

Exploiting Fill-in and Fill-out in Gaussian-like Elimination Procedures on the Extended Jacobian Matrix

Andrew Lyons (Vanderbilt U.) / Uwe Naumann (RWTH Aachen)

Outline

1. Elementary Row/Column Operations in E'
2. Sparse Gaussian Elimination
3. Algorithms for exploiting fill

Motivation

Given function $\mathbf{y} = F(\mathbf{x})$, $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ as evaluation procedure

$$v_i = x_i, \quad i = 1, \dots, n$$

$$v_i = \varphi_i(v_h)_{h \prec i}, \quad i = n + 1, \dots, n + p + m$$

$$y_i = v_{n+p+i}, \quad i = 1, \dots, m$$

where $l \prec k \Leftrightarrow v_l$ is an argument of φ_k .

Want Jacobian matrix

$$F' = F'(\mathbf{x}) \equiv \left(\frac{\partial y_j}{\partial x_i}(\mathbf{x}) \right)_{\substack{j=1, \dots, m \\ i=1, \dots, n}} \in \mathbb{R}^{m \times n}$$

The Extended Jacobian

Extended system, from evaluation procedure:

$$0 = E(\mathbf{x}; \mathbf{v}) = (\varphi_i(u_h) - v_i)_{i=1, \dots, n+p+m}^{h \prec i}$$

Differentiate with respect to $\mathbf{v} = (v_i)_{i=n+p+m}$ to get:

$$E' = E'(\mathbf{x}; \mathbf{v}) \equiv (c_{i,j})_{i=1, \dots, (n+p+m)}^{j=1, \dots, (n+p+m)} - I \in \mathbb{R}^{(n+p+m) \times (n+p+m)}$$

Example

$$y_1 = x_1 * x_2 * x_3^2, \quad y_2 = x_1^2 * x_2 * x_3, \quad y_3 = x_1 * x_2^2 * x_3$$

$$v_1 = x_1; \quad v_2 = x_2; \quad v_3 = x_3$$

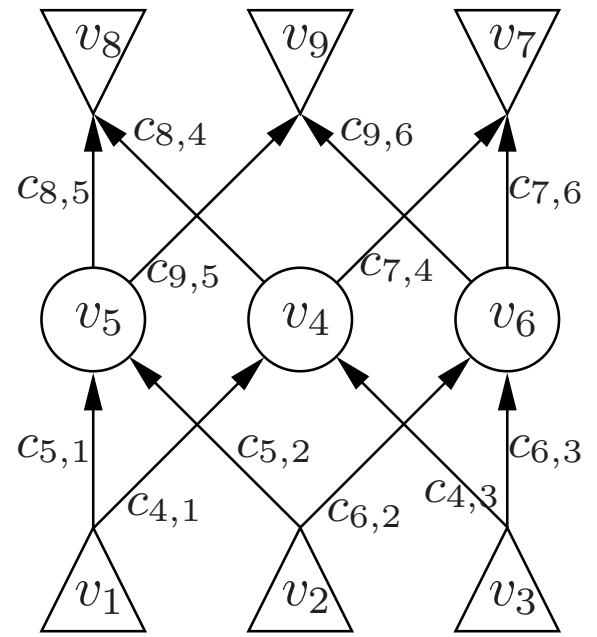
$$v_4 = v_1 * v_3; \quad v_5 = v_1 * v_2; \quad v_6 = v_2 * v_3$$

$$v_7 = v_4 * v_6; \quad v_8 = v_4 * v_5; \quad v_9 = v_5 * v_6$$

$$y_1 = v_7; \quad y_2 = v_8; \quad y_3 = v_9$$

Example continued

$$\begin{bmatrix}
 -1 & 0 & \dots & \dots & \dots & \dots & \dots & \dots & 0 \\
 0 & -1 & \ddots & & & & & & \vdots \\
 0 & 0 & -1 & \ddots & & & & & \vdots \\
 c_{4,1} & 0 & c_{4,3} & -1 & \ddots & & & & \vdots \\
 c_{5,1} & c_{5,2} & 0 & 0 & -1 & \ddots & & & \vdots \\
 0 & c_{6,2} & c_{6,3} & 0 & 0 & -1 & \ddots & & \vdots \\
 0 & 0 & 0 & c_{7,4} & 0 & c_{7,6} & -1 & \ddots & \vdots \\
 0 & 0 & 0 & c_{8,4} & c_{8,5} & 0 & 0 & -1 & 0 \\
 0 & 0 & 0 & 0 & c_{9,5} & c_{9,6} & 0 & 0 & -1
 \end{bmatrix}$$



Elementary Row and Column Operations

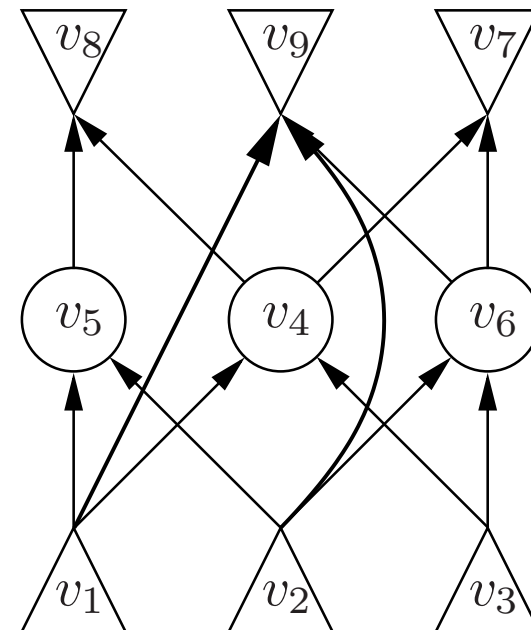
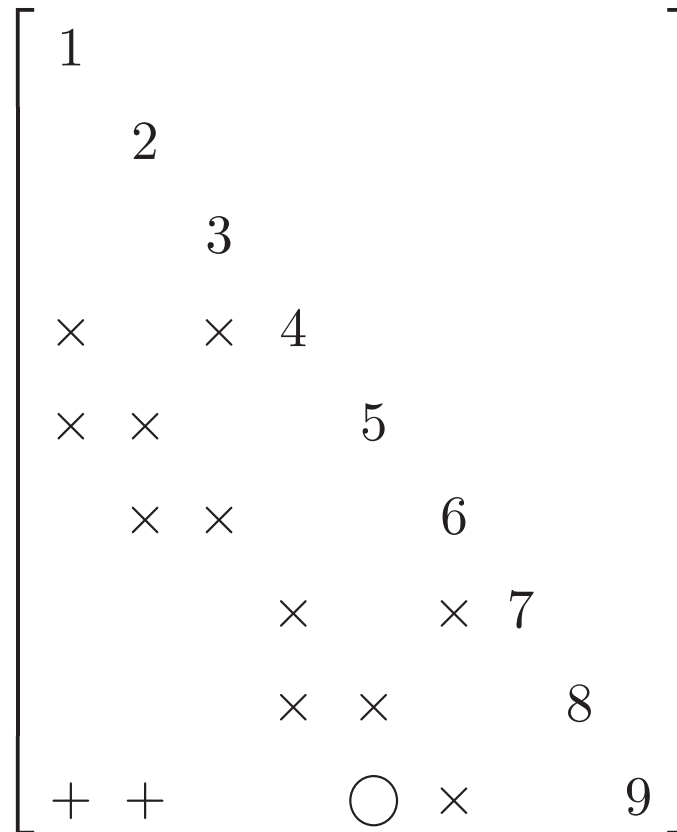
Correspond to back- and front-eliminations of edges in the Linearized computational graph:

$$V = (v_i)_{i=1, \dots, n+p+m}$$

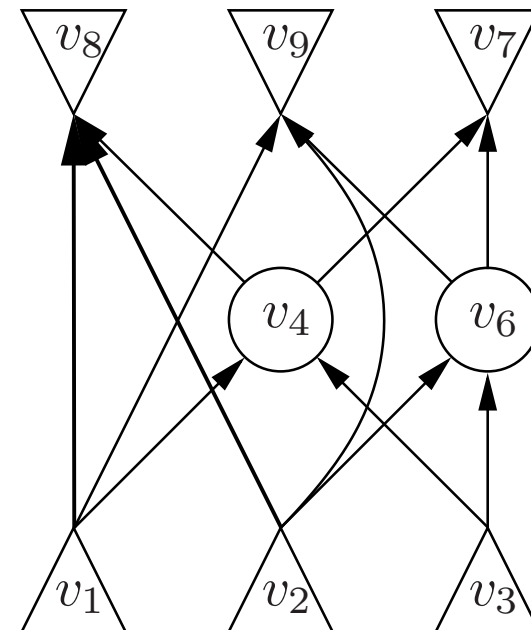
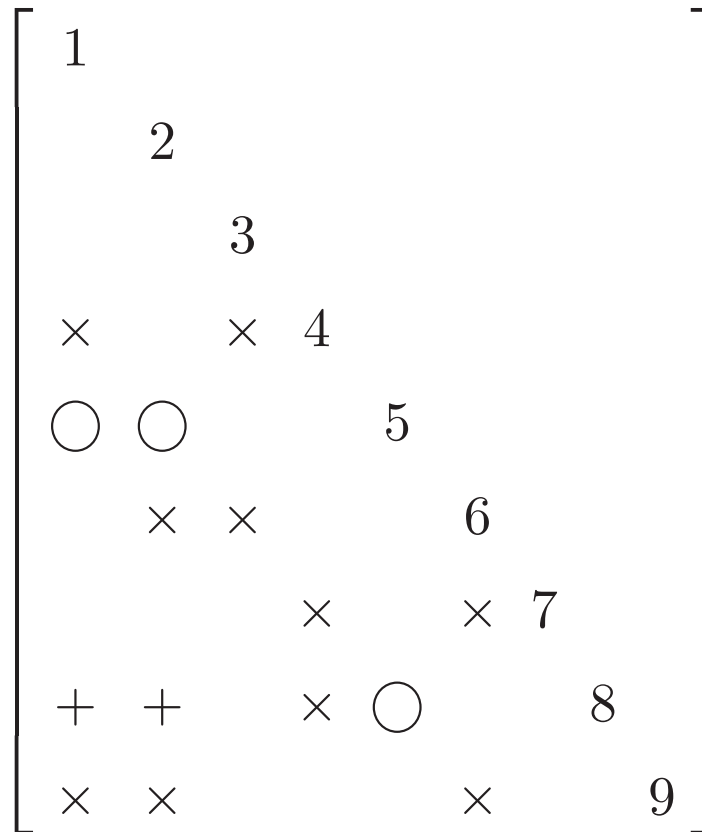
$$(v_i, v_j) \in E \Leftrightarrow i \prec j$$

- Rows contain in-edges, columns contain out-edges
- eliminating all in- or out-edges equates to *vertex elimination*
- not the most general elimination technique - see face elimination

Elementary Row and Column Operations



Elementary Row and Column Operations



Digression/Review – Sparse Gaussian Elimination

Part I - Symbolic Fill Prediction

Compressed row storage (CRS):

$$\begin{bmatrix} a & 0 & b & 0 \\ c & d & 0 & e \\ 0 & 0 & f & 0 \\ g & 0 & 0 & h \end{bmatrix}$$

α :

a	b	c	d	e	f	g	h
---	---	---	---	---	---	---	---

κ :

1	3	1	2	4	3	1	4
---	---	---	---	---	---	---	---

ρ :

1	3	6	7
---	---	---	---

Problem: Need to allocate memory for *fill-in* that occurs in $L + U$ during the elimination process

Solution: use a graph model to symbolically predict where such fill-in will occur, and allocate space for it in CRS scheme.

Sparse Gaussian Elimination

Part I - Symbolic Fill Prediction

- directed graph captures structure under symmetric row/column permutations
- Finding a pivot sequence that minimizes fill is NP-hard (Rose/Tarjan 78, corrected later by Gilbert)
- for jacobians, emphasis has been on ops: elimination of a vertex “costs”
num. predecessors * num. successors
- ops minimization for Jacobian accumulation is NP-hard (Naumann2005)
- there is a lot of theory behind “perfect elimination graphs” with respect to fill, but there are no ‘perfect elimination’ computational graphs with respect to ops

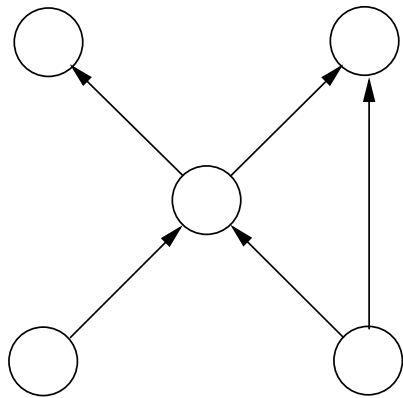
Sparse Gaussian Elimination

Part II - Numerical Phase

- Symbolic phase works for all matrices with same sparsity pattern.
- searching through indices is a large part of the cost?

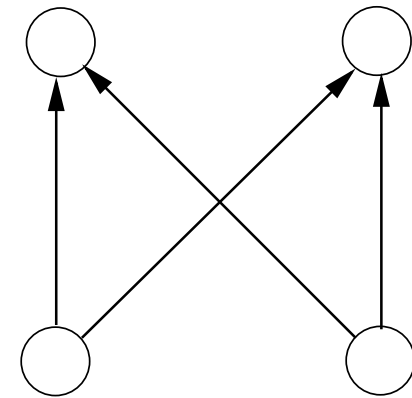
Sparse Gaussian Elimination and Jacobian Accumulation

They're **Similar**



→

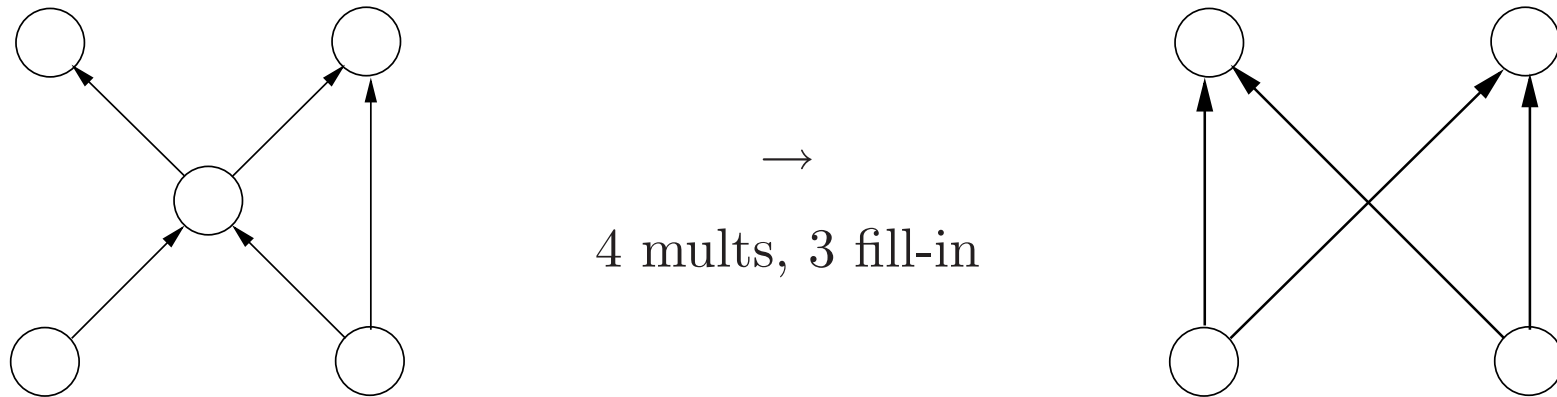
vertex elimination



- E' is analogous to A
- This relationship is hardly new, research draws from the large body of research in sparse linear systems
 - Griewank/Reese 91 - Markowitz heuristic
 - Pryce introduced a scheme for crout-doolittle $A = LU$ factorization.

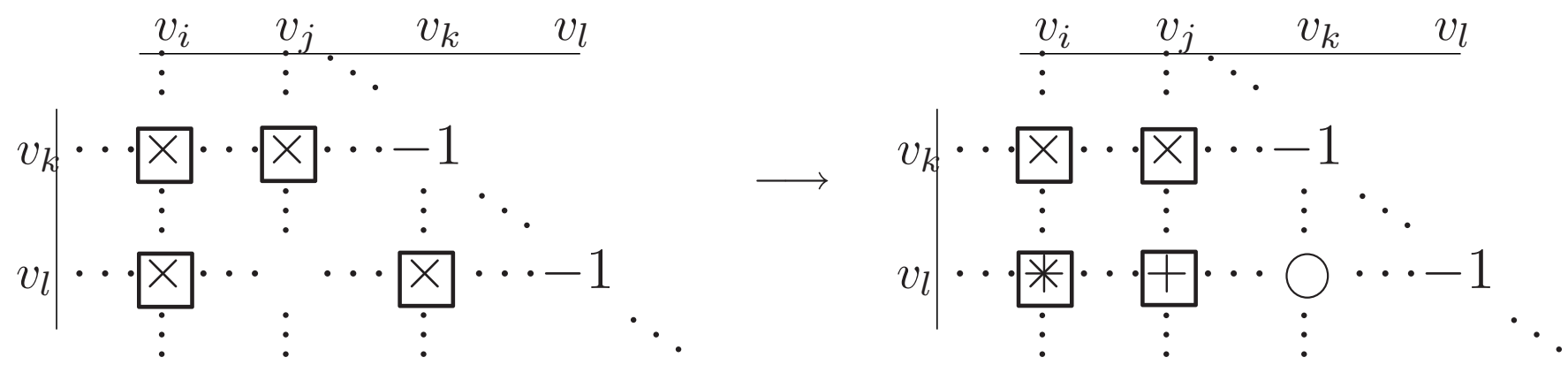
Sparse Gaussian Elimination and Jacobian Accumulation

They're **Different**



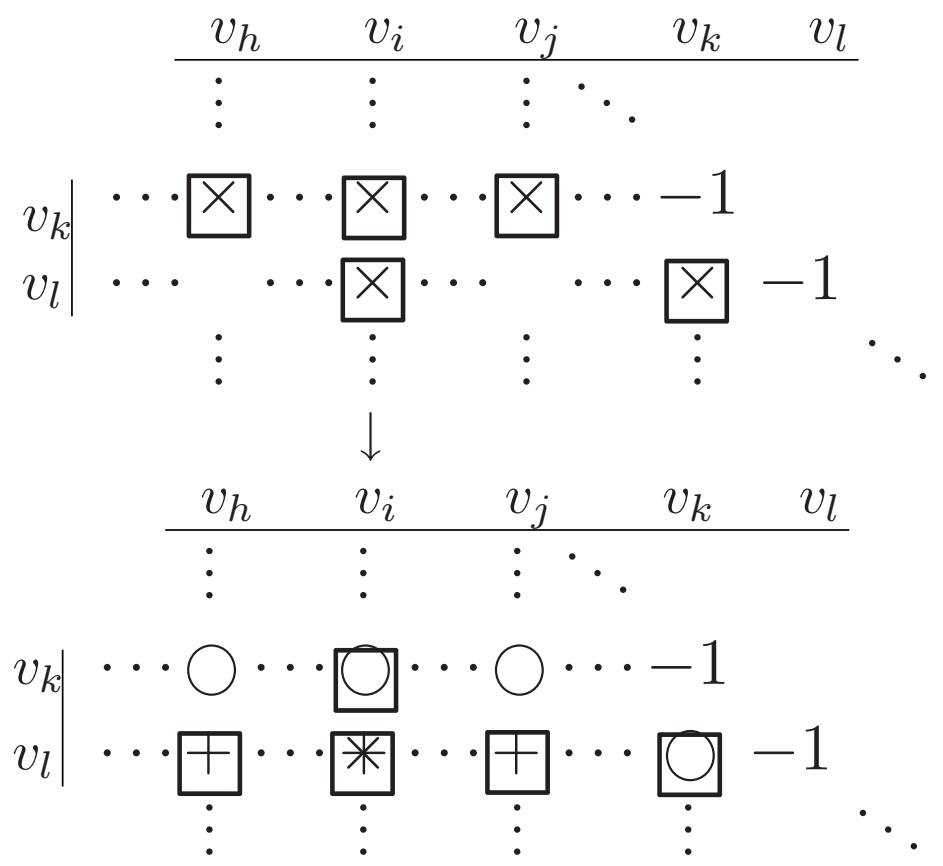
- lower triangular E' means the computational graph is acyclic \Rightarrow edge elimination sequences terminate
- don't eliminate all vertices (not a big deal)
- fill-out (potential for exploitation)

Technique 1



Back elimination of (k, l)

Technique 2

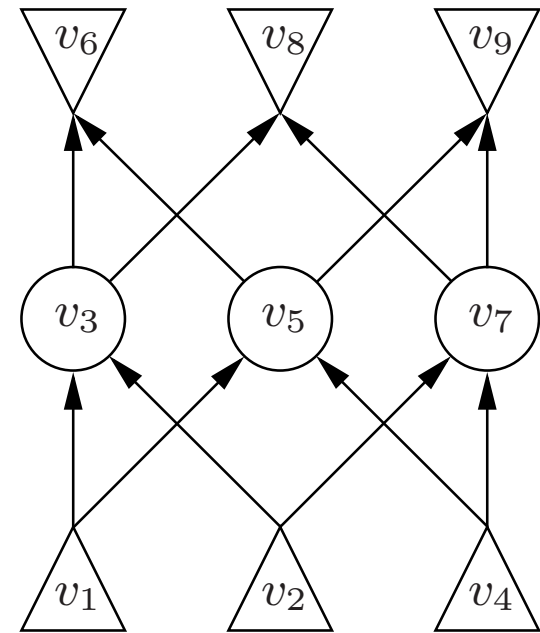
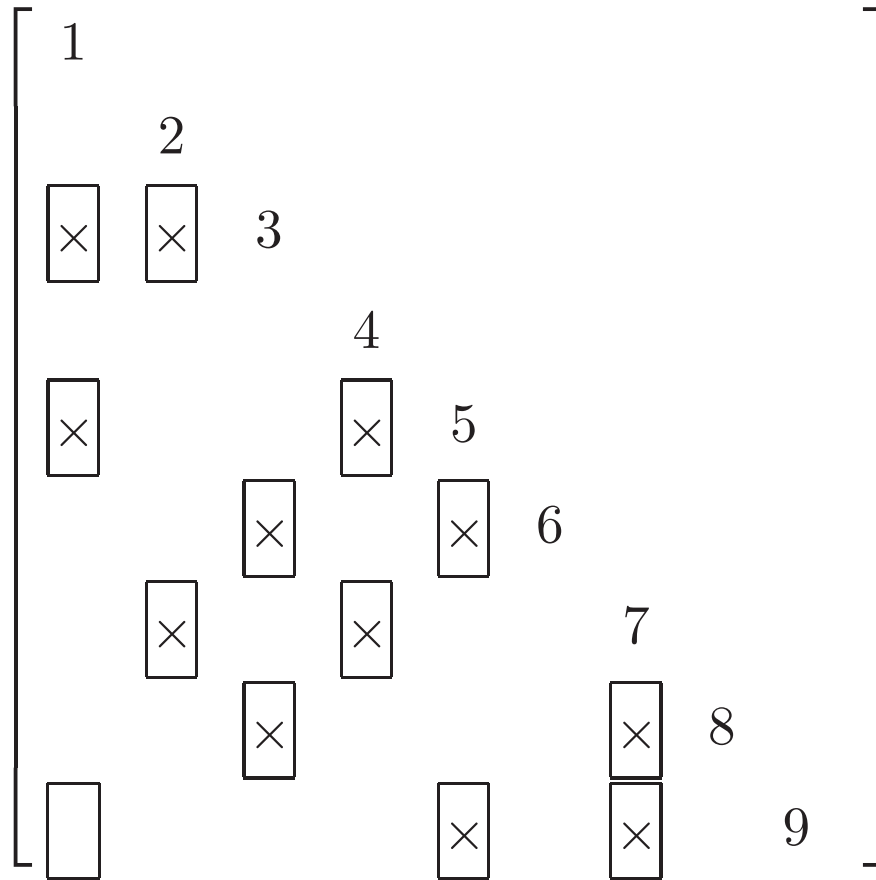


Maximum Immediate Successor Enumeration

We have extra freedom: \prec^* is a partial order. we can symmetrically permute E' in order to get more nonzero elements one off the diagonal.

Problem: Find a topological sort of G that maximizes the number of edges from v_i to v_{i+1}

Maximum Immediate Successor Enumeration



MISE is NP-complete

Reduction from covering by bicliques (CCBS):

- CCBS is NP-complete for bipartite graphs.
- Every “immediate successor” corresponds to exactly one biclique.

⇒ MISE is NP-complete

To do:

- support for general edge elimination sequences (heuristics?)
- speed up fill prediction