Multilinear Operations on Sparse Tensors & Graphs

Tim Baer

CS 598: Tensor Computations
University of Illinois at Urbana-Champaign

December 7, 2020

Overview

BFS

Given an undirected graph G represented as an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, compute a Breadth First Search starting from vertex s.

Let $\boldsymbol{f}^{(i)}$ denote the frontier and $\boldsymbol{u}^{(i)}$ denote the unvisited vertices at iteration i.

Algorithm BFS

1: $\boldsymbol{f}^{(0)}$ is zero everywhere except $\boldsymbol{f}_s^{(0)} = 1$
2: $\boldsymbol{u}^{(0)}$ is one everywhere except $\boldsymbol{u}_s^{(0)} = 0$
3: **for** $i = 1$ to D **do**
4: $\quad \boldsymbol{f}^{(i+1)} = \boldsymbol{u}^{(i)} \odot (\boldsymbol{A}\boldsymbol{f}^{(i)})$
5: $\quad \boldsymbol{u}^{(i+1)} = \boldsymbol{u} - \boldsymbol{f}^{(i+1)}$

BFS Matrix-Vector Product

Notice that $\boldsymbol{A}$, $\boldsymbol{f}^{(i)}$, and $\boldsymbol{u}^{(i)}$ are sparse!

Consider the BFS matrix-vector product $\boldsymbol{A}\boldsymbol{f}^{(i)}$.

- Sparse-matrix-vector-product (SpMV)
- Sparse-matrix-sparse-vector product (SpMSpV)

- Which matrix-vector formulation is better?
  Depends on PRAM model, storage format, etc.
- How about the output filter $\boldsymbol{u}^{(i)} \odot (\boldsymbol{A}\boldsymbol{f}^{(i)})$?
  Leveraging both sparsity of filter and of input vector is hard!
- Since BFS is a special instance of SSSP, can we do something similar for shortest paths?
  Use semirings!

Motivation

Multilinear operations on sparse tensors and graphs arise in many computing applications.

- BFS: compute the connected component containing a given vertex
- Output filters: compute output for a subset of indices
- MTTKRP: compute batches of multilinear operations
- Balls: compute the b closest vertices to a given vertex
- Minimum spanning trees: compute the minimum weight tree that connects all vertices
- TTTP and beyond

Semirings

A semiring is a set equipped with two binary operators $(+, *)$ satisfying the following conditions:

- Additive associativity
- Additive commutativity
- Multiplicative associativity
- Left and right distributivity

Examples:

- $(+, *)$ on $\mathbb{R}$
- $(\min, +)$ on $\mathbb{R}$

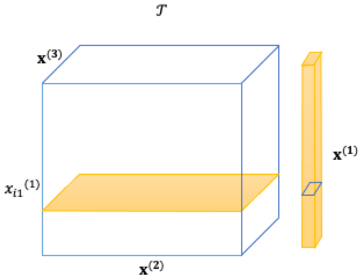
Note: Strassen's algorithm holds for rings[1], but not necessarily semirings (ex: $(\min, +)$).

[1]Semiring with additive inverse and 0 element

Multilinear Operations

Given an order d tensor $\mathcal{T}$ with indices $\{i_1, \ldots, i_d\}$, define a multilinear operation $\boldsymbol{f}^{\mathcal{T}} : \mathbb{R}^{n_2} \times \cdots \mathbb{R}^{n_d} \to \mathbb{R}^{n_1}$ given by

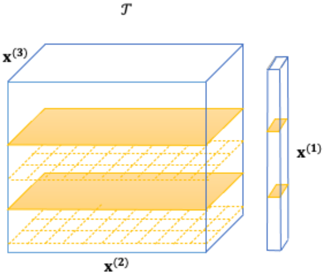
$$x_{i_1}^{(1)} = \sum_{i_2 \ldots, i_d} t_{i_1 \ldots i_d} x_{i_2}^{(2)} \cdots x_{i_d}^{(d)} \implies \boldsymbol{f}^{\mathcal{T}}(\boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(d)}) = \mathcal{T}_{(1)} \bigotimes_{j=2}^{d} \boldsymbol{x}^{(j)}$$



Output Filters in Bilinear Algorithms

Given a bilinear map $\boldsymbol{f}^{\mathcal{T}}(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)})$ where $\boldsymbol{x}^{(2)} \in \mathbb{R}^{n_2}$ and $\boldsymbol{x}^{(3)} \in \mathbb{R}^{n_3}$, we can equip it with an output filter $\boldsymbol{\lambda} \in \{0, 1\}^{n_1}$ to construct a multilinear map

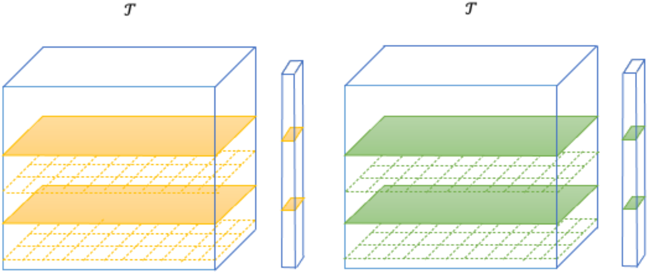
$$\boldsymbol{f}^{\mathcal{T}}_{\lambda}(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)}) = \boldsymbol{\lambda} \odot \boldsymbol{f}^{\mathcal{T}}(\boldsymbol{x}^{(2)}, \boldsymbol{x}^{(3)})$$



MTTKRP

We can interpret MTTKRP as a set of multilinear operations. Given a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n\times\cdots\times n}$ and matrices $\boldsymbol{U}^{(1)}, \dots, \boldsymbol{U}^{(d)} \in \mathbb{R}^{n\times R}$, compute

$$u^{(1)}_{i_1 r} = \sum_{i_2\dots i_d} t_{i_1\dots i_d} u^{(2)}_{i_2 r} \cdots u^{(d)}_{i_d r} \implies \boldsymbol{u}^{(1)}_r = \boldsymbol{f}^{\boldsymbol{\mathcal{T}}}(\boldsymbol{u}^{(2)}_r, \dots, \boldsymbol{u}^{(d)}_r) = \boldsymbol{\mathcal{T}} \times_2 \boldsymbol{u}^{(2)}_r \times_3$$



MTTKRP in CSF with Output Filter

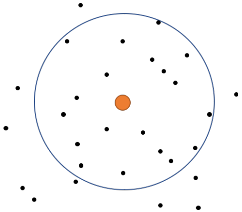
Consider computing MTTKRP $u_{ir} = \sum_{j,k} t_{ijk} v_{jr} w_{kr}$ with output filter $\boldsymbol{\lambda}$

```
for (i,T_i) in T_CSF:
    for (j,T_ij) in T_i:
        for r in range(R):
            if lambda_ir == 1:
                f_ij = 0
                for (k,t_ijk) in T_ij:
                    f_ij += t_ijk * w[k,r]
                u[i,r] += f_ij * v[j,r]
```

- Unamortized algorithm more efficient
- Exact number of operations depends on distribution of nonzeros

Ball Computation

Consider the ball computation: given a vertex v, compute the b closest vertices to v.



Let $B_u^{(i)}$ be a sorted b-vector containing tentative distances to the b closest vertices to vertex u.

$$B_u^{(i+1)} = B_u^{(i)} \oplus \Big(\bigoplus_{(u,v)\in E} \{w(u,v) + B_v^{(i)}\} \Big)$$

where the reduction operator $x \oplus y$ merges x and y and returns the first b distances.

Ball Computation

We can express

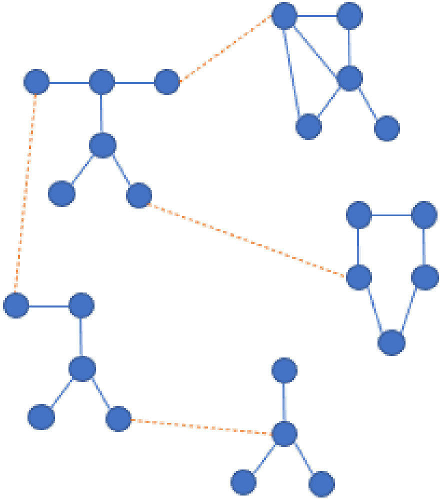
$$B_u^{(i+1)} = B_u^{(i)} \oplus \Big(\bigoplus_{(u,v)\in E} \{w(u,v) + B_v^{(i)}\}\Big)$$

as a multilinear operation

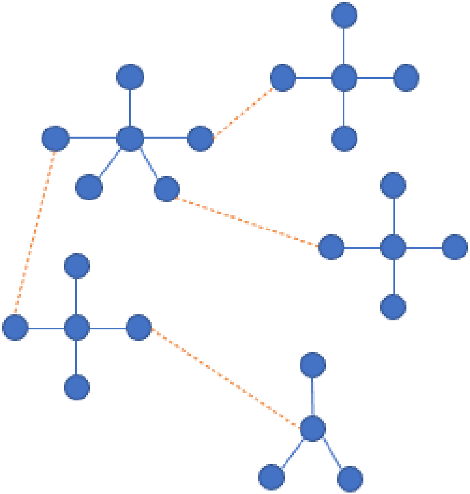
$$B_u^{(i+1)} = \bigoplus_{(u,v)\in E} \{B_u^{(i)} \oplus (w(u,v) + B_v^{(i)})\}$$

1. Requires $\mathcal{O}(b)$ iterations
2. Can be sparsified by specifying a row filter
3. If $B_u^{(i)}[b] < B_v^{(i)}[1]$ for many v, multilinear approach is more efficient

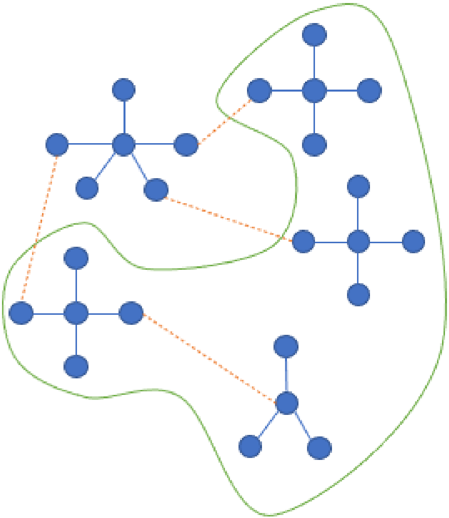
Forest of Trees



Stars



Minimum Outgoing Edges



Awerbuch-Shiloach Algorithm

Given an undirected graph $G = (V, E)$ with n vertices and m edges equipped with weights $w : E \to \mathbb{R}$, compute a minimum spanning tree.

Let $p \in \mathbb{R}^n$ denote the parent vector and T_i denote the tree containing vertex i.

1. Unconditional star hooking: joins two trees together.

$$p[i] \leftarrow p[\operatorname*{argmin}_{e \in \text{cut}(T_i, V \setminus T_i)} w(e)]$$

2. Tie breaking: prevents cycles by only allowing hooking in one direction

$$\begin{cases} \text{write succeeds if } i < j \\ \text{write fails otherwise} \end{cases}$$

3. Shortcutting: reduces the height of each tree in the forest by a factor of nearly two.

$$p[i] = p[p[i]]$$

MST as a Multilinear Operation

Let $\boldsymbol{p} \in \mathbb{R}^n$ denote the parent vector and $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ denote the adjacency matrix, on the $(\min)$ monoid (think semiring without $*$).

f computes for each pair of nodes $\{i, j\}$, whether they share a parent.

$$f(p_i, a_{ij}, p_j) = \begin{cases} a_{ij} : p_i \neq p_j \\ \infty : \text{otherwise} \end{cases}$$

Accumulating $f(p_i, a_{ij}, p_j)$ over min computes for each node i, its lightest edge that *could* discover a new tree.

$$q_i = \bigoplus_j f(p_i, a_{ij}, p_j)$$

Next, for each node i, we contract its children's lightest "discovery" edges into itself

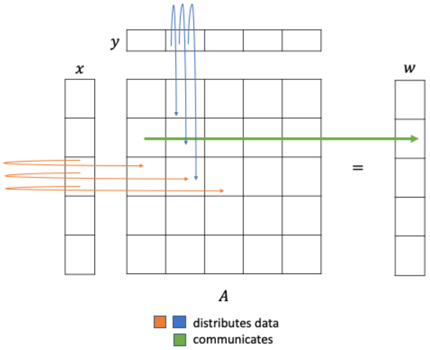
$$r_{p_j} \underset{\textsc{MinWeight}}{\leftarrow} q_j$$

Data Distribution

How can we leverage the additional structure of $f(p_i, a_{ij}, p_j)$?

$$w_i = \bigoplus_j f(x_i, a_{ij}, y_j)$$

where process (r, s) owns $A^{(r,s)}$ and needs $x^{(r)}$ and $y^{(s)}$. Process (r, s) contributes $w^{(r)}$ where $w^{(r)} = f(x^{(r)}, A^{(r,s)}, y^{(s)})$.



Sparse Multilinear Kernel

How can we leverage additional structure for other classes of sparse multilinear operations?

- TTTP: Given a sparse tensor $\boldsymbol{S} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and a list of N matrices $\boldsymbol{A}^{(1)} \in \mathbb{R}^{I_1 \times R}$, ..., $\boldsymbol{A}^{(N)} \in \mathbb{R}^{I_N \times R}$, compute

$$x_{i_1 \dots i_d} = t_{i_1 \dots i_d} \sum_{r=1}^{R} \prod_{j=1}^{d} a_{i_j r}^{(j)}$$

- MTTKRP: Given a tensor $\boldsymbol{\mathcal{T}} \in \mathbb{R}^{n \times \dots \times n}$ and matrices $\boldsymbol{U}^{(1)}, \dots, \boldsymbol{U}^{(d)} \in \mathbb{R}^{n \times R}$, compute

$$u_{i_1 r}^{(1)} = \sum_{i_2 \dots i_d} t_{i_1 \dots i_d} u_{i_2 r}^{(2)} \cdots u_{i_d r}^{(d)}$$

TTTP Data Distribution[2]

Consider an order 3 TTTP:

$$x_{i_1 i_2 i_3} = t_{i_1 \dots i_3} \sum_{r=1}^{R} a^{(1)}_{i_1 r} a^{(2)}_{i_2 r} a^{(3)}_{i_3 r}$$

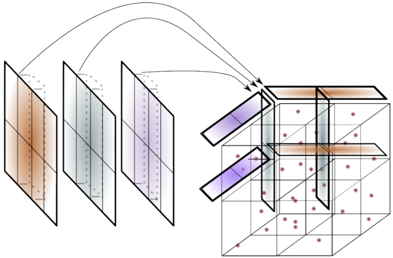


Figure 2: Depiction of 8 processor parallelization of TTTP computing one of four smaller TTTP substeps.

Note: MTTKRP has same data distribution but communicates (reduces) across slices.

[2]Enabling Distributed-Memory Tensor Completion in Python using New Spare Tensor Kernels by Z. Zhang, Wu, N. Zhang, S. Zhang, and Solomonik (2019)

Summary

1. Multilinear Operations
2. Output Filters
3. MTTKRP
4. Balls
5. Minimum Spanning Trees

References

- CS: 598 Provably Efficient Algorithms for Numerical and Combinatorial Problems
- CS 598: Tensor Computations
- *Parallel Approximate Undirected Shortest Paths Via Low Hop Emulators* by Andoni, Stein, and Zhong (STOC 2020)
- *New Connectivity and MSF Algorithms for Shuffle-Exchange Network and PRAM* by Awerbuch and Shiloach (1987)
- *Enabling Distributed-Memory Tensor Completion in Python using New Sparse Tensor Kernels* by Z. Zhang, Wu, N. Zhang, S. Zhang, and Solomonik (2019)

Fin

Thanks. Questions?