Elimination Methods on DAGs

Perspectives on Matrix Chaining

Uwe Naumann

Informatik 12:
Software and Tools for Computational Engineering (STCE)

RWTH Aachen University

Contents

Outline

Objective

- Introduction to vertex and edge elimination of directed acyclic graphs of [sparse] matrix chain products.

Learning Outcomes

- You will understand
  - evaluation of matrix [chain] products by vertex elimination
  - NP-completeness of the Vertex Elimination (VE) problem
  - edge elimination and NP-completeness of the Edge Elimination (EE) problem
  - suboptimality of VE and EE
- You will be able to
  - implement vertex and edge elimination

Outline

Consider the sparse matrix chain product

$$\begin{pmatrix} m^2_{0,0} & 0 & 0 \\ 0 & m^2_{1,1} & m^2_{1,2} \end{pmatrix} \begin{pmatrix} m^1_{0,0} & m^1_{0,1} & 0 \\ 0 & m^1_{1,1} & m^1_{1,2} \\ 0 & 0 & m^1_{2,2} \end{pmatrix} \begin{pmatrix} m^0_{0,0} & 0 \\ m^0_{1,0} & 0 \\ 0 & m^0_{2,1} \end{pmatrix}.$$

The search space of the corresponding SMCP instance consists of two configurations. Bracketing from the left yields

$$\begin{pmatrix} m^2_{0,0}m^1_{0,0} & m^2_{0,0}m^1_{0,1} & 0 \\ 0 & m^2_{1,1}m^1_{1,1} & m^2_{1,1}m^1_{1,2} + m^2_{1,2}m^1_{2,2} \end{pmatrix} \begin{pmatrix} m^0_{0,0} & 0 \\ m^0_{1,0} & 0 \\ 0 & m^0_{2,1} \end{pmatrix}$$

$$= \begin{pmatrix} m^2_{0,0}m^1_{0,0}m^0_{0,0} + m^2_{0,0}m^1_{0,1}m^0_{1,0} & 0 \\ m^2_{1,1}m^1_{1,1}m^0_{1,0} & (m^2_{1,1}m^1_{1,2} + m^2_{1,2}m^1_{2,2})m^0_{2,1} \end{pmatrix}$$

at the expense of 9 `fma` (fused multiply-add operations)[1].

[1] equivalently: number of scalar multiplications

Bracketing from the right yields

$$\begin{pmatrix} m^2_{0,0} & 0 & 0 \\ 0 & m^2_{1,1} & m^2_{1,2} \end{pmatrix} \begin{pmatrix} m^1_{0,0}m^0_{0,0} + m^1_{0,1}m^0_{1,0} & 0 \\ m^1_{1,1}m^0_{1,0} & m^1_{1,2}m^0_{2,1} \\ 0 & m^1_{2,2}m^0_{2,1} \end{pmatrix}$$

$$= \begin{pmatrix} m^2_{0,0}(m^1_{0,0}m^0_{0,0} + m^1_{0,1}m^0_{1,0}) & 0 \\ m^2_{1,1}m^1_{1,1}m^0_{1,0} & m^2_{1,1}m^1_{1,2}m^0_{2,1} + m^2_{1,2}m^1_{2,2}m^0_{2,1} \end{pmatrix}$$

also at the expense of 9 `fma`.

Note that the results of both bracketings are equal to

$$\begin{pmatrix} m^2_{0,0}(m^1_{0,0}m^0_{0,0} + m^1_{0,1}m^0_{1,0}) & 0 \\ m^2_{1,1}m^1_{1,1}m^0_{1,0} & (m^2_{1,1}m^1_{1,2} + m^2_{1,2}m^1_{2,2})m^0_{2,1} \end{pmatrix}$$

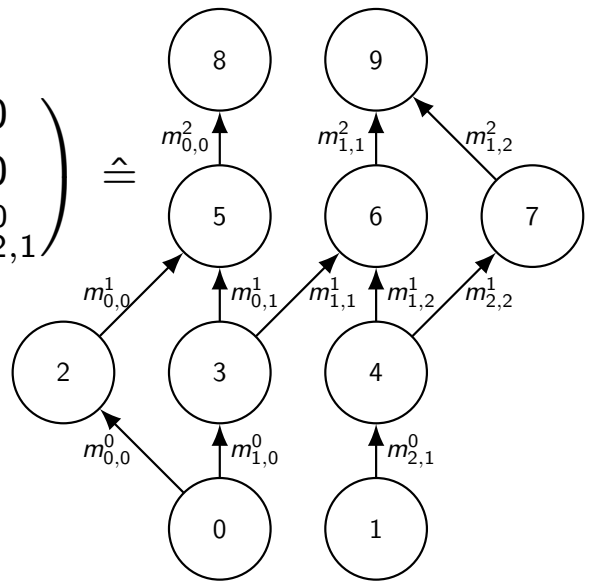
which involves only 8 `fma`.

Outline

Sparse matrices are equivalent to bipartite DAGs with partitions representing columns and rows. Edges represent and are labeled with the nonzero entries of the matrix.

[Sparse] matrix chain products become equivalent to concatenations of the corresponding bipartite DAGs $\rightarrow$ layered DAGs. For example,

$$\begin{pmatrix} m^2_{0,0} & 0 & 0 \\ 0 & m^2_{1,1} & m^2_{1,2} \end{pmatrix} \begin{pmatrix} m^1_{0,0} & m^1_{0,1} & 0 \\ 0 & m^1_{1,1} & m^1_{1,2} \\ 0 & 0 & m^1_{2,2} \end{pmatrix} \begin{pmatrix} m^0_{0,0} & 0 \\ m^0_{1,0} & 0 \\ 0 & m^0_{2,1} \end{pmatrix} \hat{=}$$



Let

$$M \equiv (m_{j,i}) = \prod_{\nu=p-1}^{0} M_\nu \in \boldsymbol{R}^{r_{p-1},c_0}$$

where $M_\nu = (m^\nu_{j,i}) \in \boldsymbol{R}^{r_\nu,c_\nu}$ induce a DAG $G = (V, E)$ with integer vertices $V = \{X, Z, Y\}$ comprising c_0 sources $X = \{0, \ldots, c_0 - 1\}$, $q = \sum_{\nu=0}^{p-2} r_\nu$ intermediate vertices $Z = \{c_0, \ldots, c_0 + q - 1\}$ and r_{p-1} sinks $Y = \{c_0 + q, \ldots, c_0 + q + r_{p-1} - 1\}$ such that $X \cap Z \cap Y$.

Edges $(k, l) \in E$ are labeled with corresponding matrix entries denoted as $\lambda_{k,l}$.

For all $i \in X$ and $j : j + c_0 + q \in Y$ holds the following chain rule:

$$m_{j,i} = \sum_{\pi : i \to j+c_0+q} \prod_{(k,l)\in\pi} \lambda_{k,l} \quad .$$

A [sparse] matrix product can be regarded as the elimination of all vertices in the intermediate layer.

According to the chain rule a vertex $j \in V$ is eliminated from a DAG $G = (V, E)$ by connecting its predecessors $i \in P_j$ with its successors $k \in S_j$ followed by removal of j together with all incident edges. New edges (i, k) are labeled with $m_{i,j} \cdot m_{j,k}$. If i and k are adjacent prior to the elimination of j, then its label is incremented with $m_{i,j} \cdot m_{j,k}$.

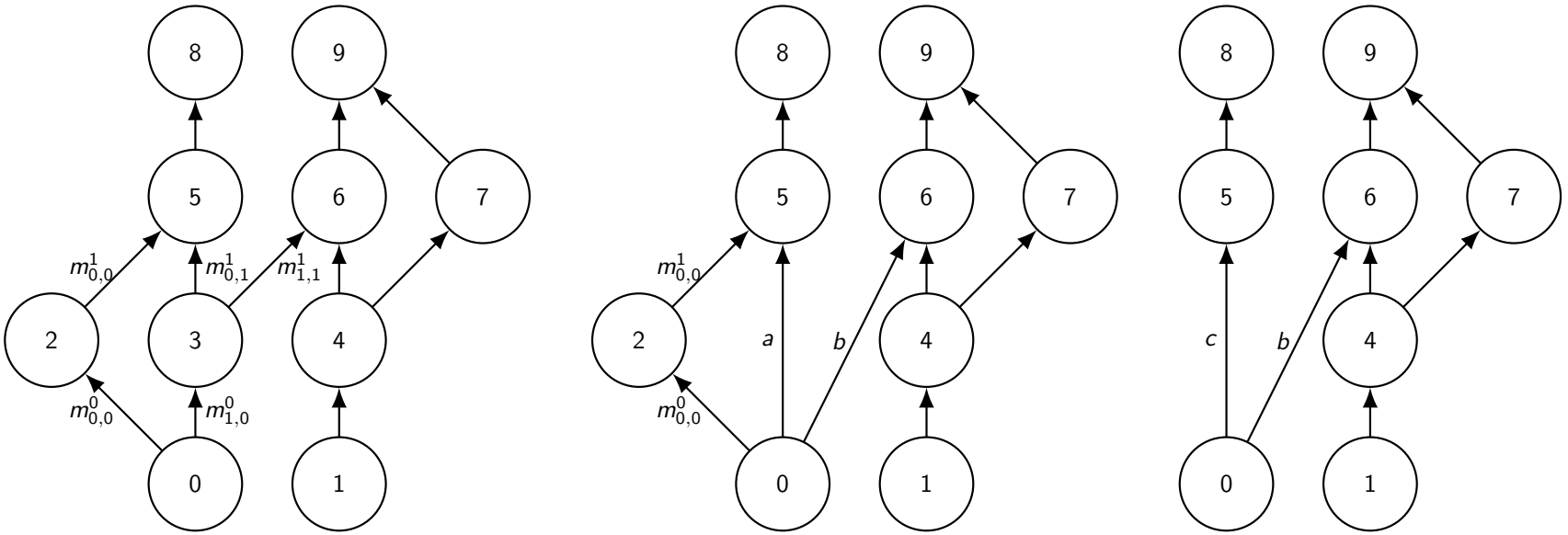
The number of `fma` operations involved in the elimination of j is equal to its [Markowitz] degree $|P_j||S_j|$.

The resulting DAG is bipartite. It represents the result of the [sparse] matrix product.

Different bracketings of [sparse] matrix chain product eliminate the intermediate layers in different orders yielding potentially distinct computational costs.

Vertex Elimination

Example



(a) Layered DAG showing edge labels relevant for consecutive elimination of vertices (b) 3 yielding $a = m^0_{1,0} \cdot m^1_{0,1}$ and $b = m^0_{1,0} \cdot m^1_{1,1}$, and (c) 2 yielding $c = a + m^0_{0,0} \cdot m^1_{0,0}$ etc.

The evaluation of the sample sparse matrix chain product with 8 `fma` corresponds, for example, to the vertex elimination sequence $[\{2,3\},\{5,6,7\},4]$ where the order of elimination within a set $\{\ldots\}$ is irrelevant.

Given a [sparse] matrix chain product the Vertex Elimination (VE) problem asks for an elimination sequence for all intermediate vertices in the corresponding DAG that minimizes the number of `fma` operations.

Vertex Elimination is NP-complete. (See below for proof)

The Markowitz heuristic eliminates vertices in the order of growing degrees. Additional criteria are required to break ties, e.g, (reverse) natural order.

Application of the Markowitz heuristic natural order tie-breaker to the DAG of the sample sparse matrix chain product yields the vertex elimination sequence $[2, 7, 3, 5, 4, 6]$ and hence also 8 `fma`.

Outlook: Vertex elimination in the context of algorithmic differentiation.

Vertex Elimination is NP-Complete

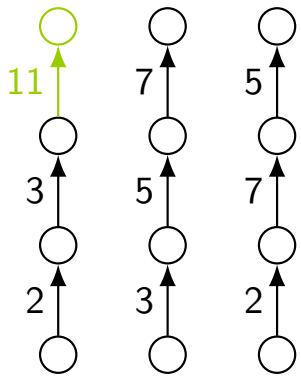
Proof (See module on Matrix Chain Products)

Reduction from Ensemble Computation:

$$A = \{a_1, a_2, a_3, a_4\}$$
$$C = \{\{a_1, a_2\}, \{a_2, a_3, a_4\}, \{a_1, a_4, a_3\}\}$$
$$K = 4?$$

Yes, e.g,

$$C_1 = \{a_1\} \cup \{a_2\}, \quad X = \{a_3\} \cup \{a_4\}$$
$$C_2 = \{a_2\} \cup X, \quad C_3 = \{a_1\} \cup X$$



Diagonal MCP: $a_i \to (i+1)$th prime + fill

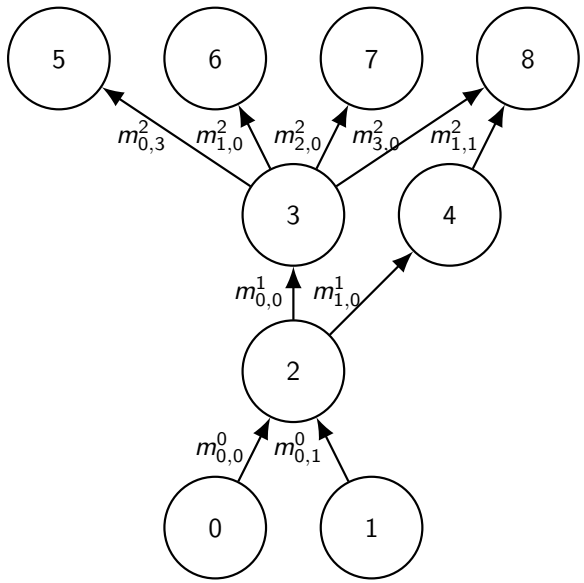
$$\begin{pmatrix} 11 & & \\ & 7 & \\ & & 5 \end{pmatrix} \cdot \begin{pmatrix} 3 & & \\ & 5 & \\ & & 7 \end{pmatrix} \cdot \begin{pmatrix} 2 & & \\ & 3 & \\ & & 2 \end{pmatrix}$$

Outline

Consider the sparse matrix chain product

$$\begin{pmatrix} m^2_{0,0} & 0 \\ m^2_{1,0} & 0 \\ m^2_{2,0} & 0 \\ m^2_{3,0} & m^2_{3,1} \end{pmatrix} \begin{pmatrix} m^1_{0,0} \\ m^1_{0,1} \end{pmatrix} \begin{pmatrix} m^0_{0,0} & m^0_{0,1} \end{pmatrix}$$

with DAG shown on the right. The optimal vertex elimination elimination sequence $[4, 2, 3]$ involves 13 `fma`. Separate application of the Markowitz heuristic with natural order tie-breaker to the DAGs induced by the first three and by the fourth rows (sharing the subgraph induced by vertices 0,1,2,3) yields a cost of only 12 `fma`.



An edge is front-eliminated by connecting its source i with successors $k \in S_j$ of its target and followed by its removal. If $|P_j| = 0$ in the resulting DAG, then the target j is also removed together with all edges emanating from it. Treatment of edge labels is similar to the vertex elimination case. The number of `fma` involved in the front-elimination of (i, j) is equal to $|S_j|$.

An edge (j, k) is back-eliminated by connecting predecessors $i \in P_j$ of its source with k and followed by its removal. If $|S_j| = 0$ in the resulting DAG, then the source j is also removed together with all edges emanating from it. Edge labels are treated as above. The number of `fma` involved in the front-elimination of (j, k) is equal to $|P_j|$.

Elimination of a vertex amounts to front-elimination of its incoming edges as well as to back-elimination of all edges emanating from it.

Given a sparse matrix chain product the Edge Elimination problem asks for an elimination sequence for all eliminatable edges of the corresponding DAG that minimizes the number of `fma` operations.

Edge elimination may result in new edges due to *fill-in* which makes the proof of termination less obvious than in the case vertex elimination.

Edge Elimination is NP complete. The proof is the same as for vertex elimination as both problem turn out to be equivalent for the instances resulting from the reduction.

Degree-based greedy heuristics follow analogously.

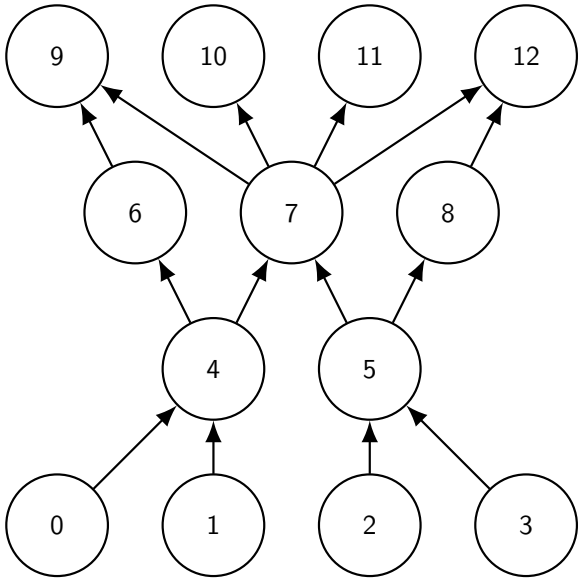
An optimal edge elimination sequence for the *Lion* DAG is $[4, (3, 8), 2, 3]$.

Outline

Face Elimination (Outlook)

Motivation: *Bat* DAG

A solution to Edge Elimination does not necessarily minimize the number of `fma` either.
The *Bat* DAG on the right consists of two overlapping *Lion* DAGs inducing columns 0,1 and 2,3 of the result of the corresponding sparse matrix chain product.



Optimal edge elimination sequences applied to both *Lion* DAGs yield a conflict in the *Bat* DAG which requires face elimination for its resolution.

The corresponding Face Elimination problem is NP-complete. The proof is the same as for vertex/edge elimination as all three problems turn out to be equivalent for the instances resulting from the reduction.

Outline

- The effort for solving the elimination problems needs to be compensated for by the corresponding savings which is not necessarily straightforward.
- Ideally, a sufficiently large number of matrix chain products with similar sparsity patterns should be evaluated. The solution of the combinatorial problem becomes a preprocessing step. Amortization is simplified.
- It remains unclear whether implementation of edge or even face elimination instead of the combinatorially simpler vertex elimination yields adequate improvements.
- Many other problems related to elimination techniques are still waiting to be solved.
- Vertex elimination has been applied successfully in the context of algorithmic differentiation. See module ExtendedJacobianChainProducts.

Outline

Summary

- Introduction to vertex and edge elimination of directed acyclic graphs of [sparse] matrix chain products.

Next Steps

- Run a few examples.
- Compare cost with that obtained by dynamic programming.
- Continue the course to find out more ...