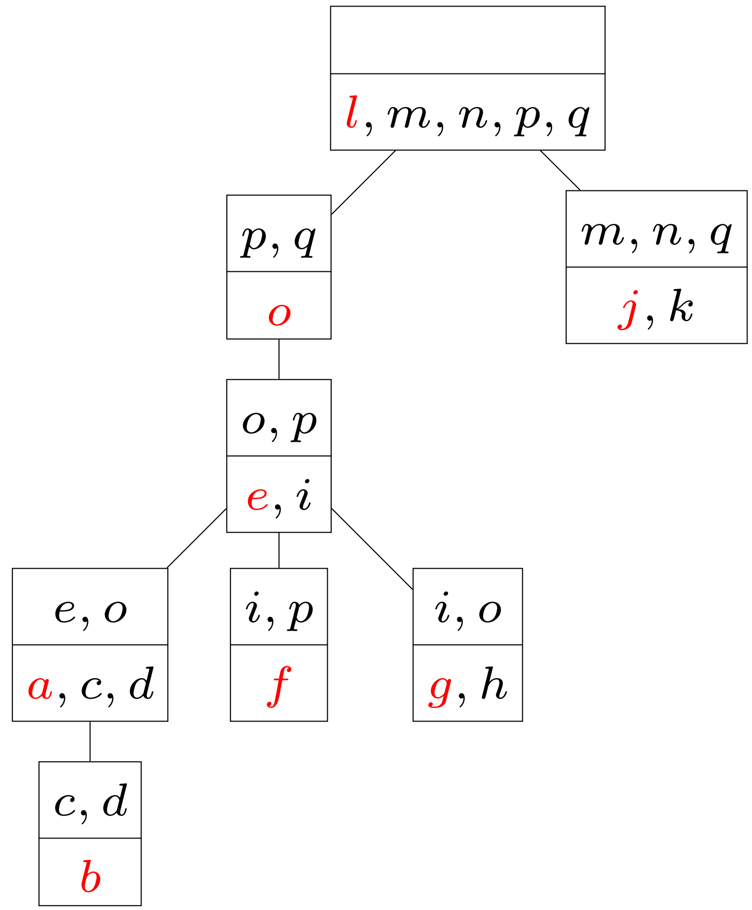
Clique tree and maximal supernode partition

supernodal elimination tree



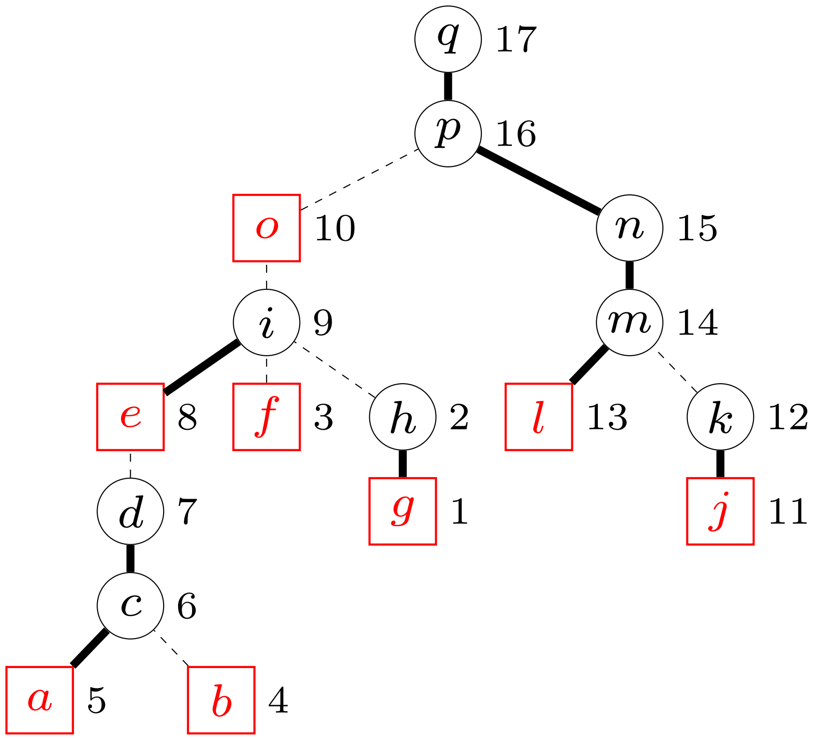


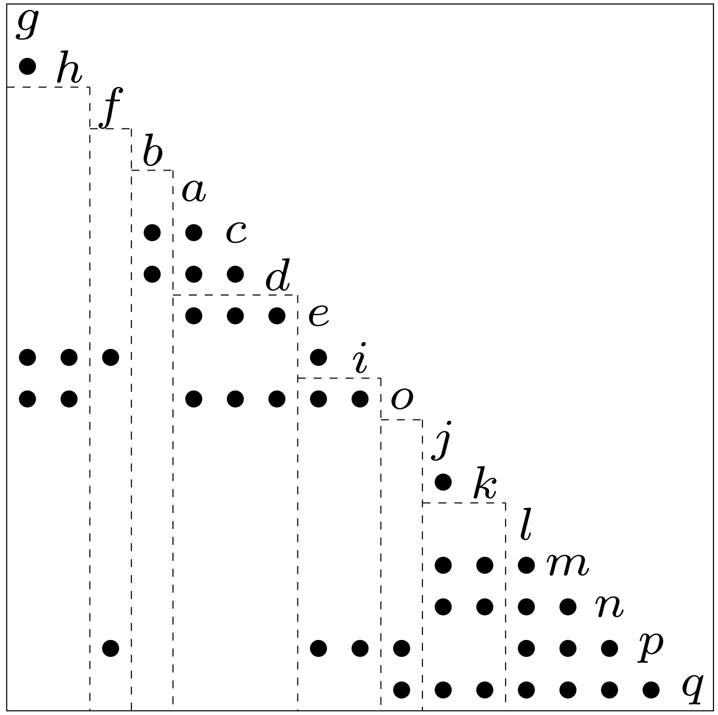
clique tree



- $\mathrm{snd}(v)$ is residual of clique $\mathrm{col}(v)$
- clique separator is $\mathrm{col}(v) \setminus \mathrm{snd}(v)$

Postordering



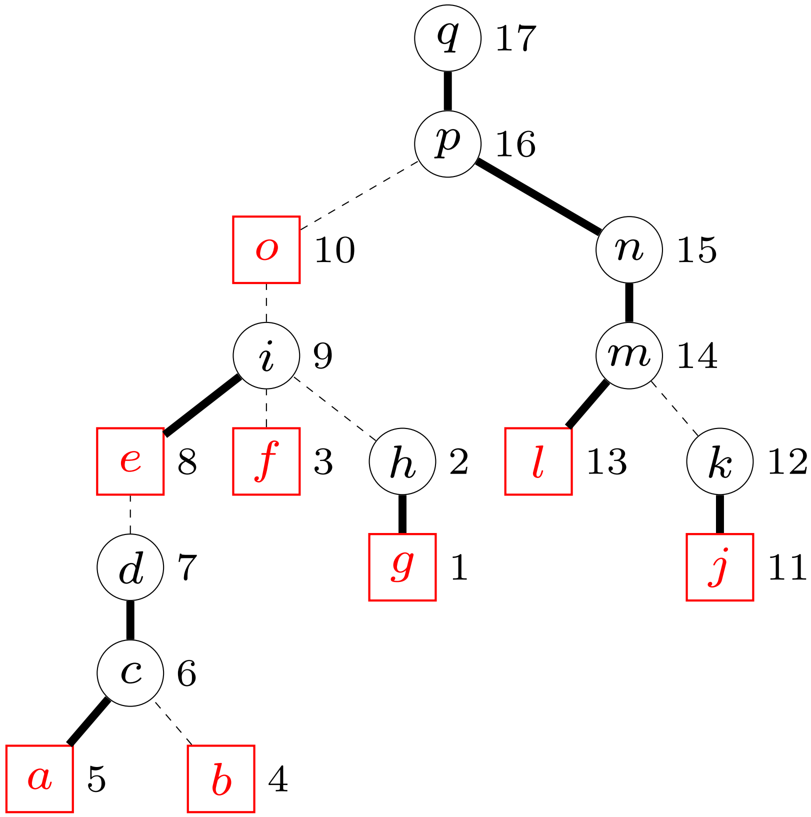


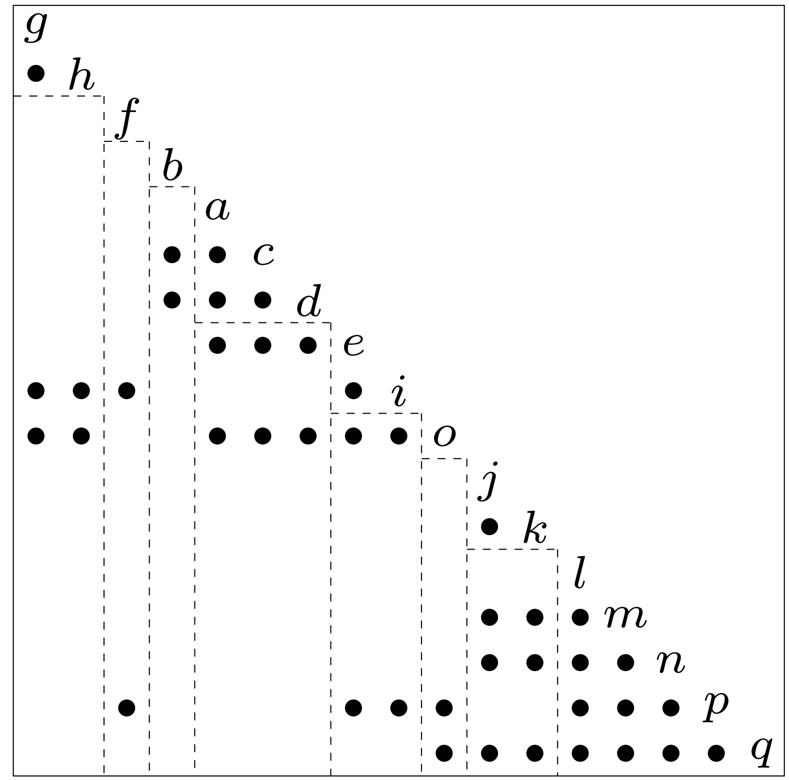
based on a supernode partition we can define a new perfect elimination ordering

- elements of each supernode $\mathrm{snd}(v)$ are numbered consecutively, starting at v
- if $\mathrm{snd}(w)$ is the parent of $\mathrm{snd}(v)$ in supernodal elim. tree, then $w \succ v$
- hence, vertices in $\mathrm{col}(v) \setminus \mathrm{snd}(v)$ follow those in $\mathrm{snd}(v)$

this can be achieved by a postordering of the elimination tree (without changing it)

Example





	$\mathrm{col}(a)$	$\mathrm{snd}(a)$	$\mathrm{col}(a) \setminus \mathrm{snd}(a)$
vertices v	a, c, d, e, o	a, c, d	e, o
numbers $\sigma^{-1}(v)$	$5, 6, 7, 8, 9$	$5, 6, 7$	$8, 10$

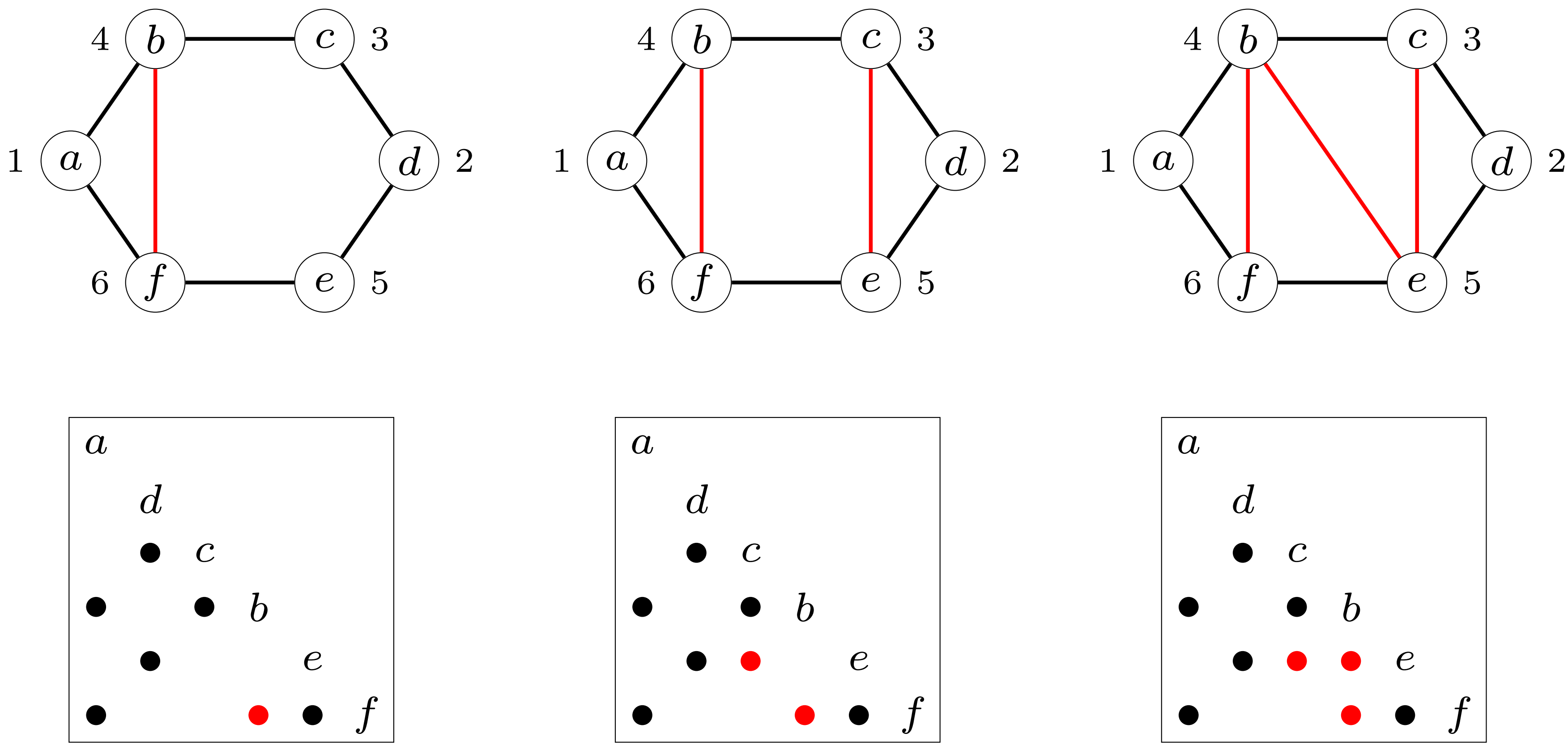
I. Chordal graphs

- undirected graphs
- origins
- definition
- clique trees
- perfect elimination
- elimination trees
- supernodes
- **graph elimination**

Elimination graph

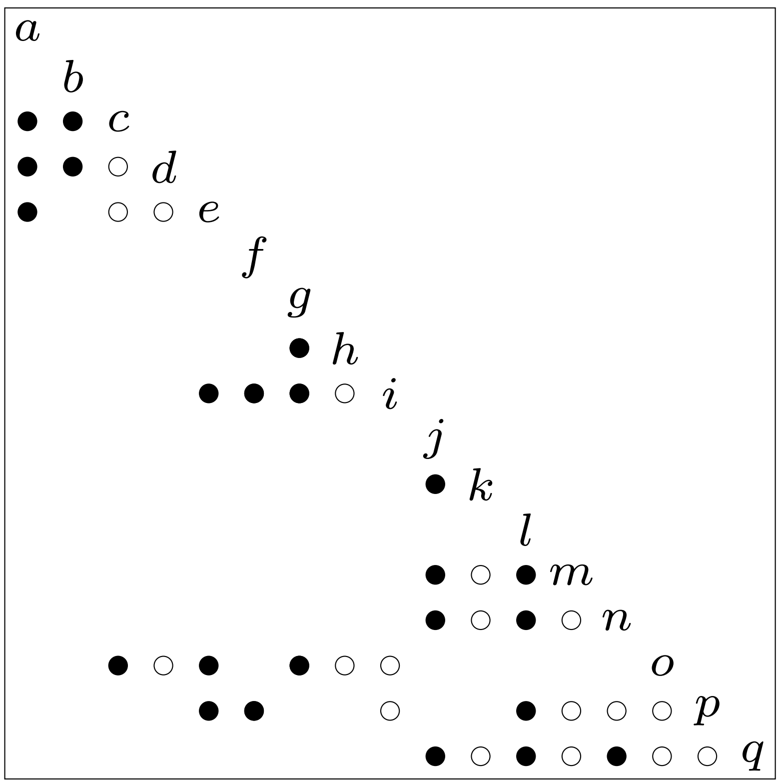
a filled graph $G^*_\sigma = (V, E^*_\sigma, \sigma)$ constructed from $G_\sigma = (V, E, \sigma)$ as follows:

- start with $E^*_\sigma = E$, enumerate vertices $v = \sigma(i)$ for $i = 1, 2, \ldots, |V|$
- in step i, add edges to make higher neighborhood $\operatorname{adj}^+(v)$ complete



Chordal extension

- the graph (V, E_σ^*) is chordal by construction, with perfect elimination ordering σ
- (V, E_σ^*) is called a **chordal extension** or **triangulation** of (V, E)
- the added edges $E_\sigma^* \setminus E$ during graph elimination are called **fill-in** or **fill**



●: edges of non-chordal graph

○: filled entries

Cholesky factorization of positive definite matrix

$$A = LDL^T \qquad L \text{ unit lower-triangular, } D \text{ positive diagonal}$$

Recursive ('outer product') algorithm

- write A as

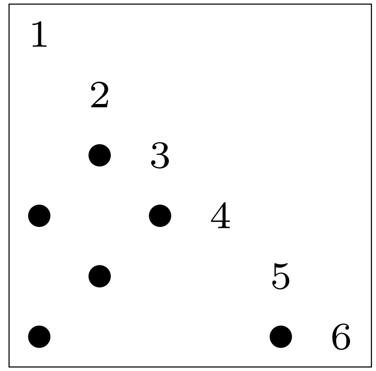
$$\begin{aligned} A &= \begin{bmatrix} d_1 & b^T \\ b & C \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 \\ (1/d_1)b & I \end{bmatrix} \begin{bmatrix} d_1 & 0 \\ 0 & C - (1/d_1)bb^T \end{bmatrix} \begin{bmatrix} 1 & (1/d_1)b^T \\ 0 & I \end{bmatrix} \\ &= L_1 D_1 L_1^T \end{aligned}$$

- Cholesky factorization of $C - (1/d_1)bb^T = \tilde{L}_2 D_2 \tilde{L}_2^T$:

$$\begin{aligned} A &= L_1 \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L}_2 \end{bmatrix} \begin{bmatrix} d_1 & 0 \\ 0 & D_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \tilde{L}_2^T \end{bmatrix} L_1^T \\ &= LDL^T \end{aligned}$$

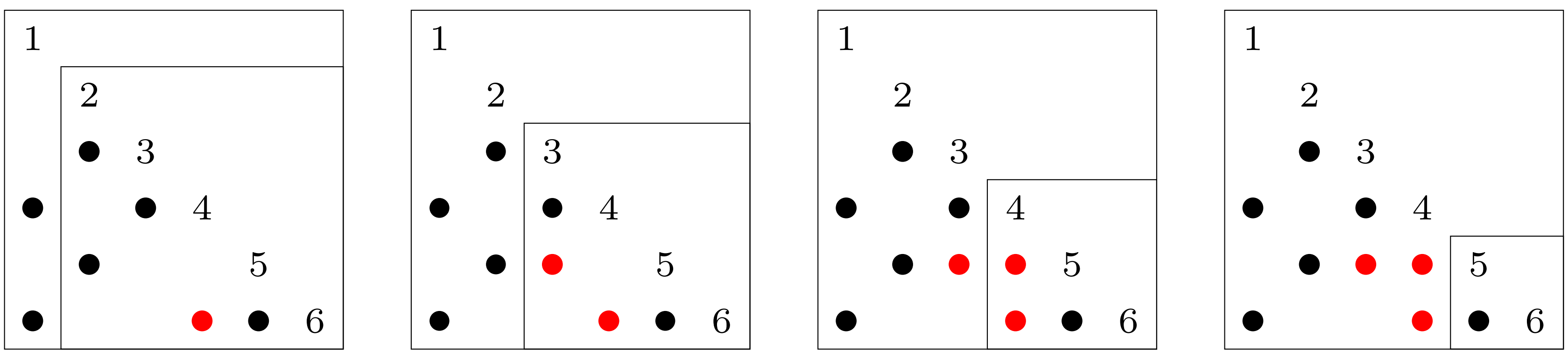
Sparsity pattern during factorization

suppose A has sparsity pattern E, and define $\sigma = (1, 2, \ldots, n)$



- sparsity pattern after each step of the recursion

$L_1 + D_1 + L_1^T$



- final sparsity pattern is E_σ^*

Choosing an elimination ordering

Minimum ordering

- minimizes the number of edges in the elimination graph
- finding a minimum ordering is is NP-complete (Yannakakis 1981)

Minimize clique number

- minimize the size of the largest clique in the elimination graph
- smallest clique number over all possible orderings is called the *treewidth*
- finding this ordering is also NP-complete

Minimal ordering: there exists no ordering σ' with $E^*_{\sigma'} \subset E^*_\sigma$

- if the graph is chordal, any minimal ordering is a perfect elimination ordering
- several algorithms for finding a minimal ordering with complexity $O(|V| \cdot |E|)$

Non-minimal heuristics: faster than minimal ordering; may give smaller fill-in

Analysis of elimination graph

algorithms exist for analyzing chordal extension (V, E^*_σ) before constructing it:

- constructing elimination tree
- calculating monotone (higher and lower) degrees
- calculating number of filled edges
- finding clique representatives
- finding supernodes, supernodal elimination tree

complexity is linear or nearly linear in $|V| + |E|$ (the size of original graph)

(Liu 1990, Gilbert, Ng, Peyton 1994, Davis 2006)

Applications of graph elimination

Elimination algorithms: common in many applications, for example

- linear equations: Gauss elimination
- linear inequalities: Fourier-Motzkin elimination
- optimization: dynamic programming
- probability: computing marginal distributions

Graph elimination

- describes complexity of many types of elimination algorithms
- we discuss two examples with discrete variables

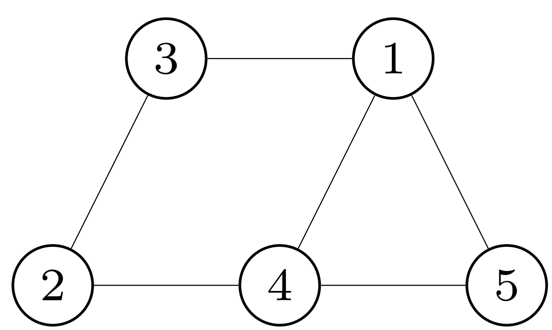
Interaction graph

- n discrete variables $x_1, \ldots, x_n$;
- x_i takes values in finite set X_i of size $s_i = |X_i|$
- l index sets (ordered subsets of $\{1, 2, \ldots, n\}$) $\beta_1, \ldots, \beta_l$
- l functions (tables) $f_k(x_{\beta_k})$, *i.e.*, f_k depends only on variables x_i with $i \in \beta_k$
- the **interaction graph** (co-occurrence graph) is defined as

$$V = \{1, \ldots, n\}, \qquad \{i, j\} \in E \iff i, j \in \beta_k \text{ for some } k$$

Example: five variables, four functions

$$f_1(x_1, x_4, x_5), \qquad f_2(x_1, x_3), \qquad f_3(x_2, x_3), \qquad f_4(x_2, x_4)$$



Discrete dynamic programming

$$\begin{array}{ll} \text{minimize} & f(x) = \sum_{k=1}^{l} f_k(x_{\beta_k}) \\ \text{subject to} & x \in X = X_1 \times \cdots \times X_n \end{array}$$

- brute-force enumeration requires enumerating $\prod_i s_i$ values of x
- solution by elimination computes minimum as

$$\min f(x) = \min_{x_{\sigma(n)}} \cdots \min_{x_{\sigma(2)}} \min_{x_{\sigma(1)}} f(x_1, \ldots, x_n)$$

  complexity depends on interaction graph and elimination order

- we explain this for the example

$$f(x) = f_1(x_1, x_4, x_5) + f_2(x_1, x_3) + f_3(x_2, x_3) + f_4(x_2, x_4)$$

  for simplicity we assume $s_1 = \cdots = s_5 = s$

Example

consider the elimination order $\sigma = (1, 2, 3, 4, 5)$

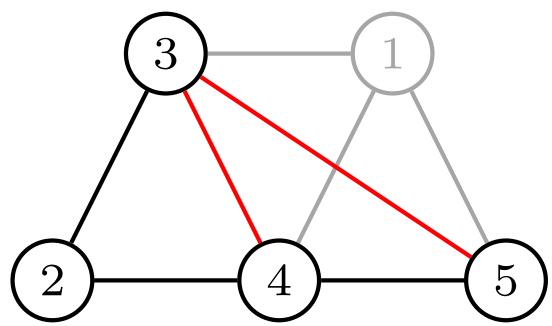
Minimize over x_1

$$\min_{x_1} f(x) = \min_{x_1} \left(f_1(x_1, x_4, x_5) + f_2(x_1, x_3) + f_3(x_2, x_3) + f_4(x_2, x_4) \right)$$

- requires enumerating s^4 possible values of (x_1, x_3, x_4, x_5) to find

$$u_1(x_3, x_4, x_5) = \min_{x_1} \left(f(x_1, x_4, x_5) + f_2(x_1, x_3) \right)$$

- interaction graph of $u_1(x_3, x_4, x_5) + f_3(x_2, x_3) + f_4(x_2, x_4)$ is



Example

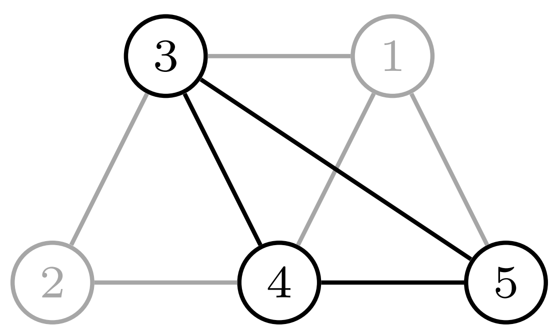
Minimize over x_2

$$\min_{x_2, x_1} f(x) = \min_{x_2} \left(u_1(x_3, x_4, x_5) + f_3(x_2, x_3) + f_4(x_2, x_4) \right)$$

- requires enumerating s^3 possible values of (x_2, x_3, x_4) to find

$$u_2(x_3, x_4) = \min_{x_2} \left(f_3(x_2, x_3) + f_4(x_2, x_4) \right)$$

- interaction graph of $u_1(x_3, x_4, x_5) + u_2(x_3, x_4)$ is



Example

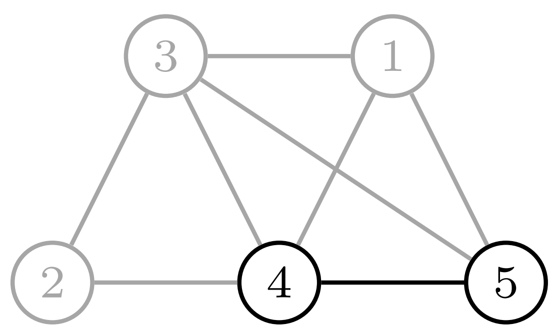
Minimize over x_3

$$\min_{x_3,x_2,x_1} f(x) = \min_{x_3} \left(u_1(x_3, x_4, x_5) + u_2(x_3, x_4)\right)$$

- requires enumerating s^3 possible values of (x_3, x_4, x_5) to find

$$u_3(x_4, x_5) = \min_{x_3} \left(u_1(x_3, x_4, x_5) + u_2(x_3, x_4)\right)$$

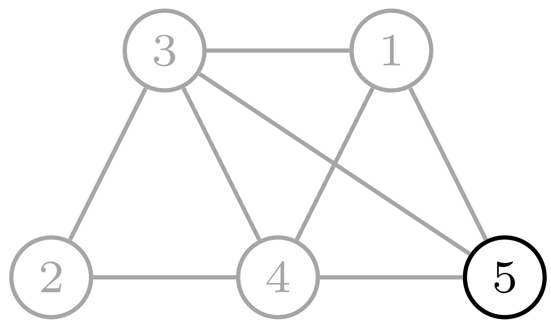
- interaction graph of $u_3(x_4, x_5)$ is



Example

Minimize over x_4: enumerate s^2 values to get

$$\min_{x_4,x_3,x_2,x_1} f(x) = \min_{x_4} u_3(x_4, x_5) = u_4(x_5)$$



Minimize over x_5: enumerate s values to get final answer

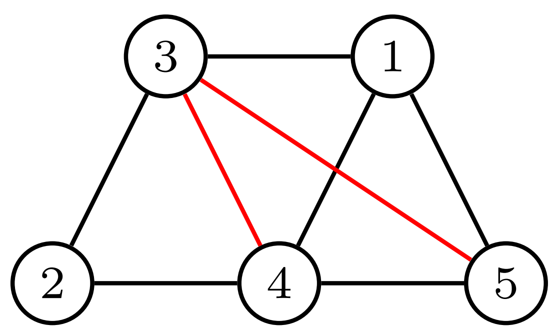
$$\min_{x} f(x) = \min_{x_5} u_4(x_5)$$

Example

- the algorithm can be summarized as a nested minimization formula

$$\min_x f(x) = \min_{x_5} \min_{x_4} \min_{x_3} \left(\min_{x_1} \left(f_1(x_1, x_4, x_5) + f_2(x_1, x_3) \right) + \min_{x_2} \left(f_3(x_2, x_5) + f_4(x_2, x_4) \right) \right)$$

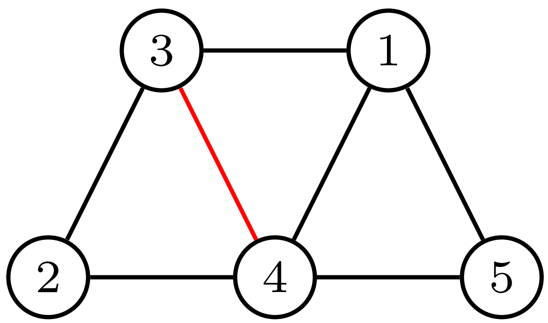
- cost is s^4 because largest clique in elimination graph has size 4



Example

consider the elimination order $\sigma = (5, 1, 2, 3, 4)$

$$
\begin{aligned}
\min_x f(x) &= \min_x \left(f(x_1, x_4, x_5) + f_2(x_1, x_3) + f_3(x_2, x_3) + f_4(x_2, x_4)\right) \\
&= \min_{x_4} \min_{x_3} \left(\min_{x_1} \left(\min_{x_5} f_1(x_1, x_4, x_5) + f_2(x_1, x_3) \right) \right. \\
&\qquad \left. + \min_{x_2} \left(f_3(x_2, x_3) + f_4(x_2, x_4)\right) \right)
\end{aligned}
$$



complexity is s^3

Probabilistic networks

the 'min-sum' algorithm for $\min \sum_{k=1}^{l} f_k(x_{\beta_k})$ is easily adapted to a 'sum-product'

$$\sum_{x \in X} \prod_{k=1}^{l} f_k(x_{\beta_k})$$

- used for inferencing in probabilistic networks
- $\prod_k f_k(x_{\beta_k})$ is a discrete probability distribution
- interaction graph shows conditional independence relations
- complexity is exponential in the size of the largest clique
- ordering heuristics that yield small cliques are important

References

- M. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, 2nd edition, 2004.
- J. Blair and B, Peyton, *An introduction to chordal graphs and clique trees*, in: *Graph Theory and Sparse Matrix Computation*, 1993.
- T. Davis, *Direct Methods for Sparse Linear Systems*, 2006.

II. Sparse matrices

- symmetric sparse matrices
- positive semidefinite matrices
  - Cholesky factorization
  - clique decomposition
  - multifrontal factorization
  - projected inverse
  - logarithmic barrier
- positive semidefinite completion
  - clique decomposition
  - minimum rank positive semidefinite completion
  - maximum determinant completion
  - logarithmic barrier
- Euclidean distance matrix completion
  - Euclidean distance matrices
  - clique decomposition
  - minimum dimension completion

Symmetric sparsity pattern

- sparsity pattern E (of order n) is a set

$$E \subseteq \{\{i,j\} \mid i,j \in \{1,2,\ldots,n\}\}$$

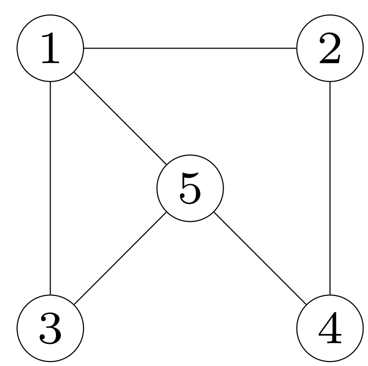
- symmetric matrix A of order n has sparsity pattern E if

$$i \neq j, \quad \{i,j\} \notin E \qquad \Longrightarrow \qquad A_{ij} = A_{ji} = 0$$

  notation: $A \in \mathbf{S}^n_E$

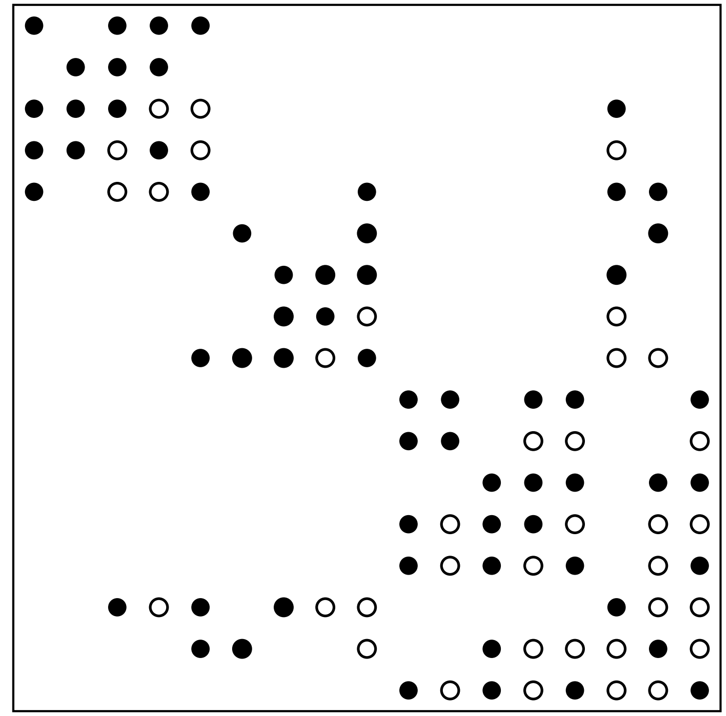
- the graph $G = (V, E)$ with $V = \{1, 2, \ldots, n\}$ is called the sparsity graph

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} & 0 & A_{15} \\ A_{21} & A_{22} & 0 & A_{24} & 0 \\ A_{31} & 0 & A_{33} & 0 & A_{35} \\ 0 & A_{42} & 0 & A_{44} & A_{45} \\ A_{51} & 0 & A_{53} & A_{54} & A_{55} \end{bmatrix}$$



Chordal sparsity patterns

Sparsity pattern of a Cholesky factor

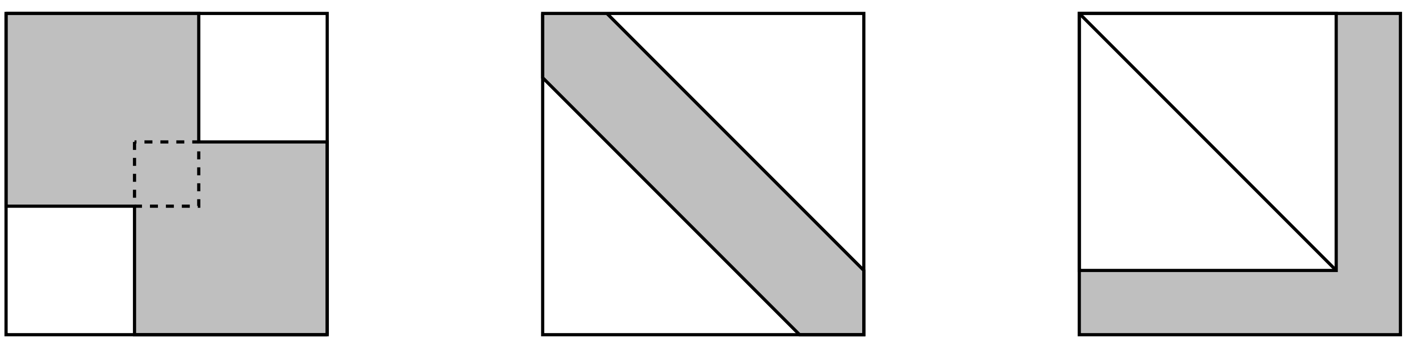


$\bullet$: nonzeros in positive definite matrix A

$\circ$: nonzeros in $L + L^T$, where $A = LDL^T$

this is a chordal extension of the pattern of A

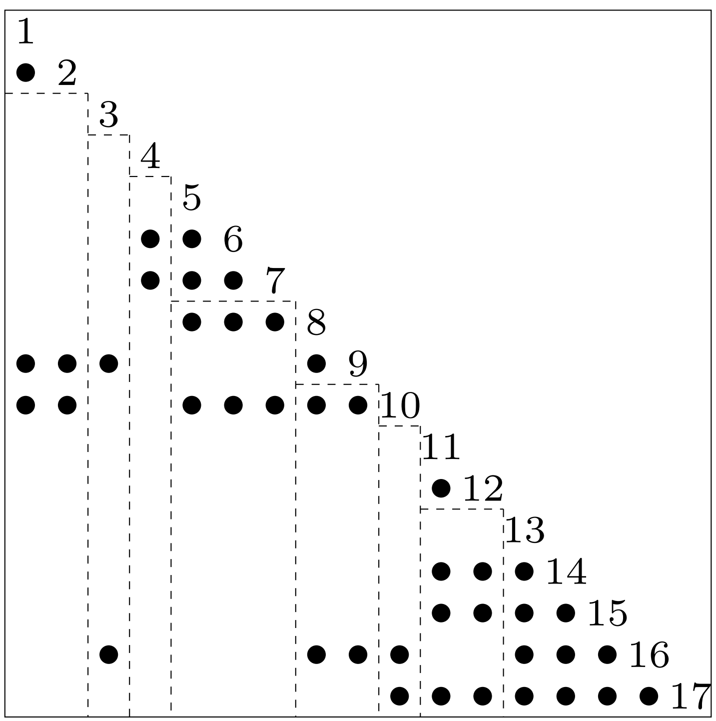
Simple examples

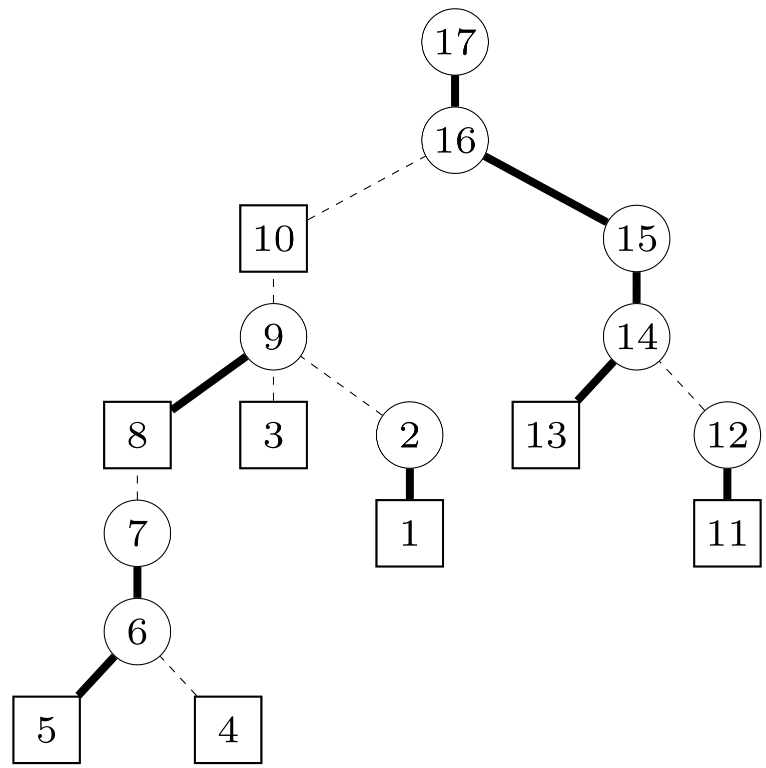


Ordering

when discussing chordal patterns, we make the assumptions of page 47

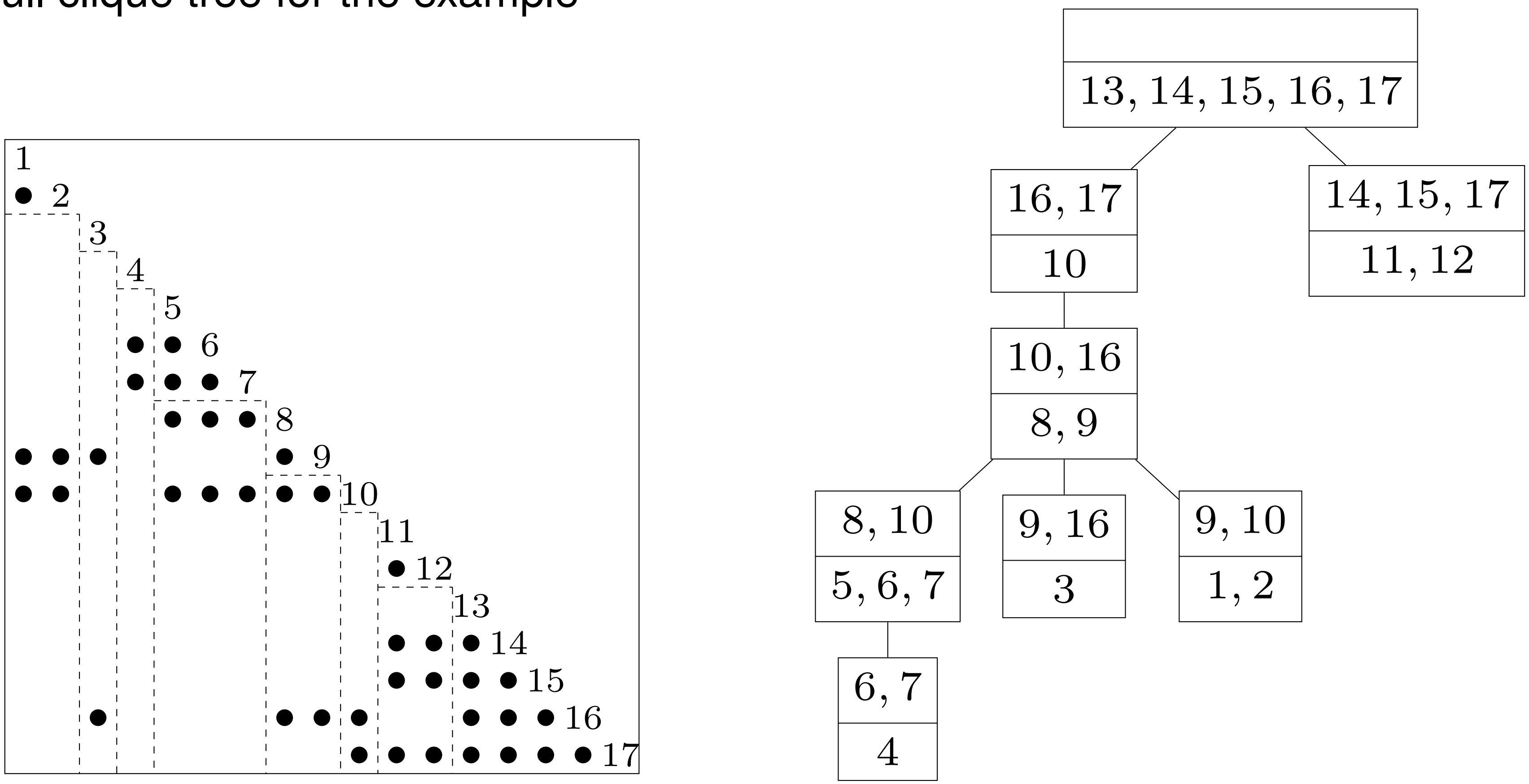
- $\sigma = (1, 2, \ldots, n)$ is a perfect elimination ordering
- indices in maximal supernodes (clique residuals) are numbered consecutively
- if $\mathrm{snd}(i)$ is the parent of $\mathrm{snd}(j)$ in the supernodal elimination tree, then $i > j$
- hence, indices in clique separator $\mathrm{col}(i) \setminus \mathrm{snd}(i)$ follow those in $\mathrm{snd}(i)$





Example

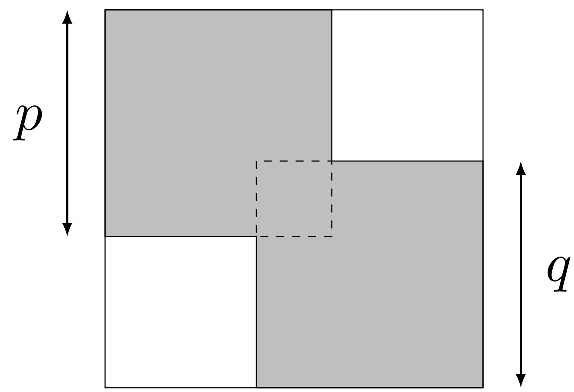
the full clique tree for the example



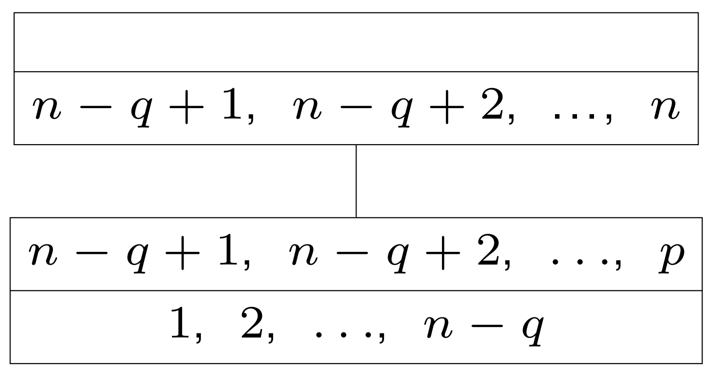
- maximal supernodes (bottom rows) numbered consecutively
- clique representatives (first element of each block) numbered before parent
- clique residuals (top rows): numbers follow indices in bottom row

Overlapping diagonal blocks

- the simplest non-complete chordal pattern has two overlapping diagonal blocks



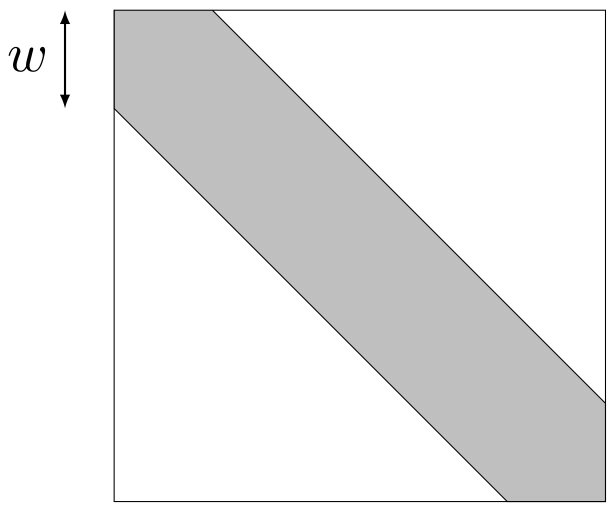
- the clique tree

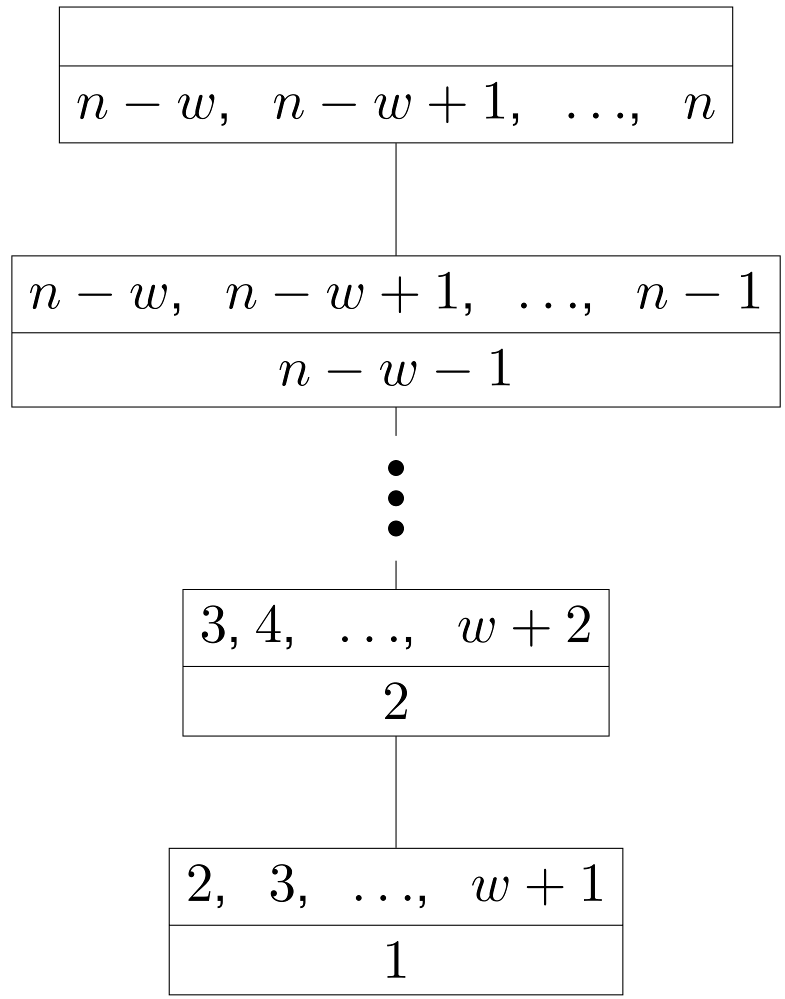


- results for this pattern can often be generalized using properties of clique trees

Band pattern

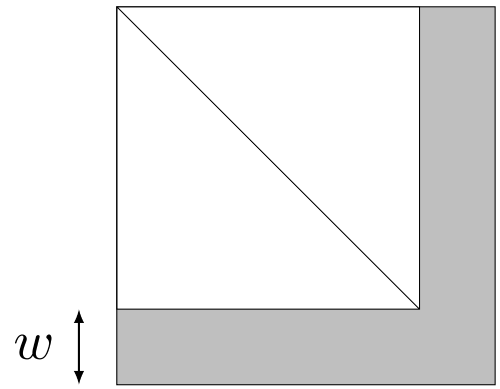
band pattern with bandwidth $2w + 1$ and clique tree

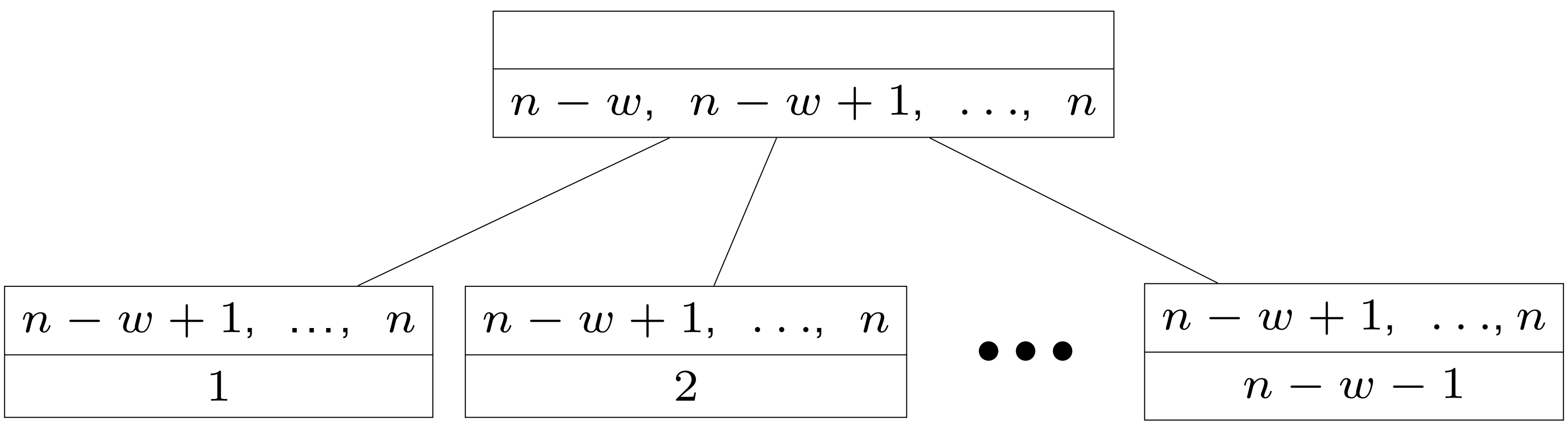




Block arrow pattern

block arrow pattern with block width w and clique tree





Indexing subvectors and submatrices

Index set: an ordered list of distinct elements of $V = \{1, 2, \ldots, n\}$

Selection matrix:

if $\beta = (\beta_1, \ldots, \beta_r)$ is an index set, then P_β stands for the $r \times n$ matrix

$$(P_\beta)_{ij} = 1 \quad \text{if } j = \beta_i, \qquad (P_\beta)_{ij} = 0 \quad \text{otherwise}$$

- this is a permutation matrix if $r = n$
- used to select subvectors or principal submatrices:

$$P_\beta x = x_\beta, \qquad P_\beta X P_\beta^T = X_{\beta\beta}$$

- adjoint defines subvector or submatrix in otherwise zero vector or matrix

$$(P_\beta^T y)_i = \begin{cases} y_j & j = \beta_i \\ 0 & j \notin \beta, \end{cases} \qquad (P_\beta^T Y P_\beta)_{kl} = \begin{cases} Y_{ij} & i = \beta_k, j \in \beta(l) \\ 0 & (i,j) \notin \beta \times \beta \end{cases}$$

Example

$$n = 5, \qquad \beta = (2, 4, 5), \qquad P_\beta = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

- for $x \in \mathbf{R}^5$ and $X \in \mathbf{S}^5$,

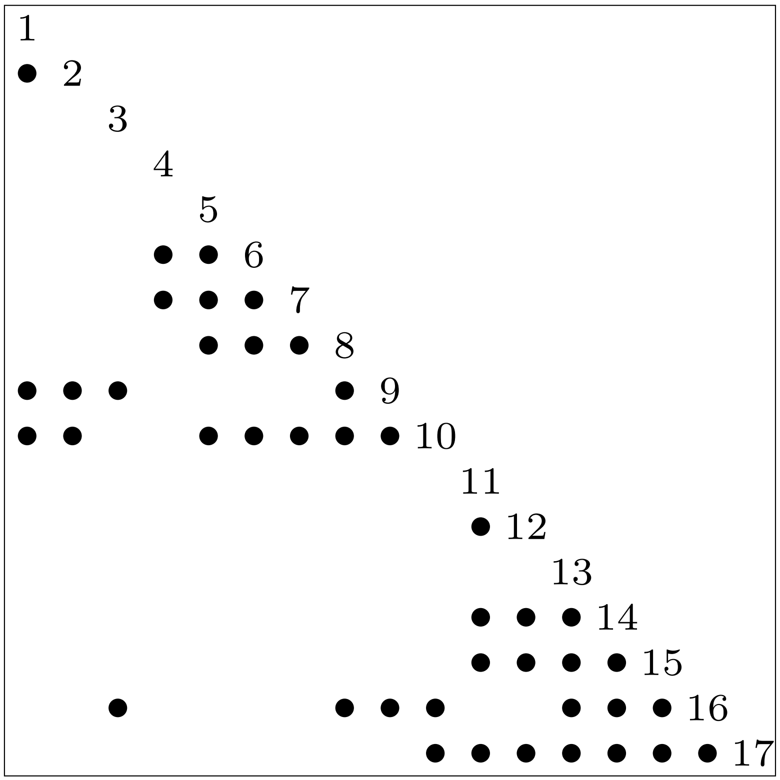
$$P_\beta x = x_\beta = \begin{bmatrix} x_2 \\ x_4 \\ x_5 \end{bmatrix}, \qquad P_\beta X P_\beta^T = X_{\beta\beta} = \begin{bmatrix} X_{22} & X_{24} & X_{25} \\ X_{42} & X_{44} & X_{45} \\ X_{52} & X_{54} & X_{55} \end{bmatrix}$$

- for $y \in \mathbf{R}^3$ and $Y \in \mathbf{S}^3$,

$$P_\beta^T y = \begin{bmatrix} 0 \\ y_1 \\ 0 \\ y_2 \\ y_3 \end{bmatrix}, \qquad P_\beta^T Y P_\beta = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{11} & 0 & Y_{12} & Y_{13} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & Y_{21} & 0 & Y_{22} & Y_{23} \\ 0 & Y_{31} & 0 & Y_{32} & Y_{33} \end{bmatrix}$$

Index sets for monotone neighborhoods

for $i = 1, \ldots, n$, index set γ_i contains elements of $\mathrm{col}(i)$, in numerical order



$$
\begin{aligned}
\gamma_1 &= (1, 2, 9, 10) \\
\gamma_2 &= (2, 9, 10) \\
\gamma_3 &= (3, 9, 16) \\
\gamma_4 &= (4, 6, 7) \\
\gamma_5 &= (5, 6, 7, 8, 10) \\
&\vdots
\end{aligned}
$$

Index sets for (supernodal) elimination trees

all algorithms will use recursions (in topological or inverse topological order) over

- the (nodal) elimination tree,
- or a supernodal elimination tree (clique tree)

the following notation will make the algorithm descriptions almost identical

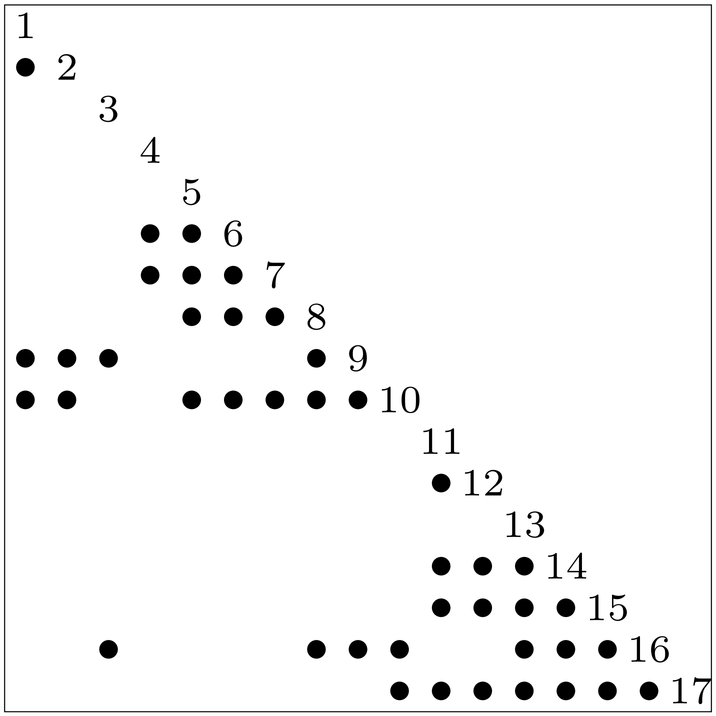
Recursions over elimination tree

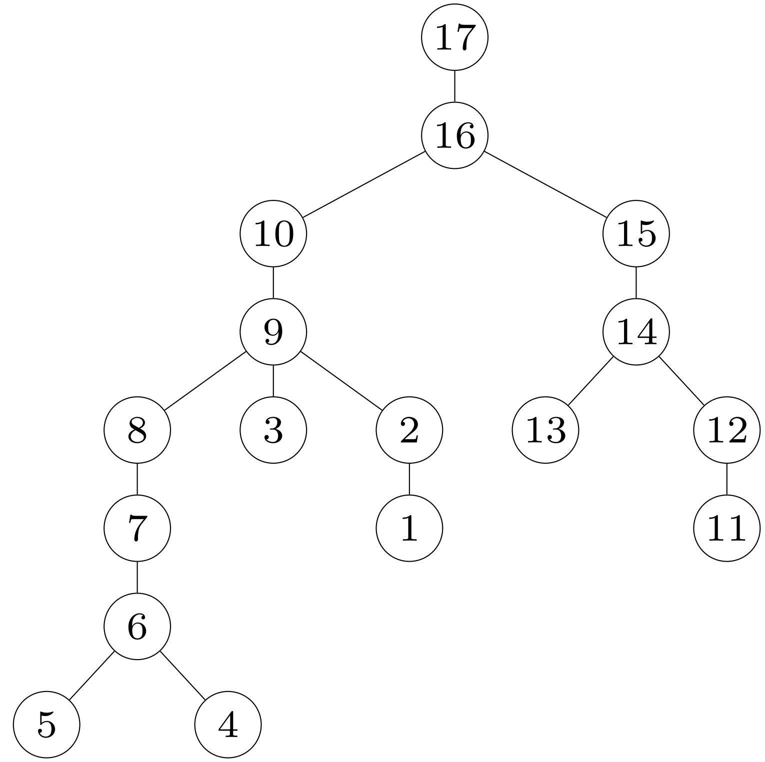
- $\nu_i = i$ for $i \in V = \{1, 2, \ldots, n\}$
- α_i: index set with elements of $\mathrm{col}(i) \setminus \{i\}$ in numerical order

Recursions over supernodal elimination tree

- $V^{\mathrm{c}} \subset V$ is set of clique representatives
- ν_i for $i \in V^{\mathrm{c}}$: index set with elements of $\mathrm{snd}(i)$ in numerical order
- α_i: index set with elements of $\mathrm{col}(i) \setminus \mathrm{snd}(i)$ in numerical order

Example: recursion over elimination tree



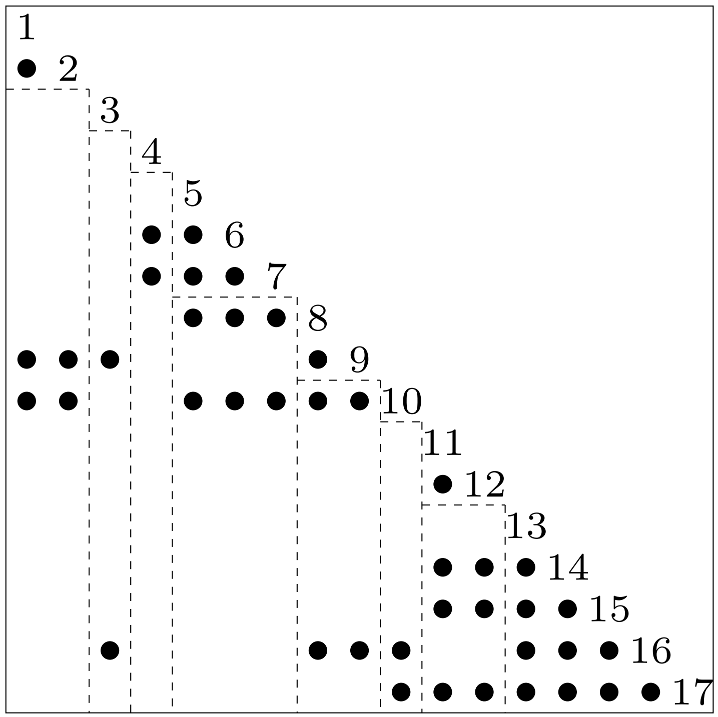


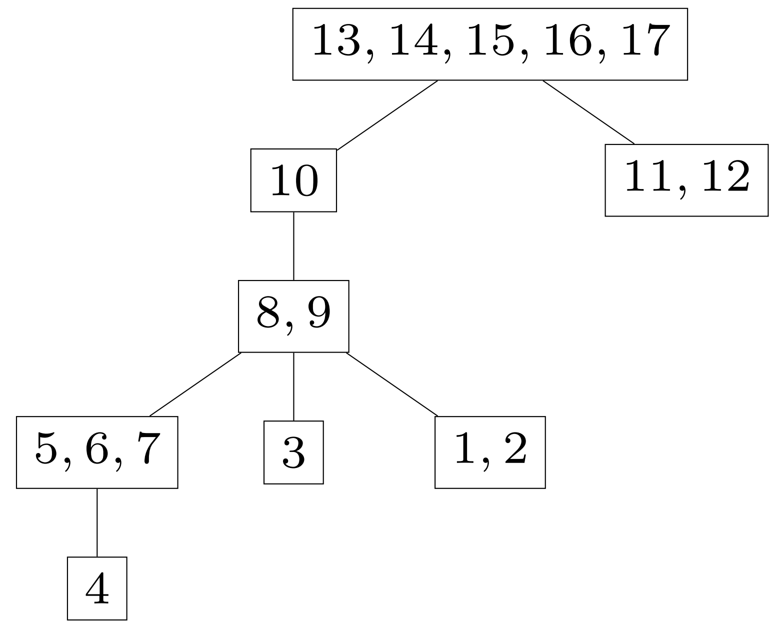
- in a recursion over the vertices of the elimination tree:

$$\nu_5 = 5, \qquad \alpha_5 = (6, 7, 8, 10), \qquad \gamma_5 = (5, 6, 7, 8, 10)$$

- elements of α_i are ancestors of vertex i

Example: recursion over supernodal elimination tree





- in a recursion over the supernodes of the supernodal elimination tree:

$$\nu_5 = (5, 6, 7), \qquad \alpha_5 = (8, 10), \qquad \gamma_5 = (5, 6, 7, 8, 9)$$

- elements of α_i are in supernodes ν_j that are ancestors of ν_i

II. Sparse matrices

- symmetric sparse matrices
- **positive semidefinite matrices**
  - Cholesky factorization
  - clique decomposition
  - multifrontal factorization
  - projected inverse
  - logarithmic barrier
- positive semidefinite completion
  - clique decomposition
  - minimum rank positive semidefinite completion
  - maximum determinant completion
  - logarithmic barrier
- Euclidean distance matrix completion
  - Euclidean distance matrices
  - clique decomposition
  - minimum dimension completion

Sparse Cholesky factorization

$$P_\sigma A P_\sigma^T = LDL^T$$

- A is positive definite
- P_σ is a permutation matrix
- L is unit lower triangular, D positive diagonal
- can be defined for singular positive semidefinite A if we allow zero D_{ii}

Sparsity pattern

$$P_\sigma^T(L + L^T)P_\sigma \in \mathbf{S}^n_{E'}$$

- $E' = E^*_\sigma$ is the edge set of the elimination graph of (V, E, σ) (see page 51)
- fill-in $E' \setminus E$ determines positions of added nonzeros

Cholesky factorization and chordal sparsity

Chordal pattern

if $A \in \mathbf{S}^n_E$ is positive definite and σ is a perfect elimination ordering for E, then

$$P_\sigma^T(L + L^T)P_\sigma \in \mathbf{S}^n_E$$

A has a 'zero fill' Cholesky factorization

Non-chordal pattern

if E is not chordal, then for every σ there exist positive definite $A \in \mathbf{S}^n_E$ for which

$$P_\sigma^T(L + L^T)P_\sigma \notin \mathbf{S}^n_E$$

(Rose 1970)

Sparse positive semidefinite matrix cone

we denote the set of positive semidefinite matrices with sparsity pattern E as

$$\mathbf{S}_+^n \cap \mathbf{S}_E^n = \{X \in \mathbf{S}_E^n \mid X \succeq 0\}$$

Properties

- a closed convex cone: intersection of closed convex cone ($\mathbf{S}_+^n$) and subspace
- nonempty interior with respect to $\mathbf{S}_E^n$: identity matrix I is in the interior
- pointed: $X \in \mathbf{S}_+^n \cap \mathbf{S}_E^n$ and $-X \in \mathbf{S}_+^n \cap \mathbf{S}_E^n$ only if $X = 0$

these properties hold for general sparsity patterns E

Positive semidefinite matrices with chordal sparsity

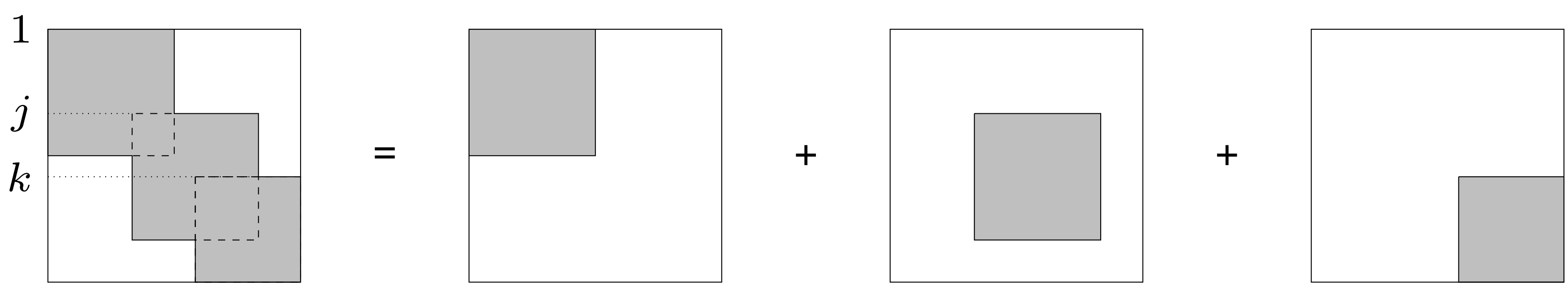
Decomposition theorem (for chordal E)

$A \in \mathbf{S}^n_E$ is positive semidefinite if and only if it can be expressed as

$$A = \sum_{i \in V^{\mathrm{c}}} P_{\gamma_i}^T H_i P_{\gamma_i} \qquad \text{with } H_i \succeq 0$$

(recall definition of P_β on page 73 and of γ_i on page 75)

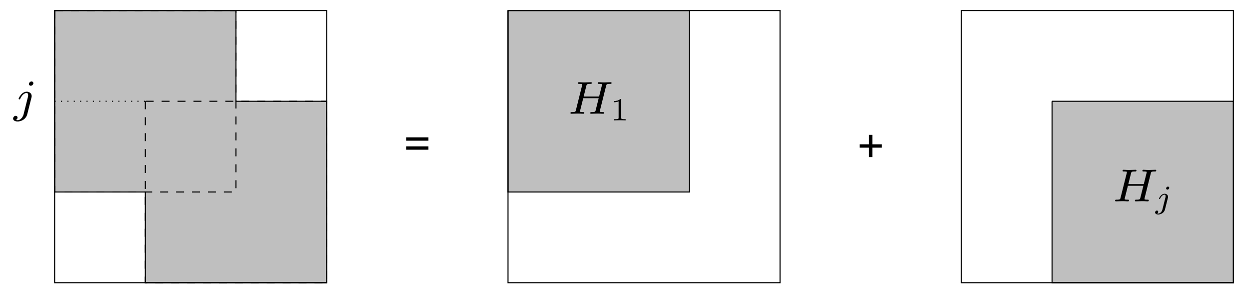
Example: three overlapping dense diagonal blocks



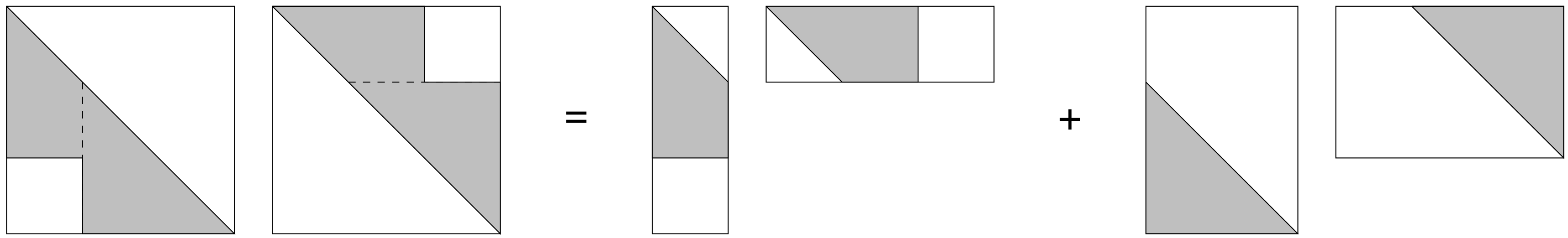
$A \succeq 0$ $\qquad P_{\gamma_1}^T H_1 P_{\gamma_1} \succeq 0$ $\qquad P_{\gamma_j}^T H_j P_{\gamma_j} \succeq 0$ $\qquad P_{\gamma_k}^T H_k P_{\gamma_k} \succeq 0$

(Griewank and Toint 1984, Agler, Helton, McCullough, Rodman 1988)

Proof (two cliques)



H_1 and H_j follow by combining columns in Cholesky factorization



Proof (general chordal pattern)

$$A = LDL^T = \sum_{j=1}^{n} D_{jj} L_j L_j^T$$

group outer products per maximal supernode $\mathrm{snd}(i)$:

$$\begin{aligned}
A &= \sum_{i \in V^{\mathrm{c}}} \sum_{j \in \mathrm{snd}(i)} D_{jj} L_j L_j^T \\
&= \sum_{i \in V^{\mathrm{c}}} \sum_{j \in \mathrm{snd}(i)} D_{jj} P_{\gamma_j}^T L_{\gamma_j j} L_{\gamma_j j} P_{\gamma_j} \\
&= \sum_{i \in V^{\mathrm{c}}} P_{\gamma_i}^T \left(\sum_{j \in \mathrm{snd}(i)} D_{jj} L_{\gamma_i j} L_{\gamma_i j} \right) P_{\gamma_i} \\
&= \sum_{i \in V^{\mathrm{c}}} P_{\gamma_i}^T H_i P_{\gamma_i}
\end{aligned}$$

line 3 follows because $\gamma_j \subset \gamma_i$ for $j \in \mathrm{snd}(i)$

Multifrontal Cholesky factorization

- a recursion over elimination tree in topological order (Duff and Reid 1983)
- we assume the sparsity pattern is chordal (or a chordal extension)

Factorization and elimination tree: nonzeros in column j of $A = LDL^T$

$$\begin{aligned}
\begin{bmatrix} A_{jj} \\ A_{\alpha_j j} \end{bmatrix} &= D_{jj} \begin{bmatrix} 1 \\ L_{\alpha_j} \end{bmatrix} + \sum_{k<j} D_{kk} L_{jk} \begin{bmatrix} L_{jk} \\ L_{\alpha_j k} \end{bmatrix} \\
&= D_{jj} \begin{bmatrix} 1 \\ L_{\alpha_j} \end{bmatrix} + \sum_{\substack{\text{strict descendants} \\ k \text{ of } j}} D_{kk} L_{jk} \begin{bmatrix} L_{jk} \\ L_{\alpha_j k} \end{bmatrix}
\end{aligned}$$

- α_j is index set with nonzeros below diagonal (page 76)
- no sum over $k > j$ on first line because $L_{jk} = 0$ for $k < j$
- second line because $L_{jk} = 0$ if k is not a descendant of j in elimination tree
- algorithm propagates intermediate variable for efficient computation of the sum

Update matrix

for each vertex j, (temporarily) store a dense **update matrix**

$$U_j = \sum_{k \in T_j} D_{kk} L_{\alpha_j j} L_{\alpha_j j}^T$$

T_j is the set of descendants of j in the elimination tree (a subtree with root j)

Recursion: U_j, D_{jj}, $L_{\gamma_j j}$ can be computed from

$$\left[\begin{array}{cc} A_{jj} & A_{\alpha_j j}^T \\ A_{\alpha_j j} & U_j \end{array}\right] = D_{jj} \left[\begin{array}{c} 1 \\ L_{\alpha_j j} \end{array}\right] \left[\begin{array}{c} 1 \\ L_{\alpha_j j} \end{array}\right]^T + P_{\gamma_j} \left(\sum_{i \text{ is child of } j} P_{\alpha_i}^T U_i P_{\alpha_i} \right) P_{\gamma_j}^T$$

given A_{jj}, $A_{\alpha_j,j}$ and the update matrices U_i for the children of j,

- we compute D_{jj} from the 1,1 element of equation
- $L_{\alpha_j j}$ from the 2,1 block
- U_j from the 2,2 block

Multifrontal algorithm

enumerate the vertices of the elimination tree in topological order

- at vertex j, first form the **frontal matrix**

$$\left[\begin{array}{cc} F_{11} & F_{12} \\ F_{21} & F_{22} \end{array}\right] = \left[\begin{array}{cc} A_{jj} & A_{\alpha_j j}^T \\ A_{\alpha_j j} & 0 \end{array}\right] - P_{\gamma_j}\left(\sum_{\text{children } i \text{ of } j} P_{\alpha_i}^T U_i P_{\alpha_i}\right) P_{\gamma_j}^T$$

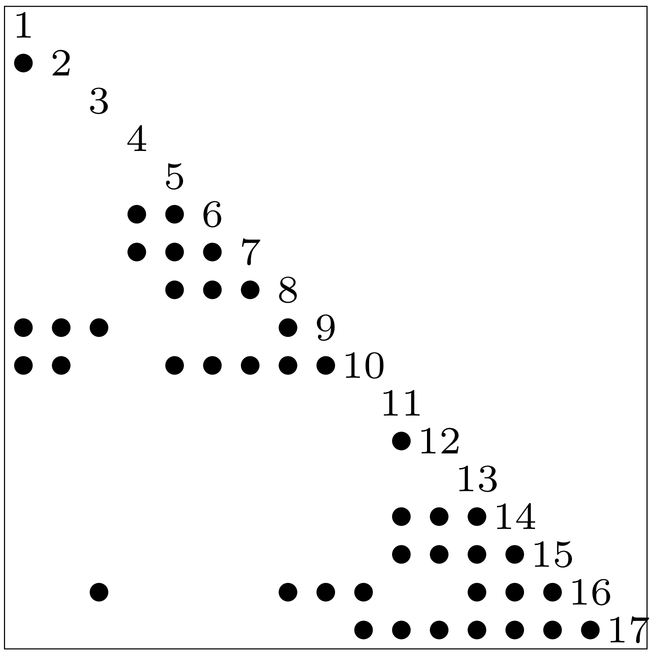
- then solve the equation

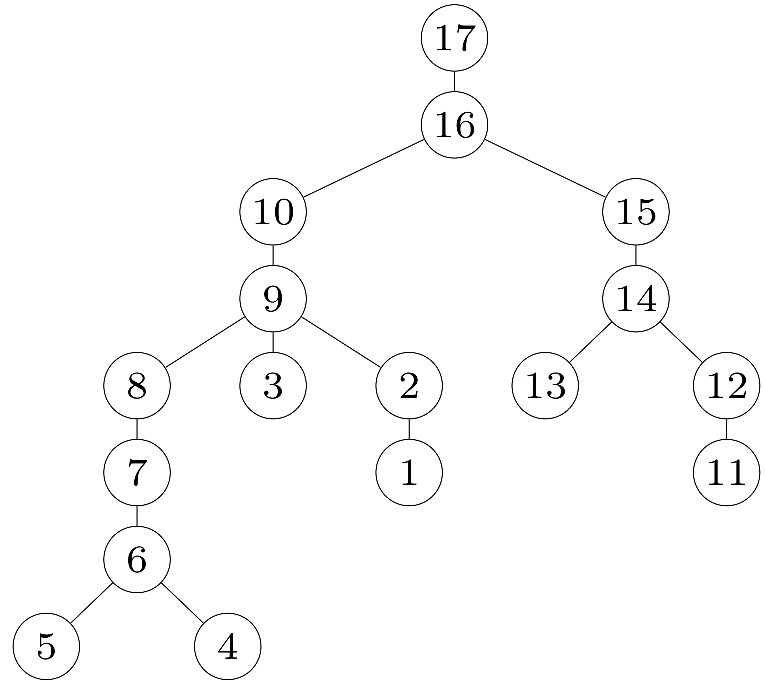
$$\left[\begin{array}{cc} F_{11} & F_{12} \\ F_{21} & F_{22} \end{array}\right] = D_{jj}\left[\begin{array}{c} 1 \\ L_{\alpha_j j} \end{array}\right]\left[\begin{array}{c} 1 \\ L_{\alpha_j j} \end{array}\right]^T + \left[\begin{array}{cc} 0 & 0 \\ 0 & U_j \end{array}\right]$$

  to find column j of the factorization and the update matrix U_j:

$$D_{jj} = F_{11}, \qquad L_{\alpha_j j} = \frac{1}{D_{jj}} F_{21}, \qquad U_j = -F_{22} + D_{jj} L_{\alpha_j j} L_{\alpha_j j}^T$$

Example





frontal matrix for index 9:

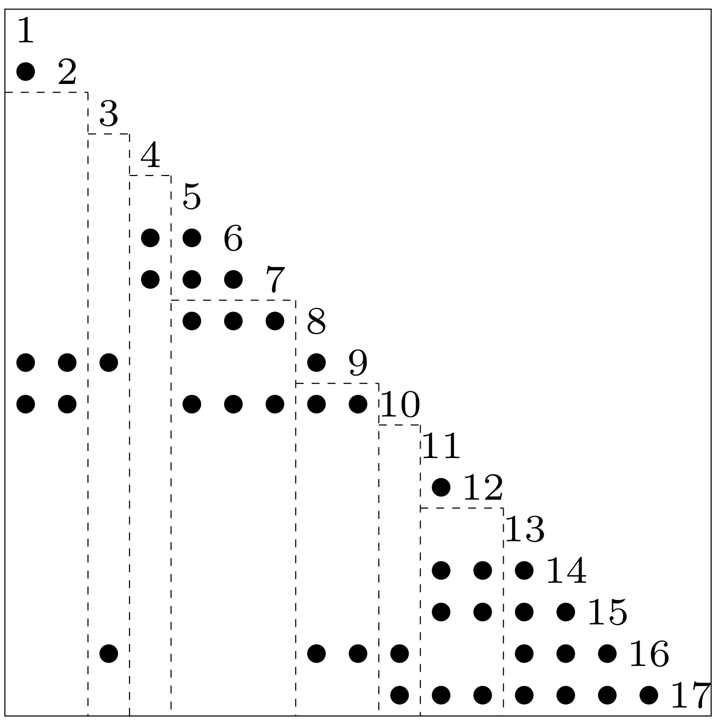
$$F = \begin{bmatrix} A_{99} & A_{9,10} & A_{9,16} \\ A_{10,9} & 0 & 0 \\ A_{16,9} & 0 & 0 \end{bmatrix} - \begin{bmatrix} (U_8)_{11} & (U_8)_{12} & (U_8)_{13} \\ (U_8)_{21} & (U_8)_{22} & (U_8)_{23} \\ (U_8)_{31} & (U_8)_{32} & (U_8)_{33} \end{bmatrix}$$

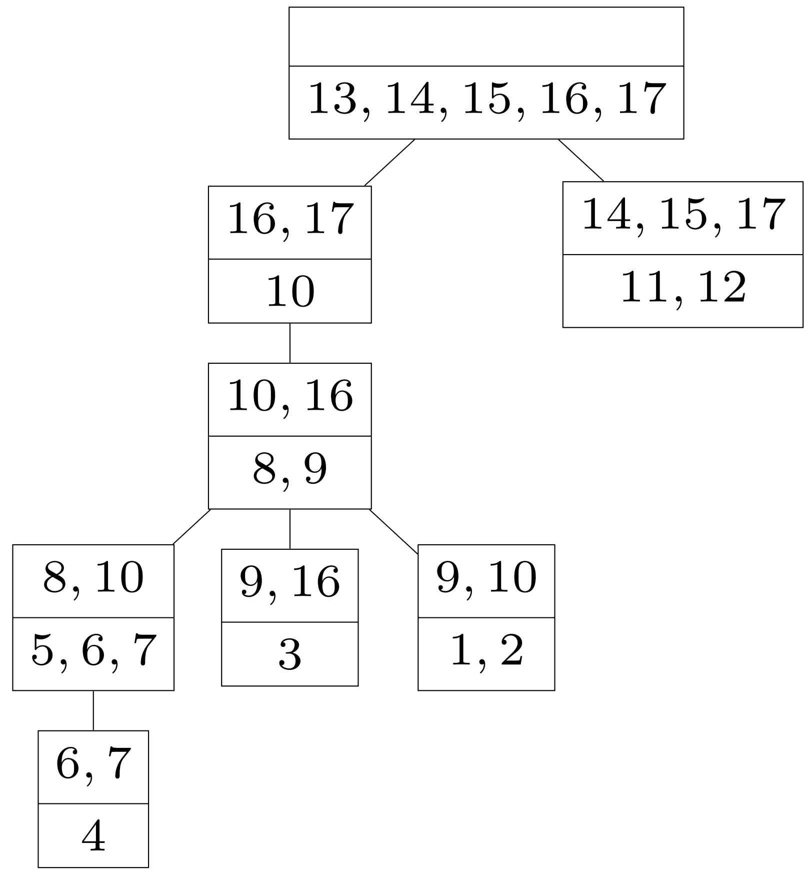
$$- \begin{bmatrix} (U_3)_{11} & 0 & (U_3)_{12} \\ 0 & 0 & 0 \\ (U_3)_{21} & 0 & (U_3)_{22} \end{bmatrix} - \begin{bmatrix} (U_2)_{11} & (U_2)_{21} & 0 \\ (U_2)_{21} & (U_2)_{22} & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Block Cholesky factorization

$$A = LDL^T$$

- D block diagonal, with positive definite diagonal blocks $D_{\nu_j \nu_j}$ for $j \in V^c$
- L lower triangular with $L_{\nu_j \nu_j} = I$, nonzero blocks $L_{\alpha_j \nu_j}$





Supernodal multifrontal algorithm

enumerate the vertices of the supernodal elimination tree in topological order

- at vertex $j \in V^{\mathrm{c}}$, form the supernodal frontal matrix

$$\left[\begin{array}{cc} F_{11} & F_{12} \\ F_{21} & F_{22} \end{array}\right] = \left[\begin{array}{cc} A_{\nu_j\nu_j} & A_{\alpha_j\nu_j}^T \\ A_{\alpha_j\nu_j} & 0 \end{array}\right] - P_{\gamma_j}\left(\sum_{\text{children } i \text{ of } j} P_{\alpha_i}^T U_i P_{\alpha_i}\right) P_{\gamma_j}^T$$

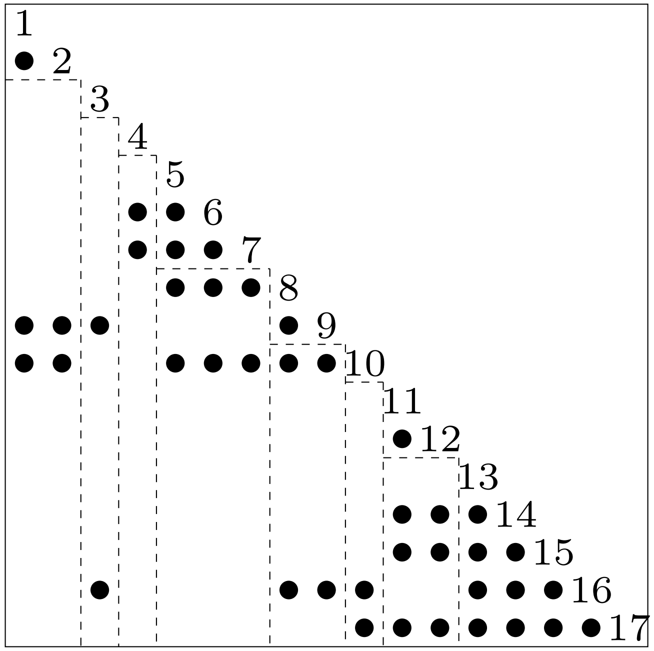
- then solve the equation

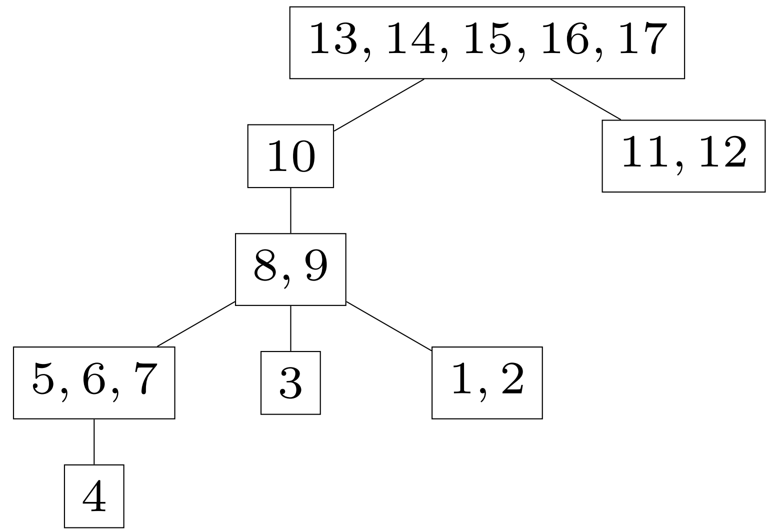
$$\left[\begin{array}{cc} F_{11} & F_{12} \\ F_{21} & F_{22} \end{array}\right] = \left[\begin{array}{c} I \\ L_{\alpha_j\nu_j} \end{array}\right] D_{\nu_j\nu_j} \left[\begin{array}{c} I \\ L_{\alpha_j\nu_j} \end{array}\right]^T + \left[\begin{array}{cc} 0 & 0 \\ 0 & U_j \end{array}\right]$$

to find block column ν_j of the factorization and the update matrix U_j:

$$D_{\nu_j\nu_j} = F_{11}, \qquad L_{\alpha_j j} = F_{21} D_{\nu_j\nu_j}^{-1}, \qquad U_j = -F_{22} + L_{\alpha_j\nu_j} D_{\nu_j\nu_j} L_{\alpha_j\nu_j}^T$$

Example



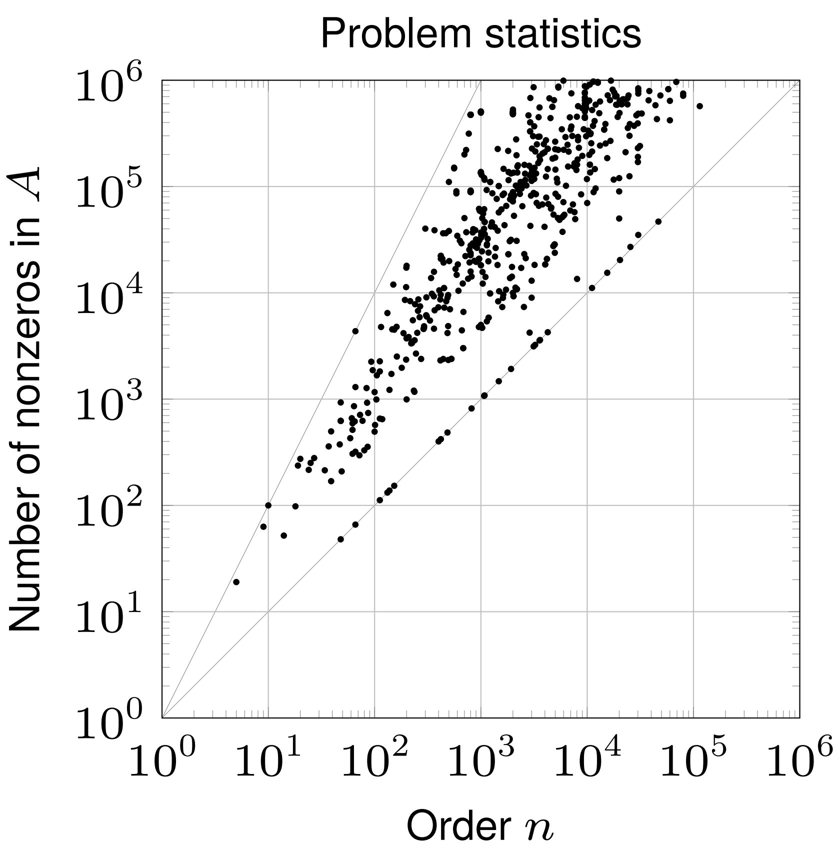


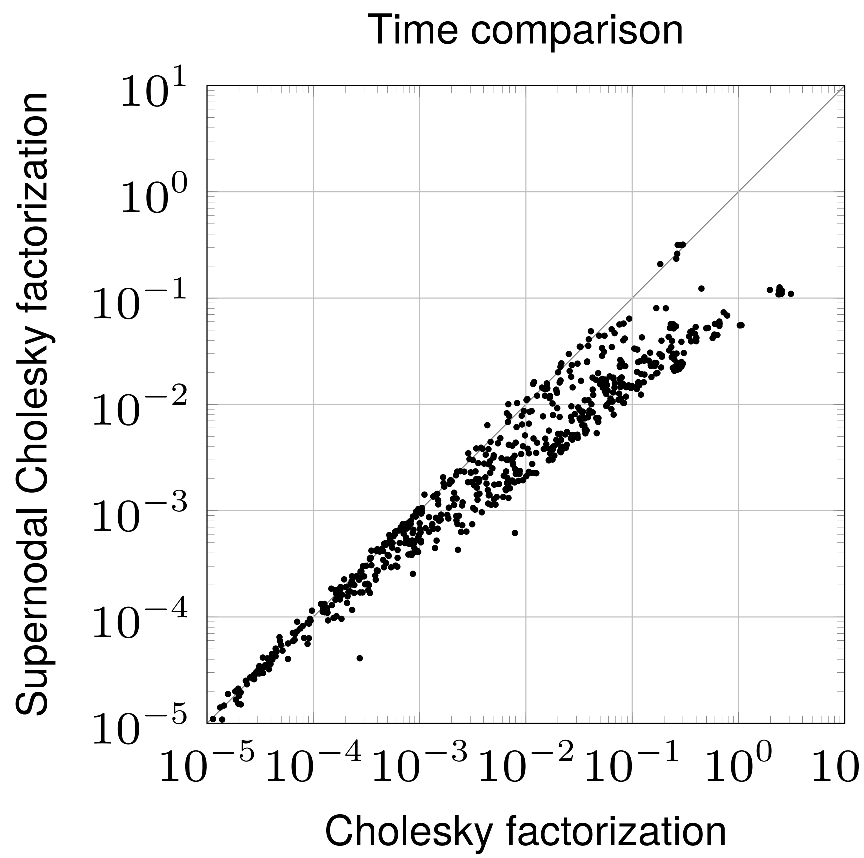
frontal matrix in step for supernode $\{8, 9\}$:

$$F = \begin{bmatrix} A_{88} & A_{89} & A_{8,10} & A_{8,16} \\ A_{99} & A_{99} & A_{9,10} & A_{9,16} \\ A_{10,8} & A_{10,9} & 0 & 0 \\ A_{16,8} & A_{16,9} & 0 & 0 \end{bmatrix} - \begin{bmatrix} (U_5)_{11} & 0 & (U_5)_{12} & 0 \\ 0 & 0 & 0 & 0 \\ (U_5)_{21} & 0 & (U_5)_{22} & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$- \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & (U_3)_{11} & 0 & (U_3)_{12} \\ 0 & 0 & 0 & 0 \\ 0 & (U_3)_{21} & 0 & (U_3)_{22} \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & (U_1)_{11} & (U_1)_{12} & 0 \\ 0 & (U_1)_{21} & (U_1)22 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Comparison





- 667 patterns from University of Florida Sparse Matrix Collection
- time in seconds for supernodal and nodal Cholesky factorization
- code at `cvxopt.github.io/chompack`

Projected inverse

we consider the problem of computing

$$\Pi_E(A^{-1})$$

for a positive definite matrix $A \in \mathbf{S}^n_E$ with chordal pattern E

- Π_E denotes projection on $\mathbf{S}^n_E$:

$$\Pi_E(X) = \begin{cases} X_{ij} & i = j \text{ or } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

- the complete inverse A^{-1} is usually dense and expensive to compute
- we are interested in computing $\Pi_E(A^{-1})$ without computing the entire inverse

Projected inverse from Cholesky factorization

we assume the sparsity pattern of A is chordal

- from Cholesky factorization $A = LDL^T$:

$$A^{-1}L = L^{-T}D^{-1}$$

- block $\gamma_j \times \gamma_j$ of projected inverse $B = \Pi_E(A^{-1})$ satisfies

$$\begin{bmatrix} B_{jj} & B_{\alpha_j j}^T \\ B_{\alpha_j j} & B_{\alpha_j \alpha_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{\alpha_j j} \end{bmatrix} = \begin{bmatrix} 1/D_{jj} \\ 0 \end{bmatrix}$$

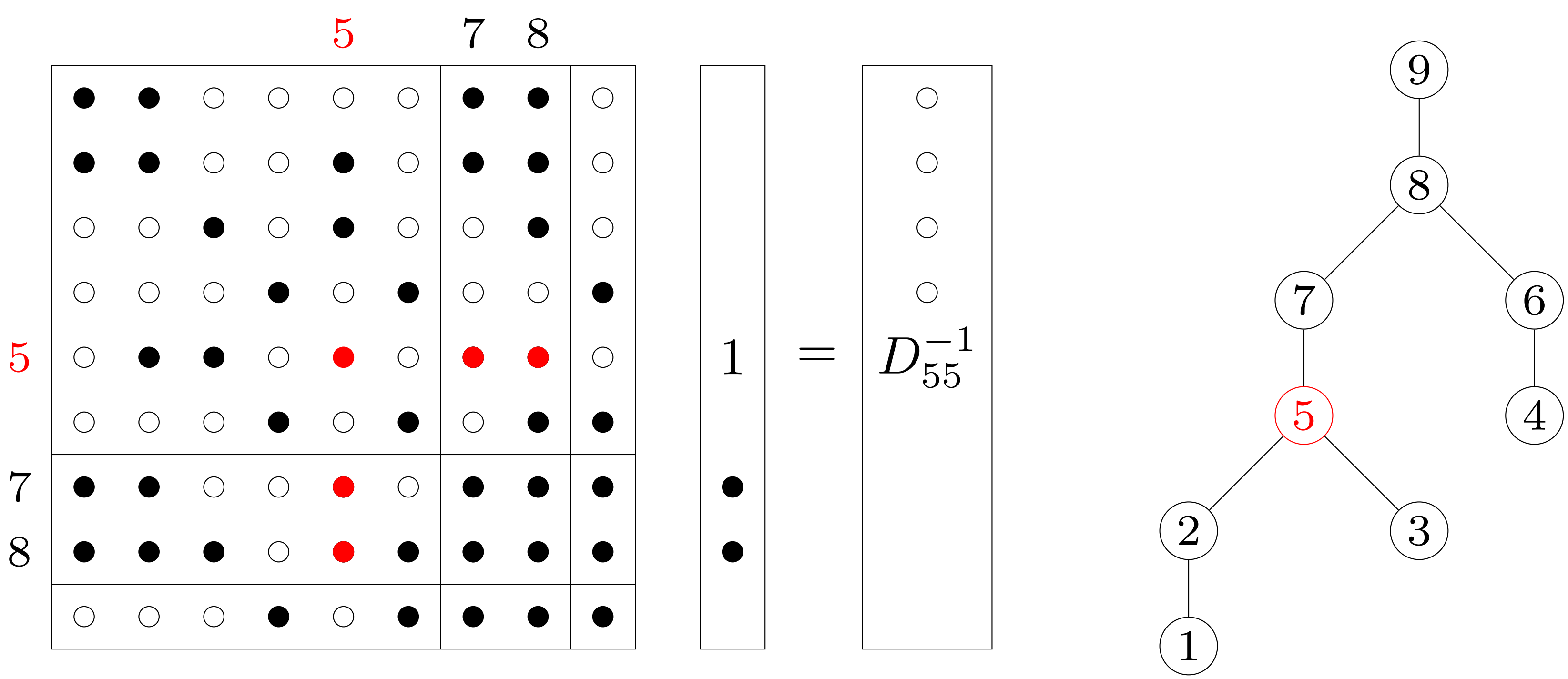
  right-hand side follows because L^{-T} is unit upper triangular

- this equation allows us to compute $B_{\alpha_j j}$ and B_{jj} from $B_{\alpha_j \alpha_j}$ (and L, D)
- the elements of α_j are ancestors of j in the elimination tree

hence B can be computed, column by column, in an *inverse* topological order

Example

$$\begin{bmatrix} B_{jj} & B_{\alpha_j j}^T \\ B_{\alpha_j j} & B_{\alpha_j \alpha_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{\alpha_j j} \end{bmatrix} = \begin{bmatrix} 1/D_{jj} \\ 0 \end{bmatrix}$$



- filled circles: entries that are known or to be computed
- open circles: nonzero but unknown or irrelevant

‘Multifrontal’ algorithm for projected inverse

$$\begin{bmatrix} B_{jj} & B_{\alpha_j j}^T \\ B_{\alpha_j j} & B_{\alpha_j \alpha_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{\alpha_j j} \end{bmatrix} = \begin{bmatrix} 1/D_{jj} \\ 0 \end{bmatrix}$$

Algorithm: recursion over elimination tree in inverse topological order

- at vertex j, compute

$$B_{\alpha_j j} = -U_j L_{\alpha_j j}, \qquad B_{jj} = \frac{1}{D_{jj}} - B_{\alpha_j j}^T L_{\alpha_j j}$$

  $U_j = B_{\alpha_j \alpha_j}$ is dense ‘update matrix’

- for each child i of j, form

$$U_i = P_{\alpha_i} P_{\gamma_j}^T \begin{bmatrix} B_{jj} & B_{\alpha_j j}^T \\ B_{\alpha_j j} & U_j \end{bmatrix} P_{\gamma_j} P_{\alpha_i}^T$$

main step is dense matrix-vector multiplication $U_j L_{\alpha_j j}$

Supernodal algorithm for projected inverse

$$\left[\begin{array}{cc} B_{\nu_j\nu_j} & B_{\alpha_j\nu_j}^T \\ B_{\alpha_j\nu_j} & B_{\alpha_j\alpha_j} \end{array}\right]\left[\begin{array}{c} I \\ L_{\alpha_j\nu_j} \end{array}\right] = \left[\begin{array}{c} D_{\nu_j\nu_j}^{-1} \\ 0 \end{array}\right]$$

Algorithm: recrusion over supernodal elimination tree in inverse topological order

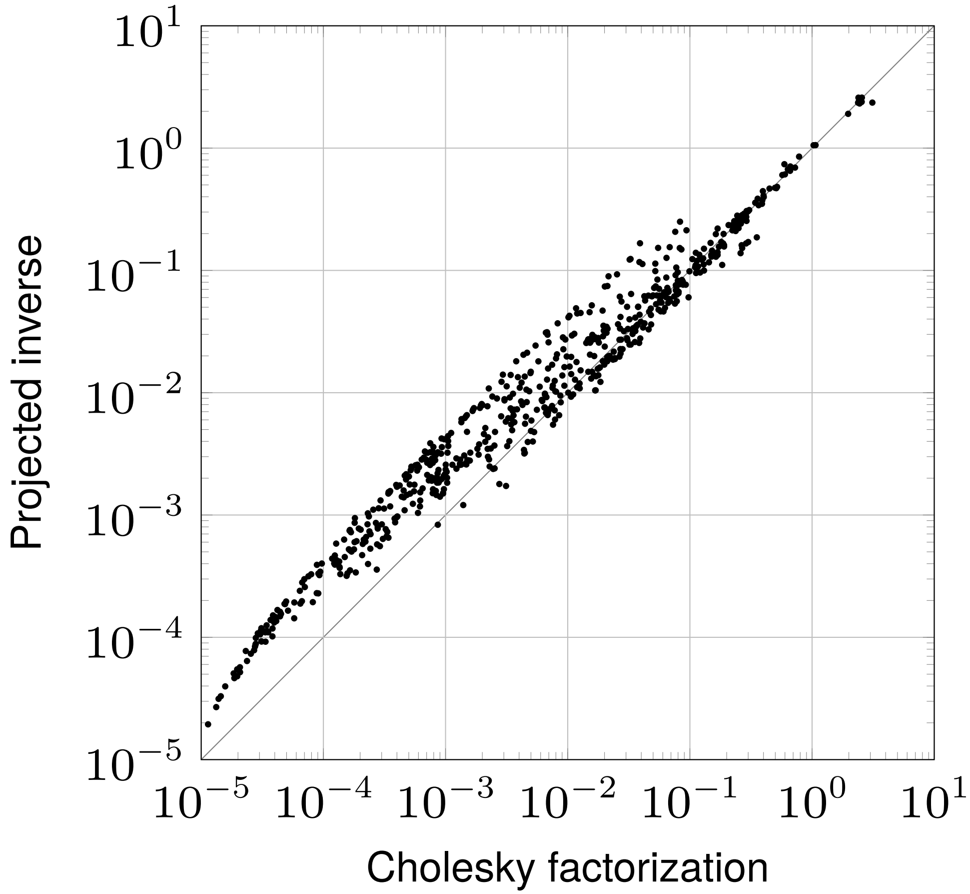
- at vertex $j \in V^{\mathrm{c}}$, compute

$$B_{\alpha_j\nu_j} = -U_j L_{\alpha_j\nu_j}, \qquad B_{\nu_j\nu_j} = D_{\nu_j\nu_j}^{-1} - B_{\alpha_j\nu_j}^T L_{\alpha_j\nu_j}$$

- for each child i of j, form

$$U_i = P_{\alpha_i} P_{\gamma_j}^T \left[\begin{array}{cc} B_{\nu_j\nu_j} & B_{\alpha_j\nu_j}^T \\ B_{\alpha_j\nu_j} & U_j \end{array}\right] P_{\gamma_j} P_{\alpha_i}^T$$

Projected inverse versus Cholesky factorization



- 667 test patterns from page 92
- time in seconds for projected inverse and Cholesky factorization
- code at `cvxopt.github.io/chompack`

Logarithmic barrier for positive semidefinite cone

Definition: the function $\phi : \mathbf{S}^n_E \to \mathbf{R}$ with

$$\phi(S) = -\log\det S, \qquad \mathrm{dom}\,\phi = \{S \in \mathbf{S}^n_E \mid S \succ 0\}$$

Value: efficiently computed from Cholesky factorization $S = LDL^T$

Gradient: the negative of the projected inverse

$$\nabla\phi(S) = -\Pi_E(S^{-1})$$

Hessian: for arbitrary $Y \in \mathbf{S}^n_E$,

$$\nabla^2\phi(S)[Y] = \frac{d}{dt}\nabla\phi(S+tY)\bigg|_{t=0} = \Pi_E\left(S^{-1}YS^{-1}\right)$$

Hessian

Gradient evaluation: for chordal E, computing

$$\nabla\phi(S) = -\Pi_E(S^{-1})$$

requires two recursions over elimination tree

- Cholesky factorization $S = LDL^T$ (recursion in topological order)
- projected inverse from D, L (recursion in inverse topological order)

Algorithm for Hessian evaluation:

linearize the recursions in the gradient algorithm to compute

$$\nabla^2\phi(S)[Y] = \Pi_E(S^{-1}YS^{-1}) = -\frac{d}{dt}\Pi_E(S+tY)^{-1}\Big|_{t=0}$$

two recursions: one in topological, one in inverse topological order

Factorization of Hessian

the linearized recursions in the evaluation of $\nabla^2\phi(S)[Y]$ turn out to be adjoints

- this gives a factorization $\nabla^2\phi(S) = \mathcal{R}_S^* \circ \mathcal{R}_S$:

$$\nabla^2\phi(S)[Y] = \mathcal{R}_S^*(\mathcal{R}_S(Y))$$

- $\mathcal{R}_S(Y)$ and $\mathcal{R}_S(Y)^{-1}$ can be computed by a recursion in topological order
- $\mathcal{R}_S^*(Y)$ and $(\mathcal{R}_S^*)^{-1}(Y)$ computed by a recursion in inverse topological order
- this also provides an algorithm for applying the inverse Hessian

$$\nabla^2\phi(S)^{-1}[Y] = \mathcal{R}_S^{-1}((\mathcal{R}_S^*)^{-1}(Y))$$

(Andersen, Dahl, Vandenberghe 2012)

II. Sparse matrices

- symmetric sparse matrices
- positive semidefinite matrices
  - Cholesky factorization
  - clique decomposition
  - multifrontal factorization
  - projected inverse
  - logarithmic barrier
- **positive semidefinite completion**
  - clique decomposition
  - minimum rank positive semidefinite completion
  - maximum determinant completion
  - logarithmic barrier
- Euclidean distance matrix completion
  - Euclidean distance matrices
  - clique decomposition
  - minimum dimension completion

Positive semidefinite completable matrix cone

we denote the set of matrices in $\mathbf{S}^n_E$ that have a positive semidefinite completion by

$$\Pi_E(\mathbf{S}^n_+) = \{\Pi_E(X) \mid X \in \mathbf{S}^n_+\}$$

Properties

- a convex cone: the projection of a convex cone on a subspace
- has nonempty interior (relative to $\mathbf{S}^n_E$): the identity matrix is in the interior
- pointed: if $A = \Pi_E(X)$ and $-A = \Pi_E(Y)$ for some $X, Y \succeq 0$, then

$$\begin{aligned} \Pi_E(X+Y) = 0 \quad &\Longrightarrow \quad \mathbf{diag}(X) = \mathbf{diag}(Y) = 0 \\ &\Longrightarrow \quad X = Y = 0 \end{aligned}$$

- closed because $\Pi_E(X) = 0$, $X \succeq 0$ only if $X = 0$

Duality

the positive semidefinite and positive semidefinite completable cones are duals

Dual of positive semidefinite completable cone

$$\begin{aligned}
(\Pi_E(\mathbf{S}^n_+))^* &= \{B \in \mathbf{S}^n_E \mid \mathbf{tr}(AB) \geq 0 \ \ \forall A \in \Pi_E(\mathbf{S}^n_+)\} \\
&= \{B \in \mathbf{S}^n_E \mid \mathbf{tr}\left(\Pi_E(X)B\right) \geq 0 \ \ \forall X \succeq 0\} \\
&= \{B \in \mathbf{S}^n_E \mid \mathbf{tr}(XB) \geq 0 \ \ \forall X \succeq 0\} \\
&= \mathbf{S}^n_+ \cap \mathbf{S}^n_E
\end{aligned}$$

Dual of positive semidefinite cone

$$(\mathbf{S}^n_+ \cap \mathbf{S}^n_E)^* = \mathrm{cl}(\Pi_E(\mathbf{S}^n_+)) = \Pi_E(\mathbf{S}^n_+)$$

- step 1: the dual of the dual of a convex cone K is the closure of K
- step 2: we have seen that $\Pi_E(\mathbf{S}^n_+)$ is closed

Positive semidefinite completable cone with chordal sparsity

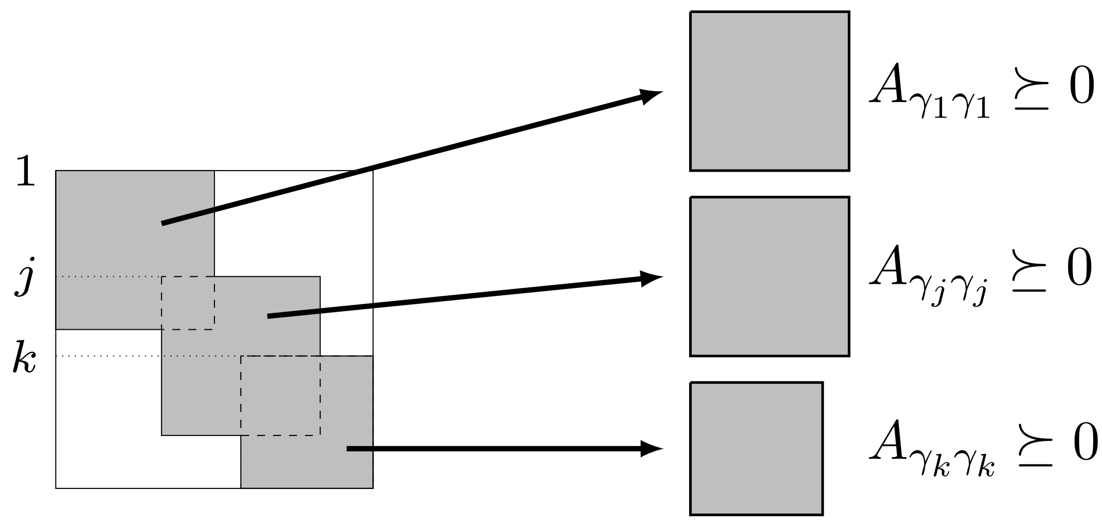
Decomposition theorem (for chordal E)

$A \in \mathbf{S}^n_E$ has a positive semidefinite completion if and only if

$$A_{\gamma_i\gamma_i} \succeq 0, \quad i \in V_{\mathrm{c}}$$

(recall that γ_i for $i \in V^{\mathrm{c}}$ are the cliques; see page 75)

Example: three overlapping dense diagonal blocks



PSD completable A

(Grone, Johnson, Sá, Wolkowicz, 1984)

Proof from duality

- positive semidefinite and PSD completable cones are dual cones:

$$A \in \Pi_E(\mathbf{S}^n_+) \qquad \Longleftrightarrow \qquad \mathbf{tr}(AB) \geq 0 \quad \forall B \in \mathbf{S}^n_+ \cap \mathbf{S}^n_E$$

- decomposition theorem (page 82): every $B \in \mathbf{S}^n_+ \cap \mathbf{S}^n_E$ can be written as

$$B = \sum_{i \in V^{\mathrm{c}}} P_{\gamma_i}^T H_i P_{\gamma_i}, \qquad \text{with } H_i \succeq 0$$

- therefore $A \in \Pi_E(\mathbf{S}^N_+)$ if and only if

$$0 \leq \mathbf{tr}\left(A \sum_{i \in V^{\mathrm{c}}} P_{\gamma_i}^T H_i P_{\gamma_i}\right) = \sum_{i \in V^{\mathrm{c}}} \mathbf{tr}\left(P_{\gamma_i} A P_{\gamma_i}^T H_i\right) \qquad \forall H_i \succeq 0$$

- this is equivalent to $P_{\gamma_i} A P_{\gamma_i}^T \succeq 0$ for all $i \in V^{\mathrm{c}}$

Minimum rank positive semidefinite completion

Positive semidefinite completion problem: given $A \in \Pi_E(\mathbf{S}^n_+)$, find X s.t.

$$A = \Pi_E(X), \qquad X \succeq 0$$

Minimum rank completion: if E is chordal, then there is a completion with

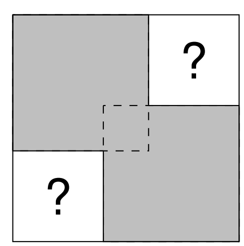
$$\operatorname{rank}(X) = \max_{i \in V^c} \operatorname{rank} A_{\gamma_i \gamma_i}$$

(Dancis 1992)

- this is the minimum possible rank, since for any PSD completion X,

$$\operatorname{rank}(X) \geq \max_{i \in V^c} \operatorname{rank} A_{\gamma_i \gamma_i}$$

- to show the result we first consider the simple two-block completion problem



Two-block completion problem

find the minimum rank positive semidefinite completion of

$$A = \begin{bmatrix} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{bmatrix}$$

- a completion exists if and only if

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \succeq 0, \qquad \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} \succeq 0$$

- define $r = \max\{r_1, r_2\}$ where

$$\mathrm{rank} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = r_1, \qquad \mathrm{rank} \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} = r_2$$

Two-block completion algorithm

- compute matrices U, V, $\tilde{V}$, W of column dimension r such that

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} U \\ V \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix}^T, \qquad \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} = \begin{bmatrix} \tilde{V} \\ W \end{bmatrix} \begin{bmatrix} \tilde{V} \\ W \end{bmatrix}^T$$

- since $VV^T = \tilde{V}\tilde{V}^T$, the matrices V and $\tilde{V}$ have SVDs

$$V = P\Sigma Q_1^T, \qquad \tilde{V} = P\Sigma Q_2^T$$

hence $V = \tilde{V}Q$ with $Q = Q_2 Q_1^T$ an orthogonal $r \times r$ matrix

- a completion of rank r is given by

$$\begin{bmatrix} UQ^T \\ \tilde{V} \\ W \end{bmatrix} \begin{bmatrix} UQ^T \\ \tilde{V} \\ W \end{bmatrix}^T = \begin{bmatrix} A_{11} & A_{12} & UQ^T W^T \\ A_{21} & A_{22} & A_{23} \\ WQU^T & A_{32} & A_{33} \end{bmatrix}$$

Minimum rank completion for general chordal pattern

we compute an $n \times r$ matrix Y with $\Pi_E(YY^T) = A$ and

$$r = \max_{i \in V^{\mathrm{c}}} \operatorname{rank}(A_{\gamma_i \gamma_i})$$

- the block rows Y_{ν_j} are computed in inverse topological order
- hence Y_{α_j} is known when we compute Y_{ν_j}

Algorithm: enumerate supernodes in inverse topological order

- at vertex $j \in V^{\mathrm{c}}$, compute matrices U_j, V_j with column dimension r such that

$$\left[\begin{array}{cc} A_{\nu_j \nu_j} & A_{\nu_j \alpha_j} \\ A_{\alpha_j \nu_j} & A_{\alpha_j \alpha_j} \end{array}\right] = \left[\begin{array}{c} U_j \\ V_j \end{array}\right] \left[\begin{array}{c} U_j \\ V_j \end{array}\right]^T$$

- if j is the root of the supernodal elimination tree, set $Y_{\nu_j} = U_j$
- otherwise, compute orthogonal Q such that $V_j = Y_{\alpha_j} Q$ and set $Y_{\nu_j} = U_j Q^T$

Outline of proof

suppose that when we visit vertex j, the already computed part of Y satisfies

$$Y_{\gamma_i} Y_{\gamma_i}^T = A_{\gamma_i \gamma_i}$$

for all $i \in V^{\mathrm{c}}$ that are ancestors of j in the supernodal elimination tree

- by assumption, Y_{α_j} is known when we visit supernode j, and satisfies

$$Y_{\alpha_j} Y_{\alpha_j}^T = A_{\alpha_j \alpha_j} = V_j V_j^T$$

- hence, there exists an orthogonal Q such that $V_j = Y_{\alpha_j} Q$
- the matrix $Y_{\nu_j} = U_j Q^T$ satisfies

$$\begin{bmatrix} Y_{\nu_j} \\ Y_{\alpha_j} \end{bmatrix} \begin{bmatrix} Y_{\nu_j} \\ Y_{\alpha_j} \end{bmatrix}^T = \begin{bmatrix} U_j U_j^T & U_j Q^T Y_{\alpha_j}^T \\ Y_{\alpha_j} Q U_j^T & Y_{\alpha_j} Y_{\alpha_j}^T \end{bmatrix} = \begin{bmatrix} A_{\nu_j \nu_j} & A_{\nu_j \alpha_j} \\ A_{\alpha_j \nu_j} & A_{\alpha_j \alpha_j} \end{bmatrix} = A_{\gamma_j \gamma_j}$$

Maximum determinant positive definite completion

Maximum determinant completion problem: for A in the interior of $\Pi_E(\mathbf{S}^n_+)$,

$$\begin{array}{ll} \text{maximize} & \log\det W \\ \text{subject to} & \Pi_E(W) = A \end{array}$$

with variable $W \in \mathbf{S}^n$

- we implicitly assume that the domain of the objective is $\mathbf{S}^n_{++}$
- also known as the maximum entropy completion:

$$\frac{1}{2}(\log\det W + n\log(2\pi) + n)$$

is the entropy of the normal distribution $N(0, W)$

Optimality conditions

the maximum determinant positive definite completion is the solution of

$$\begin{array}{ll} \text{minimize} & -\log\det W \\ \text{subject to} & \Pi_E(W) = A \end{array}$$

Lagrangian (using a Lagrange multiplier $Y \in \mathbf{S}^n_E$):

$$\begin{aligned} L(W,Y) &= -\log\det W + \mathbf{tr}(Y(\Pi_E(W) - A)) \\ &= -\log\det W + \mathbf{tr}(Y(W - A)) \end{aligned}$$

Optimality conditions

- feasibility: $W \succ 0$ and $\Pi_E(W) = A$
- gradient of Lagrangian with respect to W is zero: $W^{-1} = Y$
- hence W^{-1} is sparse, with sparsity pattern E

Dual of maximum determinant completion

Primal: minimize $-\log\det W$
subject to $\Pi_E(W) = A$

Dual: maximize $-\mathbf{tr}(AY) + \log\det Y + n$

dual variable is sparse matrix $Y \in \mathbf{S}^n_E$

Statistics interpretation

$\hat{\Sigma} = Y^{-1}$ is maximum likelihood estimate of $x \sim N(0, \Sigma)$, given:

- projection $\Pi_E(A)$ of sample covariance
- sparsity constraints that express conditional independence relations:

$$\{i,j\} \notin E \iff (\Sigma^{-1})_{ij} = 0$$

$$\iff x_i, x_j \text{ are conditionally independent}$$

Maximum determinant completion with chordal sparsity

$$\begin{array}{ll} \text{maximize} & \log\det W \\ \text{subject to} & \Pi_E(W) = A \end{array}$$

- for general E, can be solved by convex optimization methods
- for chordal E, explicit expressions

Cholesky factorization of inverse

- factors in $W^{-1} = LDL^T$ satisfy $WL = L^{-T}D^{-1}$
- block $\gamma_j \times \gamma_j$ in this equation is

$$\begin{bmatrix} A_{jj} & A_{\alpha_j j}^T \\ A_{\alpha_j} & A_{\alpha_j \alpha_j} \end{bmatrix} \begin{bmatrix} 1 \\ L_{\alpha_j j} \end{bmatrix} = \begin{bmatrix} 1/D_{jj} \\ 0 \end{bmatrix}$$

- solution is

$$L_{\alpha_j j} = -A_{\alpha_j \alpha_j}^{-1} A_{\alpha_j j}, \qquad D_{jj} = (A_{jj} + A_{\alpha_j j}^T L_{\alpha_j j})^{-1}$$

Algorithm for maximum determinant completion

enumerate the vertices of elimination tree in inverse topological order

- at vertex j, compute

$$L_{\alpha_j j} = -U_j^{-1} A_{\alpha_j j}, \qquad D_{jj} = (A_{jj} - A_{\alpha_j j}^T L_{\alpha_j j})^{-1}$$

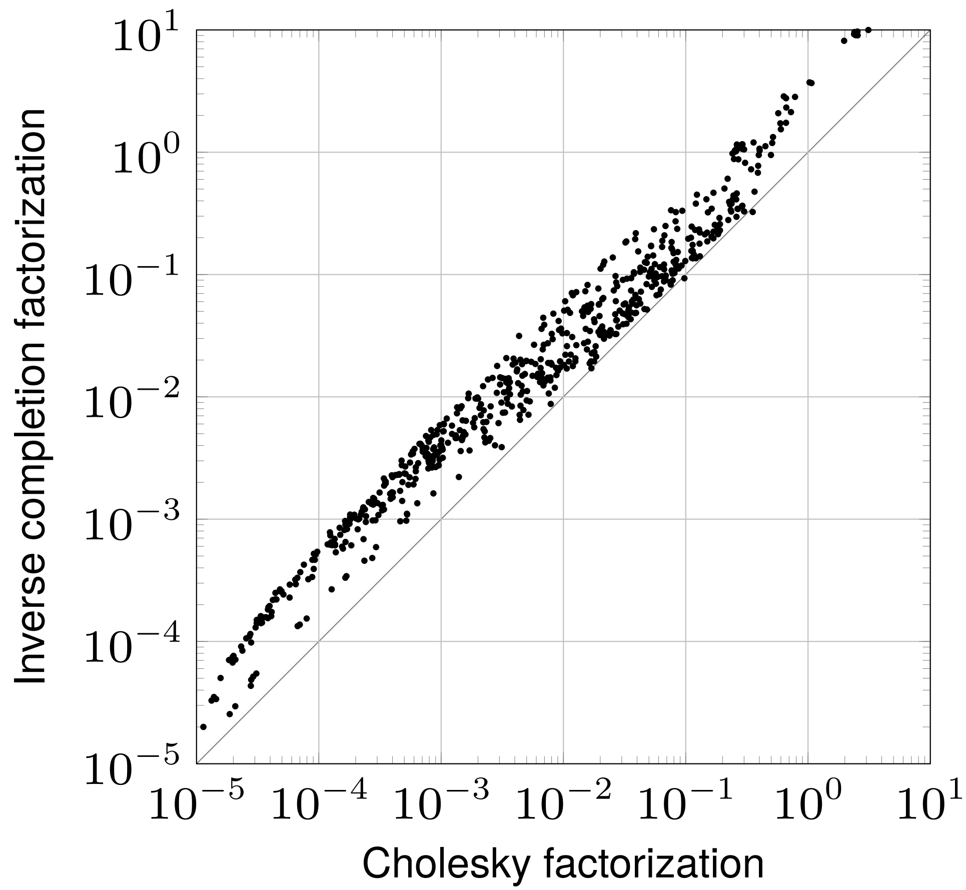
- for each child i of j, form

$$U_i = P_{\alpha_i}^T P_{\gamma_j}^T \left[\begin{array}{cc} A_{jj} & A_{\alpha_j j}^T \\ A_{\alpha_j j} & U_j \end{array} \right] P_{\gamma_j} P_{\alpha_i}^T$$

Comments

- U_i is simply $A_{\alpha_i \alpha_i}$, stored and updated as a dense matrix
- main step is solution of dense system $U_j L_{\alpha_j j} = -A_{\alpha_j j}$
- an improvement is to propagate factorization of U_j and make low-rank updates

Comparison with Cholesky factorization



- 667 test patterns from page 92
- supernodal version of algorithm on previous page vs. Cholesky factorization
- code at `cvxopt.github.io/chompack`

Barrier for positive semidefinite completable cone

$$\phi_*(X) = \sup_{S \in \mathbf{S}^n_{++} \cap \mathbf{S}^n_E} \left(-\mathbf{tr}(XS) + \log\det S\right)$$

with domain $\mathrm{dom}\,\phi_* = \{X = \Pi_E(Y) \mid Y \succ 0\}$

- this is the conjugate of the barrier $\phi(S) = -\log\det S$ for sparse PSD cone
- optimization problem in the definition is the dual of the completion problem

$$\begin{array}{ll} \text{minimize} & -\log\det Z \\ \text{subject to} & \Pi_E(Z) = X \end{array}$$

  (see page 113); optimal $\hat{S}$ in definition of $\phi_*(S)$ is $\hat{S} = Z^{-1}$
- for general E, barrier $\phi_*(X)$ must be computed by numerical optimization
- for chordal E, $\phi_*(X)$ can be computed by algorithms discussed earlier

Barrier ϕ_* for chordal sparsity pattern

suppose E is chordal and X is in the interior of $\Pi_E(\mathbf{S}^n_+)$

- to evaluate $\phi_*(X)$, we compute the (sparse) inverse of the solution of

$$\begin{array}{ll} \text{minimize} & -\log\det Z \\ \text{subject to} & \Pi_E(Z) = X \end{array}$$

  the algorithm of p.115 computes the inverse in factored form $\hat{S} = LDL^T$

- the value of the barrier is

$$\phi_*(X) = \log\det\widehat{S} - n$$

- gradient and Hessian of ϕ_* at X are

$$\nabla\phi_*(X) = -\widehat{S}, \qquad \nabla^2\phi_*(X) = \nabla^2\phi(\widehat{S})^{-1}$$

II. Sparse matrices

- symmetric sparse matrices
- positive semidefinite matrices
  - Cholesky factorization
  - clique decomposition
  - multifrontal factorization
  - projected inverse
  - logarithmic barrier
- positive semidefinite completion
  - clique decomposition
  - minimum rank positive semidefinite completion
  - maximum determinant completion
  - logarithmic barrier
- **Euclidean distance matrix completion**
  - Euclidean distance matrices
  - clique decomposition
  - minimum dimension completion

Euclidean distance matrix

Euclidean distance matrix (EDM): a symmetric matrix A that can be written as

$$A_{ij} = \|y_i - y_j\|_2^2, \quad i, j = 1, \ldots, n$$

for some vectors $y_1, \ldots, y_n$

- we call the matrix Y with rows y_i^T a *realization* of A:

$$\begin{array}{rcl} A_{ij} & = & y_i^T y_i - 2 y_i^T y_j + y_j^T y_j \\ & = & (YY^T)_{ii} - 2(YY^T)_{ij} + (YY^T)_{jj} \end{array}$$

- Y is not unique: if Y is a realization of A, then

$$\begin{array}{rcl} \tilde{Y} & = & YQ^T + \mathbf{1}a^T \\ & = & \left[\begin{array}{cccc} Qy_1 + a & Qy_2 + a & \cdots & Qy_n + a \end{array} \right]^T \end{array}$$

  is a realization, for any orthogonal Q and any a

Schoenberg characterization

a symmetric $n \times n$ matrix A is a Euclidean matrix if and only if

$$\mathbf{diag}(A) = 0, \qquad P^T A P \preceq 0$$

where P is any matrix whose columns span the orthogonal complement of $\mathbf{1}$

(Schoenberg 1935, 1938)

- second condition means that $x^T A x \leq 0$ if $\mathbf{1}^T x = 0$
- a realization A can be computed from a factorization $P^T A P = -YY^T$
- a useful choice is

$$P = I - e_k \mathbf{1}^T = \begin{array}{cc} & \text{column } k \\ \left[\begin{array}{ccc} I & 0 & 0 \\ -\mathbf{1}^T & 0 & -\mathbf{1}^T \\ 0 & 0 & I \end{array}\right] & \begin{array}{c} \\ \text{row } k \\ \\ \end{array} \end{array}$$

  factorizing $P^T A P = -YY^T$ gives a realization that satisfies $y_k = 0$

- we will use the notation $\dim(A) = \operatorname{rank}(P^T A P)$
- if $\dim(A) = m$ there is a realization in $\mathbf{R}^m$ (with points y_i in $\mathbf{R}^m$)

Euclidean distance matrix completion

EDM completion problem: given $A \in \mathbf{S}^n_E$ find an EDM X such that

$$A = \Pi_E(X)$$

(or determine that no such completion exists)

- this is an SDP feasibility problem: find X such that

$$A = \Pi_E(X), \qquad P^T X P \preceq 0$$

  for any P whose columns span $\mathbf{1}^\perp$

- in many applications one is interested in the solution that minimizes

$$\dim(X) = \mathrm{rank}(P^T X P),$$

  to obtain a realization in the lowest-dimensional space

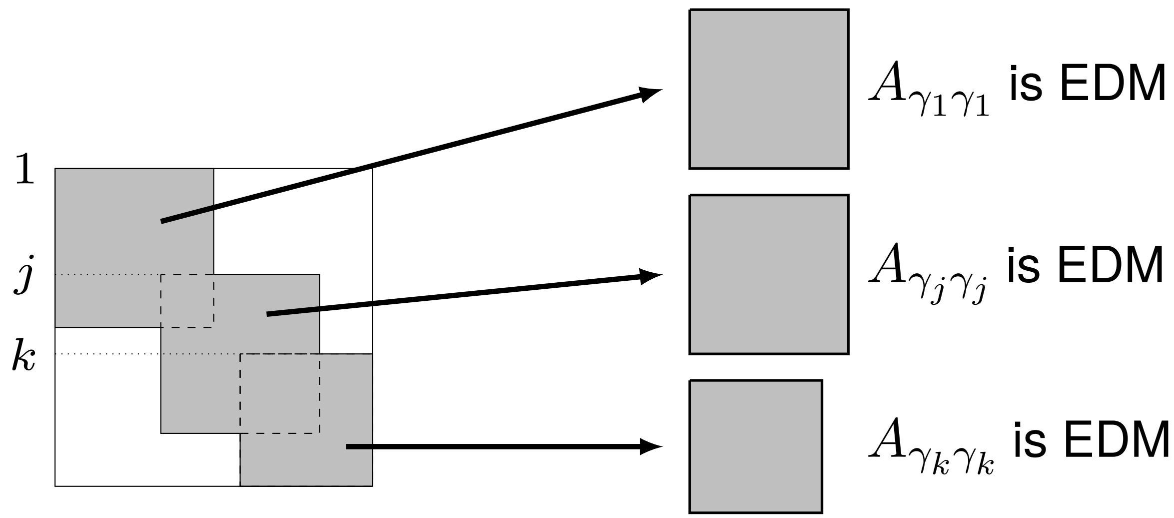
EDM completion for chordal sparsity pattern

Decomposition theorem (for chordal E)

$A \in \mathbf{S}^n_E$ has an EDM completion if and only if $A_{\gamma_i\gamma_i}$ is EDM for all $i \in V^{\mathrm{c}}$

(Bakonyi and Johnson 1995)

Example



EDM completable A

Minimum dimension completion: there exists a completion with

$$\dim(X) = \max_{i \in V^{\mathrm{c}}} \dim(A_{\gamma_i\gamma_i})$$

Minimum-dimension EDM completion for chordal patterns

we only consider the simple pattern with two overlapping diagonal blocks

$$
\begin{array}{cc}
 & \begin{array}{ccc} p & q & r \end{array} \\
A = & \left[\begin{array}{ccc} A_{11} & A_{12} & 0 \\ A_{21} & A_{22} & A_{23} \\ 0 & A_{32} & A_{33} \end{array}\right] \begin{array}{c} p \\ q \\ r \end{array}
\end{array}
$$

- from the decomposition theorem, a solution exists if

$$
C_1 = \left[\begin{array}{cc} A_{11} & A_{12} \\ A_{21} & A_{22} \end{array}\right] \text{ is an EDM,} \qquad C_2 = \left[\begin{array}{cc} A_{22} & A_{23} \\ A_{32} & A_{33} \end{array}\right] \text{ is an EDM}
$$

- we compute a completion with dimension $m = \max\{\dim(C_1), \dim(C_2)\}$

for general E, use the 2-block algorithm and a recursion on the clique tree in inverse topological order

Two-block EDM completion

- define matrices

$$P_1 = I - e_{p+1}\mathbf{1}^T \in \mathbf{R}^{(p+q)\times(p+q)}, \qquad P_2 = I - e_1\mathbf{1}^T \in \mathbf{R}^{(q+r)\times(q+r)}$$

and compute matrices U, V, $\tilde{V}$, W with column dimension m such that

$$\begin{aligned} P_1^T \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} P_1 &= -\begin{bmatrix} U \\ V \end{bmatrix} \begin{bmatrix} U \\ V \end{bmatrix}^T \\ P_2^T \begin{bmatrix} A_{22} & A_{23} \\ A_{32} & A_{33} \end{bmatrix} P_2 &= -\begin{bmatrix} \tilde{V} \\ W \end{bmatrix} \begin{bmatrix} \tilde{V} \\ W \end{bmatrix}^T \end{aligned}$$

- since $VV^T = \tilde{V}\tilde{V}^T$ there exists an orthogonal Q such that $V = \tilde{V}Q$
- the matrix

$$Y = \begin{bmatrix} UQ^T \\ \tilde{V} \\ W \end{bmatrix}$$

is a realization of an EDM completion of A

References

- References can be found in the bibliography of

L. Vandenberghe, M. S. Andersen, *Chordal Graphs and Semidefinite Optimization*, Foundations and Trends in Optimization, 2015.

- The minimum rank completion algorithm is from

Y. Sun, *Decomposition Methods for Sparse Semidefinite Optimization*, Ph.D. Thesis, UCLA, 2015.

III. Applications in convex optimization

- nonsymmetric interior-point methods
- partial separability and decomposition
  - partial separability
  - first order methods
  - interior-point methods

Conic linear optimization

$$\begin{array}{llll} \text{primal:} & \text{minimize} & c^T x \\ & \text{subject to} & Ax = b \\ & & x \in \mathcal{C} \end{array} \qquad \begin{array}{llll} \text{dual:} & \text{maximize} & b^T y \\ & \text{subject to} & A^T y + s = c \\ & & s \in \mathcal{C}^* \end{array}$$

- $\mathcal{C}$ is a proper cone (convex, closed, pointed, with nonempty interior)
- $\mathcal{C}^* = \{z \mid z^T x \geq 0 \text{ for all } x \in \mathcal{C}\}$ is the dual cone

widely used in recent literature on convex optimization

- **Interior-point methods**

  a convenient format for extending interior-point methods from linear optimization to general convex optimization

- **Modeling**

  a small number of 'primitive' cones is sufficient to model most convex constraints encountered in practice

Symmetric cones

most current solvers and modeling systems use three types of cones

- nonnegative orthant
- second-order cone
- positive semidefinite cone

these cones are not only self-dual but symmetric (self-scaled)

- symmetry is exploited in primal-dual symmetric interior-point methods
- large gaps in (linear algebra) complexity between the three cones

  (see the examples on page 5–6)

Sparse semidefinite optimization problem

Primal problem

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_iX) = b_i, \quad i = 1, \ldots, m \\ & X \succeq 0 \end{array}$$

Dual problem

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & \sum_{i=1}^{m} y_i A_i + S = C \\ & S \succeq 0 \end{array}$$

Aggregate sparsity pattern

- the union of the patterns of C, $A_1, \ldots, A_m$
- feasible X is usually dense, even for problems with aggregate sparsity
- feasible S is sparse with sparsity pattern E

Equivalent nonsymmetric conic LPs

Primal problem

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_iX) = b_i, \quad i = 1, \ldots, m \\ & X \in \mathcal{C} \end{array}$$

Dual problem

$$\begin{array}{ll} \text{maximize} & b^T y \\ \text{subject to} & \sum_{i=1}^{m} y_i A_i + S = C \\ & S \in \mathcal{C}^* \end{array}$$

- variables X and S are sparse matrices in $\mathbf{S}_E^n$
- $\mathcal{C} = \Pi_E(\mathbf{S}_+^n)$ is cone of PSD completable matrices with sparsity pattern E
- $\mathcal{C}^* = \mathbf{S}_+^n \cap \mathbf{S}_E^n$ is cone of PSD matrices with sparsity pattern E
- $\mathcal{C}$ is not self-dual; no symmetric interior-point methods

Nonsymmetric interior-point methods

$$
\begin{array}{ll}
\text{minimize} & \mathbf{tr}(CX) \\
\text{subject to} & \mathbf{tr}(A_i X) = b_i, \quad i = 1, \ldots, m \\
& X \in \Pi_E(\mathbf{S}^n_+)
\end{array}
$$

- can be solved by nonsymmetric primal or dual barrier methods
- logarithmic barriers for cone $\Pi_E(\mathbf{S}^n_+)$ and its dual cone $\mathbf{S}^n_+ \cap \mathbf{S}^n_E$:

$$
\phi_*(X) = \sup_S \left(-\,\mathbf{tr}(XS) + \log\det S\right), \qquad \phi(S) = -\log\det S
$$

- fast evaluation of barrier values and derivatives if pattern is chordal

(Fukuda et al. 2000, Burer 2003, Srijungtongsiri and Vavasis 2004, Andersen et al. 2010)

Primal path-following method

Central path: solution $X(\mu)$, $y(\mu)$, $S(\mu)$ of

$$\begin{array}{rcl} \mathbf{tr}(A_i X) & = & b_i, \quad i = 1, \ldots, m \\ \sum_{j=1}^{m} y_j A_j + S & = & C \\ \mu \nabla \phi_*(X) + S & = & 0 \end{array}$$

Search direction at iterate X, y, S: solve linearized central path equations

$$\begin{array}{rcl} \mathbf{tr}(A_i \Delta X) & = & r_i, \quad i = 1, \ldots, m \\ \sum_{i=1}^{m} \Delta y_i A_i + \Delta S & = & C \\ \mu \nabla^2 \phi_*(X)[\Delta X] + \Delta S & = & -\mu \nabla \phi_*(X) - S \end{array}$$

Dual path-following method

Central path: an equivalent set of equations is

$$\begin{aligned}
\mathbf{tr}(A_i X) &= b_i, \quad i = 1, \ldots, m \\
\sum_{j=1}^{m} y_j A_j + S &= C \\
X + \mu \nabla \phi(S) &= 0
\end{aligned}$$

Search direction at iterate X, y, S: solve linearized central path equations

$$\begin{aligned}
\mathbf{tr}(A_i \Delta X) &= r_i, \quad i = 1, \ldots, m \\
\sum_{i=1}^{m} \Delta y_i A_i + \Delta S &= C \\
\Delta X + \mu \nabla^2 \phi(S)[\Delta S] &= -\mu \nabla \phi(S) - X
\end{aligned}$$

Computing search directions

eliminating ΔX, ΔS from linearized equation gives

$$H\Delta y = g$$

- in a primal method H_{ij} is the inner product of A_i and $\nabla^2\phi^*(X)[A_j]$:

$$H_{ij} = \mathbf{tr}(A_i \nabla^2 \phi^*(X)[A_j])$$

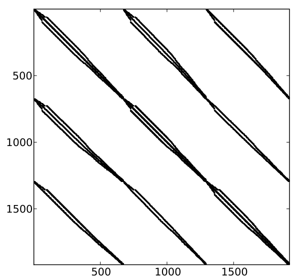
- in a dual method H_{ij} is the inner product of A_i and $\nabla^2\phi(S)[A_j]$:

$$H_{ij} = \mathbf{tr}(A_i \nabla^2 \phi(S)[A_j])$$

- the algorithms from lecture 2 can be used to evaluate gradient and Hessians
- the system $H\Delta y = g$ is solved via dense Cholesky or QR factorization

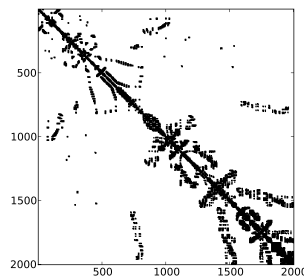
Sparsity patterns

- sparsity patterns from University of Florida Sparse Matrix Collection
- $m = 200$ constraints
- random data with 0.05% nonzeros in A_i relative to $|E|$



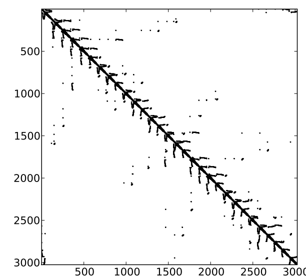
rs228

$n = 1{,}919$



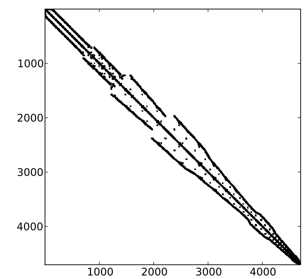
rs35

$n = 2{,}003$



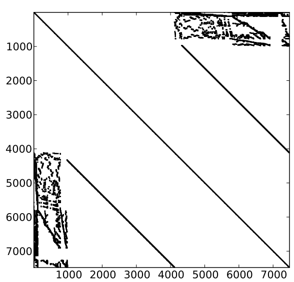
rs200

$n = 3{,}025$



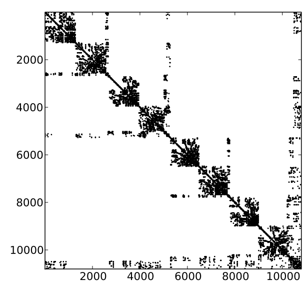
rs365

$n = 4{,}704$



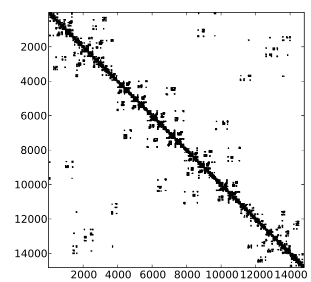
rs1555

$n = 7{,}479$



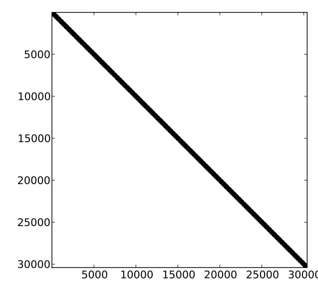
rs828

$n = 10{,}800$



rs1184

$n = 14{,}822$



rs1288

$n = 30{,}401$

Results

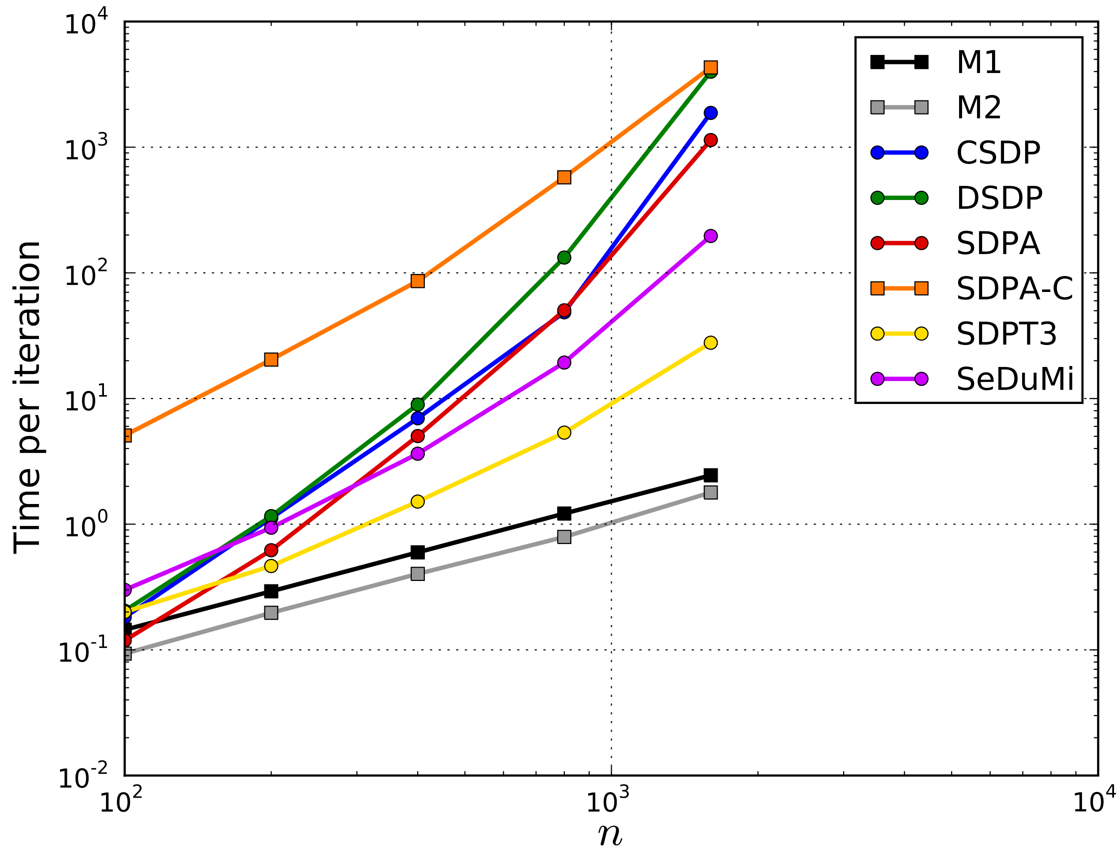
n	DSDP	SDPA	SDPA-C	SDPT3	SeDuMi	SMCP
1919	1.4	30.7	5.7	10.7	511.2	2.3
2003	4.0	34.4	41.5	13.0	521.1	15.3
3025	2.9	128.3	6.0	33.0	1856.9	2.2
4704	15.2	407.0	58.8	99.6	4347.0	18.6

n	DSDP	SDPA-C	SMCP
7479	22.1	23.1	9.5
10800	482.1	1812.8	311.2
14822	791.0	2925.4	463.8
30401	mem	2070.2	320.4

- average time per iteration for different solvers
- SMCP uses nonsymmetric matrix cone approach (Andersen et al. 2010)
- code and more benchmarks at `github.com/cvxopt/smcp`

Band pattern

SDPs of order n with bandwidth 11 and $m = 100$ equality constraints

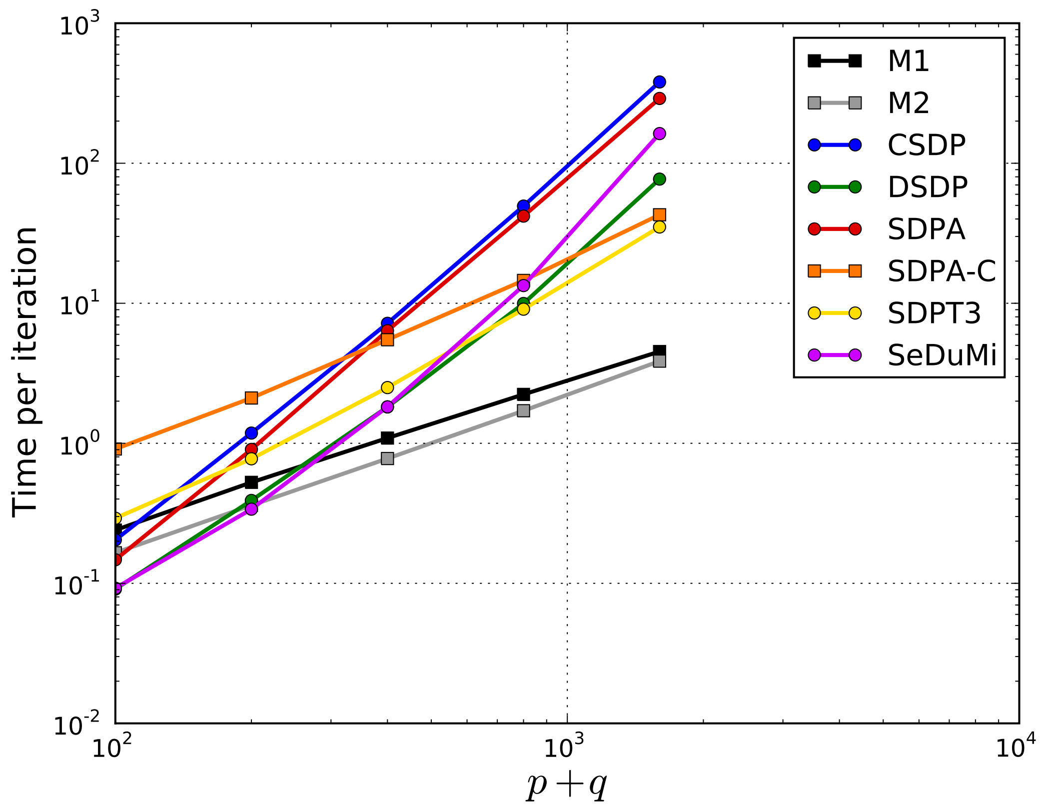


nonsymmetric solver SMCP (two variants M1, M2): complexity is *linear* in n

(Andersen et al. 2010)

Arrow pattern

- matrix norm minimization of page 6
- matrices of size $p \times q$ with $q = 10$ with $m = 100$ variables



nonsymmetric solver SMCP (M1, M2): complexity *linear* in p

III. Applications in convex optimization

- nonsymmetric interior-point methods
- **partial separability and decomposition**
  - partial separability
  - first order methods
  - interior-point methods

Partial separability

Partially separable function (Griewank and Toint 1982)

$$f(x) = \sum_{k=1}^{l} f_k(P_{\beta_k} x)$$

x is an n-vector; $\beta_1, \ldots, \beta_l$ are (small) overlapping index sets in $\{1, 2, \ldots, n\}$

Example:

$$f(x) = f_1(x_1, x_4, x_5) + f_2(x_1, x_3) + f_3(x_2, x_3) + f_4(x_2, x_4)$$

Partially separable set

$$C = \{x \in \mathbf{R}^n \mid x_{\beta_k} \in C_k, \quad k = 1, \ldots, l\}$$

the indicator function is a partially separable function

Interaction graph

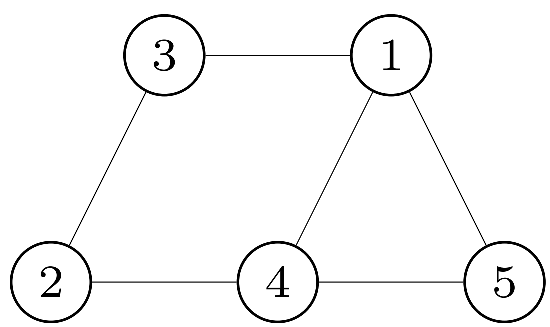
- vertices $V = \{1, 2, \ldots, n\}$,

$$\{i, j\} \in E \quad \Longleftrightarrow \quad i, j \in \beta_k \text{ for some } k$$

- if $\{i, j\} \notin E$, then f is separable in x_i and x_j if other variables are fixed:

$$f(x + se_i + te_j) = f(x + se_i) + f(x + te_j) - f(x) \qquad \forall x \in \mathbf{R}^n, s, t \in \mathbf{R}$$

Example: $f(x) = f_1(x_1, x_4, x_5) + f_2(x_1, x_3) + f_3(x_2, x_3) + f_4(x_2, x_4)$



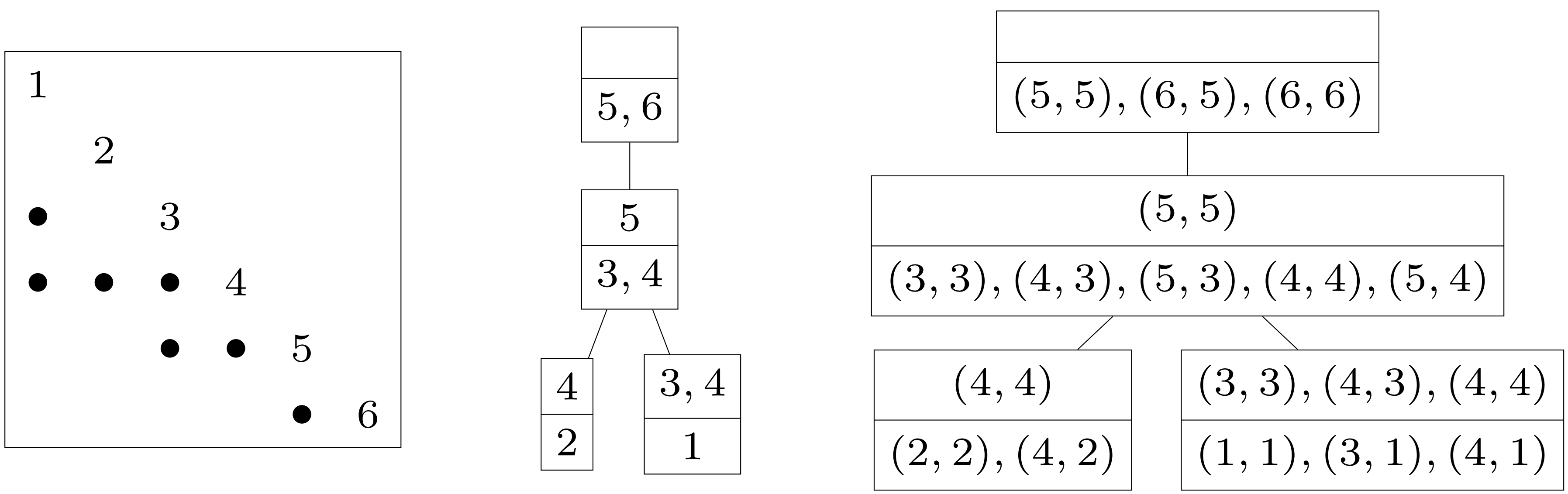
Example: PSD completable cone with chordal pattern

- for chordal E, the cone $\Pi_E(\mathbf{S}_+^n)$ is partially separable (see page 104)

$$\Pi_E(\mathbf{S}_+^n) = \{X \in \mathbf{S}_E^n \mid X_{\gamma_i \gamma_i} \succeq 0 \ \text{ for all cliques } \gamma_i\}$$

- the interaction graph is chordal

Example: chordal sparsity pattern, clique tree, clique tree of interaction graph



Partially separable convex optimization

$$\text{minimize} \quad f(x) = \sum_{k=1}^{l} f_k(P_{\beta_k} x)$$

Equivalent problem

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{l} f_k(\tilde{x}_k) \\ \text{subject to} & \tilde{x} = Px \end{array}$$

- we introduced 'splitting' variables $\tilde{x}_k$ to make cost function separable
- P, $\tilde{x}$ are stacked matrix and vector

$$P = \begin{bmatrix} P_{\beta_1} \\ \vdots \\ P_{\beta_l} \end{bmatrix}, \qquad \tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_l \end{bmatrix},$$

- $P^T P$ is diagonal ($(P^T P)_{ii}$ is the number of sets β_k that contain index i)

Decomposition via first-order methods

Reformulated problem and its its dual (f_k^* is conjugate function of f_k)

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{l} f_k(\tilde{x}_k) \\ \text{subject to} & \tilde{x} \in \operatorname{range}(P) \end{array} \qquad \begin{array}{ll} \text{maximize} & -\sum_{k=1}^{l} f_k^*(\tilde{s}_k) \\ \text{subject to} & \tilde{s} \in \operatorname{nullspace}(P^T) \end{array}$$

- cost functions are separable
- diagonal property of $P^T P$ makes projections on range inexpensive

Algorithms: many algorithms can exploit these properties, for example

- Douglas-Rachford (DR) splitting of the primal
- alternating direction method of multipliers (ADMM)

Example: sparse nearest matrix problems

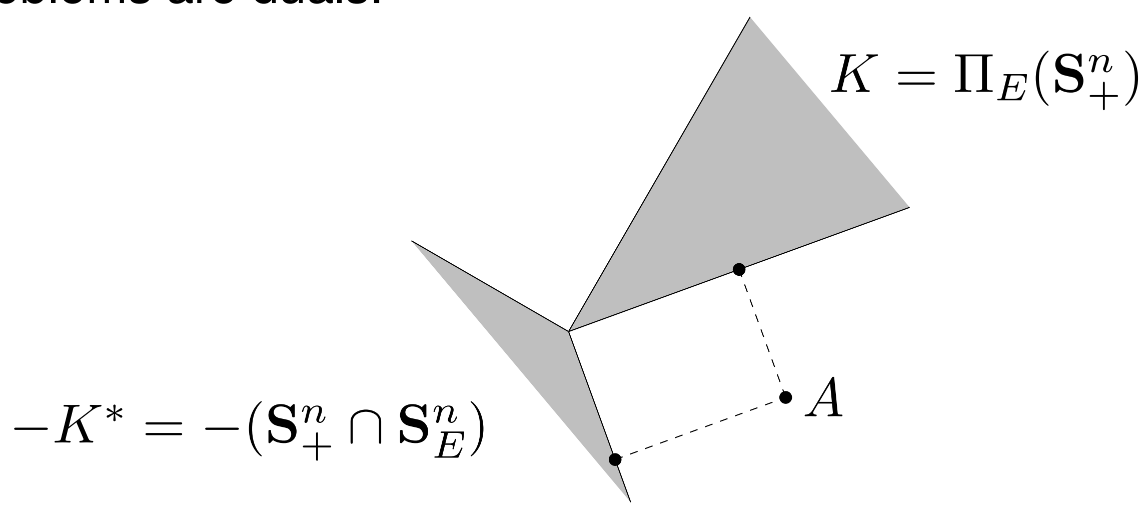
- find nearest sparse PSD-completable matrix with given sparsity pattern

$$\begin{array}{ll}\text{minimize} & \|X - A\|_F^2 \\ \text{subject to} & X \in \Pi_E(\mathbf{S}^n_+)\end{array}$$

- find nearest sparse PSD matrix with given sparsity pattern

$$\begin{array}{ll}\text{minimize} & \|S + A\|_F^2 \\ \text{subject to} & S \in \mathbf{S}^n_+ \cap \mathbf{S}^n_E\end{array}$$

these two problems are duals:



Decomposition methods

from the decomposition theorems (pages 82 and 104), the problems can be written

primal:

$$\begin{array}{ll} \text{minimize} & \|X - A\|_F^2 \\ \text{subject to} & X_{\gamma_i\gamma_i} \succeq 0 \text{ for all cliques } \gamma_i \end{array}$$

dual:

$$\begin{array}{ll} \text{minimize} & \|A + \sum\limits_{i \in V^{\mathrm{c}}} P_{\gamma_i}^T H_i P_{\gamma_i}\|_F^2 \\ \text{subject to} & H_i \succeq 0 \text{ for all } i \in V^{\mathrm{c}} \end{array}$$

Algorithms

- Dykstra's algorithm (dual block coordinate ascent)
- (fast) dual projected gradient algorithm (FISTA)
- Douglas-Rachford splitting, ADMM

sequence of projections on PSD cones of order $|\gamma_i|$ (eigenvalue decomposition)

Results

matrices from University of Florida sparse matrix collection

n	density	#cliques	avg. clique size	max. clique
20141	2.80e-3	1098	35.7	168
38434	1.25e-3	2365	28.1	188
57975	9.04e-4	8875	14.9	132
79841	9.71e-4	4247	44.4	337
114599	2.02e-4	7035	18.9	58

	total runtime (sec)			time/iteration (sec)		
n	FISTA	Dykstra	DR	FISTA	Dykstra	DR
20141	2.5e2	3.9e1	3.8e1	1.0	1.6	1.5
38434	4.7e2	4.7e1	6.2e1	2.1	1.9	2.5
57975	> 4hr	1.4e2	1.1e3	3.5	5.7	6.4
79841	2.4e3	3.0e2	2.4e2	6.3	7.6	9.7
114599	5.3e2	5.5e1	1.0e2	2.6	2.2	4.0

(Sun and Vandenberghe 2015)

Conic optimization with partially separable cones

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \in \mathcal{C} \end{array}$$

- assume $\mathcal{C}$ is partially separable:

$$\mathcal{C} = \{x \in \mathbf{R}^n \mid P_{\beta_k} x \in \mathcal{C}_k,\ k = 1, \ldots, l\}$$

- most important application is sparse semidefinite programming

  ($\mathcal{C}$ is vectorized PSD completable cone)

- bottleneck in interior-point methods is Schur complement equation

$$AH^{-1}A^T \Delta y = r$$

  (in a primal barrier method, H is the Hessian of the barrier for $\mathcal{C}$)

- coefficient of Schur complement equation is often dense, even for sparse A

Reformulation

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & P_{\beta_k} x \in \mathcal{C}_k, \quad k = 1, \ldots, l \end{array}$$

- introduce l splitting variables $\tilde{x}_k = P_{\gamma_k} x$ and add consistency constraints

$$\tilde{x} \in \mathrm{range}(P) \quad \text{where } \tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \vdots \\ \tilde{x}_l \end{bmatrix}, \; P = \begin{bmatrix} P_1 \\ \vdots \\ P_l \end{bmatrix}$$

- choose $\tilde{c}$, $\tilde{A}$ such that $\tilde{A}P = A$ and $\tilde{c}^T P = c^T$

Converted problem

$$\begin{array}{ll} \text{minimize} & \tilde{c}^T \tilde{x} \\ \text{subject to} & \tilde{A}\tilde{x} = b \\ & \tilde{x} \in \mathcal{C}_1 \times \cdots \times \mathcal{C}_l \\ & \tilde{x} \in \mathrm{range}(P) \end{array}$$

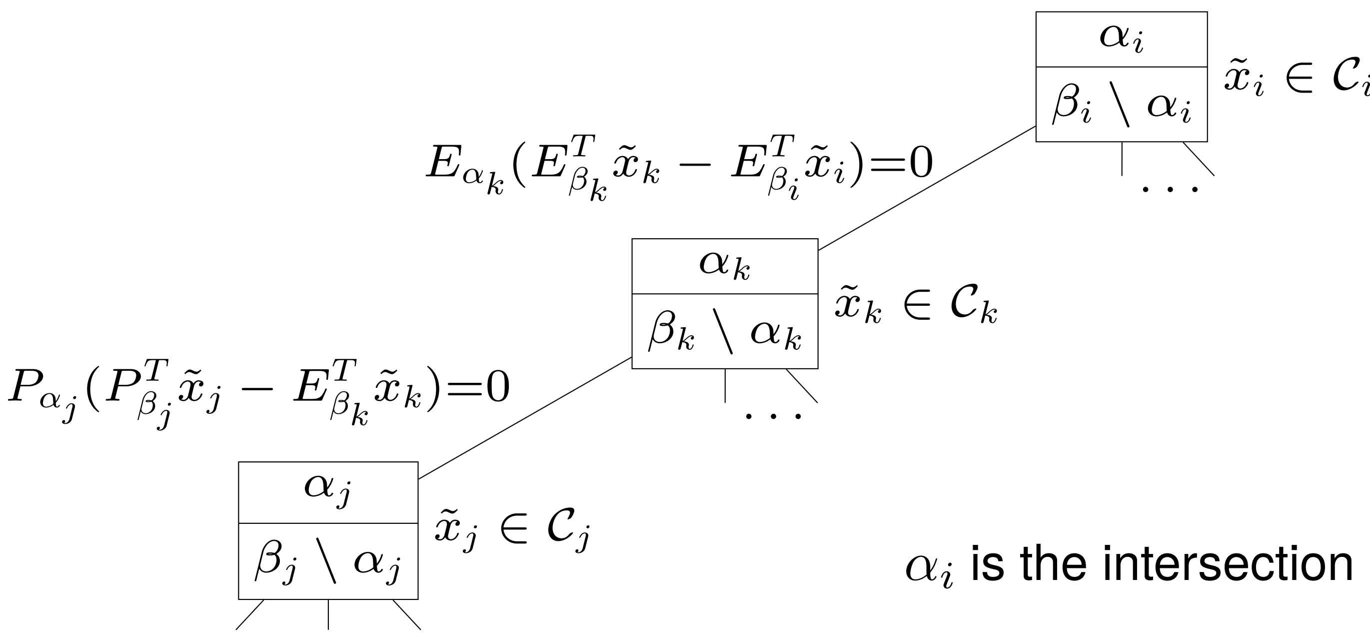
Chordal structure in interaction graph

suppose the interaction graph is chordal, and the sets β_k are cliques

- the cliques β_k that contain a given index j form a subtree of the clique tree
- therefore the consistency constraint $\tilde{x} \in \mathrm{range}(P)$ is equivalent to

$$P_{\alpha_j}(P_{\beta_k}^T \tilde{x}_k - P_{\beta_j}^T \tilde{x}_j) = 0$$

  for each vertex j and its parent k in a clique tree



α_i is the intersection of β_i and its parent

Schur complement system of converted problem

$$\begin{array}{ll} \text{minimize} & \tilde{c}^T \tilde{x} \\ \text{subject to} & \tilde{A}\tilde{x} = b \\ & \tilde{x} \in \mathcal{C}_1 \times \cdots \times \mathcal{C}_l \\ & B\tilde{x} = 0 \quad \text{(consistency eqs.)} \end{array}$$

- Schur complement equation in interior-point method

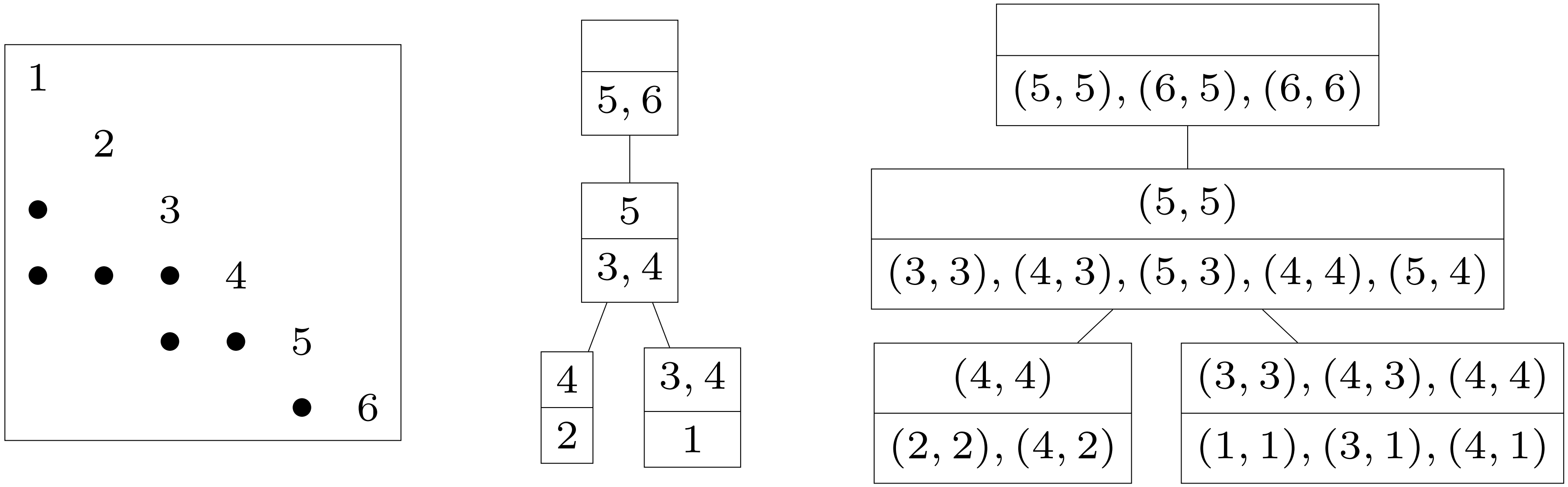
$$\left[\begin{array}{cc} \tilde{A}H^{-1}\tilde{A}^T & \tilde{A}H^{-1}B^T \\ BH^{-1}\tilde{A}^T & BH^{-1}B^T \end{array}\right] \left[\begin{array}{c} \Delta y \\ \Delta u \end{array}\right] = \left[\begin{array}{c} r_1 \\ r_2 \end{array}\right]$$

- H is block-diagonal (in primal barrier method, the Hessian of $C_1 \times \cdots \times C_k$)
- larger than Schur complement system before conversion
- however 1,1 block is often sparse

for semidefinite optimization, this is known as the ‘clique-tree conversion’ method

(Fukuda *et al.* 2000, Kim *et al.* 2011)

Example



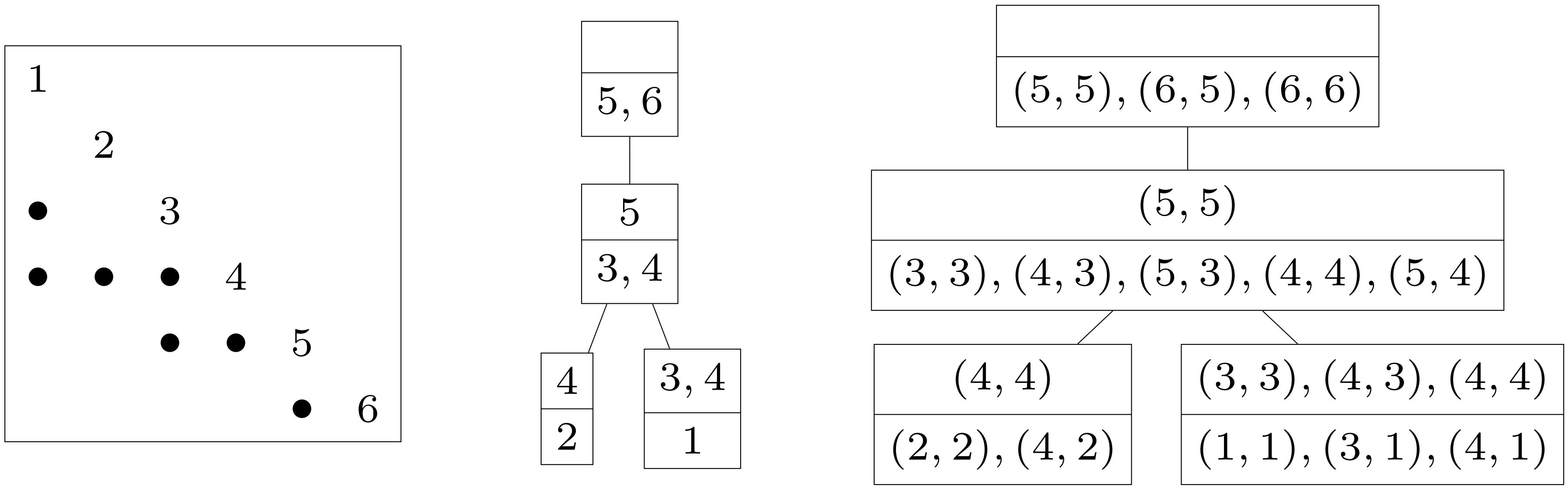
a 6×6 matrix X with this pattern is positive semidefinite if and only if the matrices

$$X_{\gamma_1\gamma_1} = \begin{bmatrix} X_{11} & X_{13} & X_{14} \\ X_{31} & X_{33} & X_{34} \\ X_{41} & X_{43} & X_{44} \end{bmatrix}, \qquad X_{\gamma_2\gamma_2} = \begin{bmatrix} X_{22} & X_{24} \\ X_{42} & X_{44} \end{bmatrix},$$

$$X_{\gamma_3\gamma_3} = \begin{bmatrix} X_{33} & X_{34} & X_{35} \\ X_{43} & X_{44} & X_{45} \\ X_{53} & X_{54} & X_{55} \end{bmatrix}, \qquad X_{\gamma_4\gamma_4} = \begin{bmatrix} X_{55} & X_{56} \\ X_{65} & X_{66} \end{bmatrix}$$

are positive semidefinite

Example



- define a splitting variable for each of the four submatrices

$$\tilde{X}_1 \in \mathbf{S}^4, \qquad \tilde{X}_2 \in \mathbf{S}^2, \qquad \tilde{X}_3 \in \mathbf{S}^4, \qquad \tilde{X}_4 \in \mathbf{S}^2$$

- add consistency constraints

$$\left[\begin{array}{cc} \tilde{X}_{1,22} & \tilde{X}_{1,23} \\ \tilde{X}_{1,32} & \tilde{X}_{1,33} \end{array}\right] = \left[\begin{array}{cc} \tilde{X}_{3,11} & \tilde{X}_{3,12} \\ \tilde{X}_{3,21} & \tilde{X}_{3,22} \end{array}\right], \quad \tilde{X}_{2,22} = \tilde{X}_{3,22}, \quad \tilde{X}_{3,33} = \tilde{X}_{4,11}$$

Summary: sparse semidefinite optimization

- sparse SDPs with chordal sparsity are partially separable

$$\begin{array}{ll} \text{minimize} & \mathbf{tr}(CX) \\ \text{subject to} & \mathbf{tr}(A_iX) = b_i, \quad i = 1, \ldots, m \\ & X_{\gamma_k\gamma_k} \succeq 0 \quad k = 1, \ldots, l \end{array}$$

- introducing splitting variables one can reformulate this as

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{l} \mathbf{tr}(\tilde{C}_k\tilde{X}_k) \\ \text{subject to} & \sum_{k=1}^{l} \mathbf{tr}(\tilde{A}_{ik}\tilde{X}_k) = b_i, \quad i = 1, \ldots, m \\ & \tilde{X}_k \succeq 0, \quad k = 1, \ldots, l \\ & \text{consistency constraints} \end{array}$$

- this was first proposed as a technique for speeding up interior-point methods
- also useful in combination with first-order splitting methods
  (Lu et al. 2007, Lam et al. 2011, Dall'Anese et al. 2013, Sun et al. 2014, ...)
- useful for distributed algorithms (Pakazad et al. 2014)