Parallel Minimum Spanning Forest Computation using Sparse Matrix Kernels

Tim Baer, Raghavendra Kanakagiri, and Edgar Solomonik

SIAM PP22
University of Illinois at Urbana-Champaign

February 25, 2022

Overview

Motivation

Generalized matrix algebra is an expressive and powerful framework for designing parallel graph algorithms. We can leverage:

- Sparsity
- Parallelism
- All-at-once kernels
- Hardware accelerators

Our Contributions

We make the following novel contributions:

- First distributed-memory implementation of the Awerbuch-Shiloach (AS) MSF algorithm
- New all-at-once kernel
- New optimization to the shortcutting step
- Scalability study on graphs with up to 11 billion edges and 183 million vertices

Background

Given an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$. We label the vertices $V = \{1, \dots, n\}$.

A minimum spanning forest (MSF) is a subset of edges that connects all the vertices in each connected component with the minimum possible total edge weight. Applications include:

- Network design for electrical grids, ...
- Approximation algorithms for traveling salesman problem, ...

We will use the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ of G where

$$a_{ij} \leftarrow \begin{cases} w(i,j) & \text{if } (i,j) \in E \\ \infty & \text{otherwise} \end{cases}.$$

A minimum spanning forest is unique if each edge has a distinct weight.

Graph Libraries

Graph libraries including CombBLAS[1], GraphBLAS[2], and CTF[3] allow users to implement graph algorithms with sparse matrix algebra over different *semirings*.

A semiring $(\mathbb{S}, \oplus, \otimes)$ is a set $\mathbb{S}$ equipped with binary operations $\oplus, \otimes : \mathbb{S} \times \mathbb{S} \to \mathbb{S}$ called addition and multiplication, respectively.

Operations with the adjacency matrix typically iterate over the neighbors of each vertex:

- SpMVs combine edge data with vertex data
- SpMSpMs combine edge data with data between vertices

All-at-once kernels consider ≥ 3 operands *simultaneously*.

[1]A. Buluç, J. Gilbert, IJHPCA 2011
[2]A. Buluç, T. Mattson, S. McMillan, J. Moreira, C. Yang, IPDPSW 2017
[3]E. Solomonik, D. Matthews, J. Hammond, J. Stanton, J. Demmel, JPDC 2014

Awerbuch-Shiloach[4] (AS) Algorithm

Maintain a forest of trees, where each vertex has a parent.

At the first iteration, the forest consists of each vertex with a self-loop.

Repeat the following steps until convergence of the forest:

(i) Star hooking: grows the forest by joining a star[5] to another tree

(ii) Tie-breaking: detects and breaks cycles

(iii) Shortcutting: compresses trees into stars

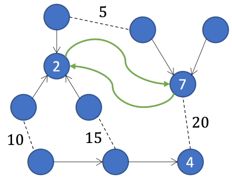
The AS algorithm computes connected components (CC) or MSF in $O(\log n)$ PRAM depth.

[4] B. Awerbuch, Y. Shiloach, IEEE Trans Comput 1987

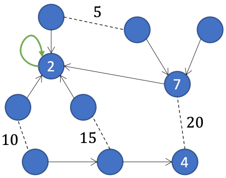
[5] A star is a directed tree of height at most 1. A star root is a vertex that is its own parent.

Awerbuch-Shiloach (AS) Algorithm

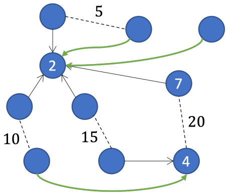
The parent vector $p \in V^n$ stores the parent of each vertex.



(a) Star hooking: star roots 2 and 7 hook. The tree rooted at 4 is not a star so it does not hook.



(b) Tie breaking: breaks the cycle introduced in star hooking by setting 2 to be its own parent.



(c) Shortcutting: changes each vertex's parent to the parent of its parent.

Figure: AS algorithm steps where dotted black lines are outgoing edges, solid black lines are parent pointers, and solid green lines are the updated parent pointers.

LACC & FastSV

LACC[6] and FastSV[7] present distributed-memory implementations of the AS CC algorithm[8]. They express each step in CombBLAS, including star hooking as SpMV.

An outgoing edge from a tree T is an edge (i, j) such that $i \in T$ and $j \notin T$. A minimum outgoing edge from a tree is an outgoing edge with the smallest weight.

The main difference between the AS CC and MSF variants is:

- For CC, we can hook with *any* outgoing edge
- For MSF, we must hook with the *minimum* outgoing edge

[6] A. Azad, A. Buluç, IPDPS 2019
[7] Y. Zhang, A. Azad, Z. Hu PP 2020
[8] FastSV is actually based on the closely related Shiloach-Vishkin algorithm.

Star Hooking

Each star hooks with its minimum outgoing edge.

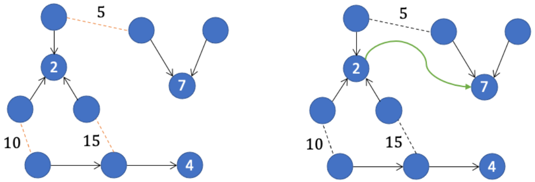


Figure: Star hooking steps where dotted orange lines are minimum outgoing edges from vertices, solid black lines are parent pointers, and solid green lines are the updated parent pointers.

We may compute this in $O(1)$ depth with the following routine:

(i) Each vertex that belongs to a star computes the minimum outgoing edge that it is adjacent to

(ii) Each star root computes the smallest minimum outgoing edge among all of its children

Star Hooking as a Multilinear Function

f decides for each pair of vertices (i, j), whether they belong to the same star,

$$f(p_i, a_{ij}, p_j) = \begin{cases} a_{ij} & : p_i \neq p_j \text{ and } i \text{ belongs to a star,} \\ \infty & : \text{otherwise.} \end{cases}$$

We may compute the weight of the minimum outgoing edge adjacent to vertex i with

$$q_i \underset{\min_j}{\leftarrow} f(p_i, a_{ij}, p_j).$$

For each star root i, we *contract* its children's minimum outgoing edges into itself,

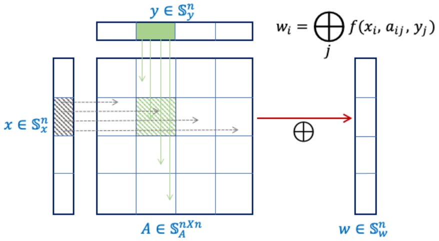
$$r_{p_j} \underset{\min}{\leftarrow} q_j.$$

Multilinear Kernel

We update vertices all-at-once by using data from an edge and its adjacent vertices. More generally,

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

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



Supports sparse input matrix $\boldsymbol{A}$ and decreases the number of writes. Cost is comparable to SpMV using a 2D distribution.

M. K. Rahman, M. H. Sujon, A. Azad, IPDPS 2021 introduced the FusedMM kernel that generalizes SDDMM and SpMM.

Complete Shortcutting

Shortcutting reduces the height of each tree by nearly half,

$$p_i \leftarrow p_{p_i}.$$

For real-world graphs, we observe that it often takes only a few shortcut iterations to turn all trees into stars.

We introduce complete shortcutting: at the end of each iteration, we shortcut repeatedly until each tree is a star. This choice

- Simplifies algorithm
- Enables new optimizations
- Faster experimentally

Even though the overall depth is worse by a $\Theta(\log n)$ factor.

Complete Shortcutting with Prefetching (CSP)

We propose the CSP optimization to perform complete shortcutting in $O(1)$ communication rounds.

We may compute this with the following routine:

(i) Locally find vertices whose parent changed after hooking

(ii) Gather these vertices and their parents on all processes

(iii) Use this local data for complete shortcutting

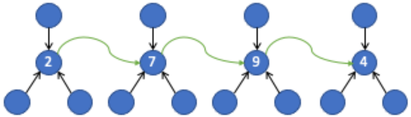


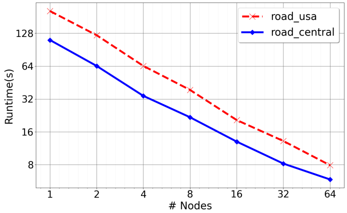
Figure: Each processor has a local copy of the map *changed* = $\{(2,7),(7,9),(9,4)\}$ so star root 2 shortcuts onto 9 then 4 without any more communication.

The *changed* map must be small enough to fit into local memory, which is usually the case after the first shortcutting iteration.

Experimental Comparison

ID	Graph	Vertices	Edges
road-usa	Full USA road-network	23.9M	28.9M
road-central	Central USA road-network	14.1M	16.9M

On *road-usa*, Panja et. al[9] report 190s and 29.6s using Pregel+ and MND-MST, respectively. They report slowdowns > 16 nodes.

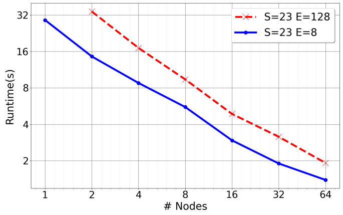


(a) DIMACS road network graphs.

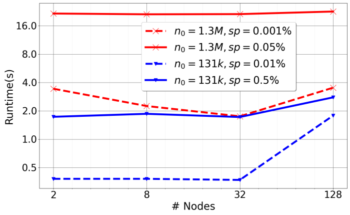
[9]R. Panja, S. Vadhiyar, ICPP 2018

Stampede2 supercomputer, each node has a KNL CPU with 68 cores. We use 16 MPI processes each with 4 OMP threads per node. Note that the computing platforms are different in each of the experimental setups.

Experimental Scaling



(b) Strong scaling of R-MAT graphs with $\log_2(n) \approx S = 23$ and average degree $k \approx E = 8, 128$.



(c) Edge weak scaling of uniform random graphs. Constant $n^2/p = n_0^2$ and edge percentage $f = 100 * m/n^2$.

Stampede2 supercomputer, each node has a KNL CPU with 68 cores. We use 16 MPI processes each with 4 OMP threads per node.

Conclusions

- All-at-once kernels like the *multilinear kernel* can be leveraged to design efficient algorithms
- *Complete shortcutting* simplifies the AS algorithm and enables new optimizations
- Our algebraic formulation with the *CSP optimization* achieves excellent strong and weak scaling for various graphs

T. Baer, R. Kanakagiri, and E. Solomonik, Parallel minimum spanning forest computation using sparse matrix kernels, Proceedings of the 2022 SIAM Conference on Parallel Processing for Scientific Computing (PP), pp. 72–83, https://epubs.siam.org/doi/abs/10.1137/1.9781611977141.7.

Fin

Thanks. Questions?