
Part 2: Algorithm Representation

Edgar Solomonik

University of Illinois at Urbana-Champaign
Straight Line Programs

- Often, we want to quantify the efficiency of an algorithm that solves any problem of size $n$ in $f(n)$ iterations, i.e., it is a *straight line program*.

- The completed execution of a program for a particular problem may always be described by a straight line program.
Algorithms as Directed Acyclic Graphs

- A directed acyclic graph (DAG) describes a straight line program in terms of elementwise operations (addition, multiplication, etc.)

- Assuming an algorithm is a straight line program, we may ask questions regarding parallelism and communication cost
Schedules of an Algorithm

- A schedule assigns the vertices of a straight-line program to instructional units and manages associated communication.
Parameterization of Algorithms

- Oftentimes, we may want to parameterize the algorithm (and not just the schedule) depending on the architecture.

- An algorithm may also be designed to be oblivious to a parameter, i.e., to minimize execution time for any choice of a particular parameter.
Matrix Multiplication as a DAG

Let's consider the matrix multiplication problem: compute $C$ such that $C = AB$ with $A, B, C \in \mathbb{R}^{n \times n}$

- Loop-nest can be used to describe algorithm/DAG (for $i$, for $j$, for $k$, $c_{ij}^{(k)} = c_{ij}^{(k-1)} + a_{ik}b_{kj}$ with $c_{ij}^{(0)} = 0$ and $c_{ij} = c_{ij}^{(n)}$)

- Recursive formulation describes another algorithm/DAG
Family of Classical Matrix Multiplication Algorithms

- The nested-loop and recursive formulations are two instances of a family of classical matrix multiplication algorithms

- Can describe family of DAGs as a hypergraph
Surface Area to Volume Ratio in Hypergraphs

- We can analyze the hypergraph to determine communication cost bounds.

- The *Loomis-Whitney* is a *volumetric inequality* that provides a way to bound expansion.
Compression and Recomputation

- Our previous discussion of communication assumed that each hypergraph edge requires communication of a matrix entry

- A method that computes bilinear products $a_{ik}b_{kj}$ may take arbitrary linear combinations of entries of $A$, $B$, or partial sums for $C$
Bilinear Algorithms

A bilinear algorithm (V. Pan, 1984) \( \Lambda = (\mathbf{F}^{(A)}, \mathbf{F}^{(B)}, \mathbf{F}^{(C)}) \) computes

\[
c = \mathbf{F}^{(C)}[(\mathbf{F}^{(A)^T} a) \odot (\mathbf{F}^{(B)^T} b)],
\]

where \( a \) and \( b \) are inputs and \( \odot \) is the Hadamard (pointwise) product.
Bilinear Algorithms as Tensor Factorizations

- A bilinear algorithm corresponds to a CP tensor decomposition

- For multiplication of $n \times n$ matrices, we can define a *matrix multiplication tensor* and consider algorithms with various bilinear rank
Strassen’s Algorithm

Strassen’s algorithm \[
\begin{bmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{bmatrix}
= \begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\cdot
\begin{bmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{bmatrix}
\]

\[
M_1 = (A_{11} + A_{22}) \cdot (B_{11} + B_{22})
\]
\[
M_2 = (A_{21} + A_{22}) \cdot B_{11}
\]
\[
M_3 = A_{11} \cdot (B_{12} - B_{22})
\]
\[
M_4 = A_{22} \cdot (B_{21} - B_{11})
\]
\[
M_5 = (A_{11} + A_{12}) \cdot B_{22}
\]
\[
M_6 = (A_{21} - A_{11}) \cdot (B_{11} + B_{12})
\]
\[
M_7 = (A_{12} - A_{22}) \cdot (B_{21} + B_{22})
\]

\[
C_{11} = M_1 + M_4 - M_5 + M_7
\]
\[
C_{21} = M_2 + M_4
\]
\[
C_{12} = M_3 + M_5
\]
\[
C_{22} = M_1 - M_2 + M_3 + M_6
\]

By performing the nested calls recursively, Strassen’s algorithm achieves cost,
Expansion in Bilinear Algorithms

- The communication cost of a bilinear algorithm depends on the amount of data needed to compute subsets of the bilinear products.

- A bilinear algorithm $\Lambda$ can be associated expansion bound $E_\Lambda : \mathbb{N}^3 \rightarrow \mathbb{N}$