CS 598: Provably Efficient Algorithms for Numerical and Combinatorial Problems

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 maanages associated communication

Parameterization of Algorithms

- Oftentimes, we may want to paramterize 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

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

- 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 $\boldsymbol{A}$, $\boldsymbol{B}$, or partial sums for $\boldsymbol{C}$

Bilinear Algorithms

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

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

where $\boldsymbol{a}$ and $\boldsymbol{b}$ 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 $\begin{bmatrix} \boldsymbol{C}_{11} & \boldsymbol{C}_{12} \\ \boldsymbol{C}_{21} & \boldsymbol{C}_{22} \end{bmatrix} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{B}_{11} & \boldsymbol{B}_{12} \\ \boldsymbol{B}_{21} & \boldsymbol{B}_{22} \end{bmatrix}$

$$\begin{aligned}
\boldsymbol{M}_1 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{22}) \\
\boldsymbol{M}_2 &= (\boldsymbol{A}_{21} + \boldsymbol{A}_{22}) \cdot \boldsymbol{B}_{11} \\
\boldsymbol{M}_3 &= \boldsymbol{A}_{11} \cdot (\boldsymbol{B}_{12} - \boldsymbol{B}_{22}) \\
\boldsymbol{M}_4 &= \boldsymbol{A}_{22} \cdot (\boldsymbol{B}_{21} - \boldsymbol{B}_{11}) \\
\boldsymbol{M}_5 &= (\boldsymbol{A}_{11} + \boldsymbol{A}_{12}) \cdot \boldsymbol{B}_{22} \\
\boldsymbol{M}_6 &= (\boldsymbol{A}_{21} - \boldsymbol{A}_{11}) \cdot (\boldsymbol{B}_{11} + \boldsymbol{B}_{12}) \\
\boldsymbol{M}_7 &= (\boldsymbol{A}_{12} - \boldsymbol{A}_{22}) \cdot (\boldsymbol{B}_{21} + \boldsymbol{B}_{22})
\end{aligned}$$

$$\begin{aligned}
\boldsymbol{C}_{11} &= \boldsymbol{M}_1 + \boldsymbol{M}_4 - \boldsymbol{M}_5 + \boldsymbol{M}_7 \\
\boldsymbol{C}_{21} &= \boldsymbol{M}_2 + \boldsymbol{M}_4 \\
\boldsymbol{C}_{12} &= \boldsymbol{M}_3 + \boldsymbol{M}_5 \\
\boldsymbol{C}_{22} &= \boldsymbol{M}_1 - \boldsymbol{M}_2 + \boldsymbol{M}_3 + \boldsymbol{M}_6
\end{aligned}$$

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 Λ can be associated expansion bound $\mathcal{E}_\Lambda : \mathbb{N}^3 \to \mathbb{N}$