My research bridges the theory and practice of parallel algorithms, with an emphasis on large graphs. Graph algorithms are often difficult to parallelize due to their highly irregular structure. Libraries that support arbitrary graph operations require a lot of custom code and careful design to fully exploit the available parallelism. My work instead leverages generalized matrix algebra as an expressive and powerful framework for designing parallel graph algorithms. Since matrix multiplications have been highly optimized in the parallel setting, this approach allows us to leverage them to achieve excellent scaling results.

The minimum spanning forest (MSF) problem has many practical applications in network design and is used as a subroutine in numerous approximation algorithms. We design the first formulation of the Awerbuch-Shiloach parallel minimum spanning forest (MSF) algorithm using matrix algebra primitives[1]. To achieve this, we introduce a novel multilinear primitive that updates graph vertices by simultaneously using information from both adjacent edges and vertices. We demonstrate scalability of our MSF algorithm on up to 256 nodes (16K cores) of Stampede2 on graphs with up to 11 billion edges and 183 million vertices. I was responsible for the design of the algorithm and implementation with the Cyclops Tensor Framework, a distributed-memory library for generalized sparse tensor algebra. When benchmarking, a bottleneck was the shortcutting step where each vertex changes its parent to be the parent of its parent. To alleviate this, I designed and implemented an optimization that collects roots on all processes and avoids shortcuts for vertices of height at most 1. In addition, I developed theoretical justification for a second optimization that shortcuts repeatedly until a tree becomes a star with only a single round of communication. I also contributed to the implementation of this multilinear kernel as a part of the CTF library.

Single source shortest path (SSSP) algorithms have been applied at a massive scale to optimize driving directions and facility location. My second project with Mengyuan (Mina) Sun and Edgar Solomonik explores connections between a recent theory paper on parallel SSSP in nearly optimal work and depth (Andoni, Stein, Zhong, STOC 20) and fast iterative solvers like algebraic multigrid. I developed a highly scalable algorithm for the main subroutine: given a vertex and a number k, find the k closest neighbors to that vertex. I expressed this calculation as a sequence of sparse matrix-vector multiplications (SpMV) with the adjacency matrix and a vector $\boldsymbol{b}$ containing a sorted list for each vertex's tentative k closest neighbors. The addition operation merges two sorted lists to keep only the k closest neighbors between them and the multiplication operation adds the cost of exploring vertices another hop away. Since b_i is unlikely to be improved many hops away from a vertex i, I found experimentally that it is faster to express this routine with the multilinear primitive described above to avoid merging when the furthest neighbor in b_i is closer than the closest neighbor in the other sorted list. With this efficient subroutine, I then implemented the algorithm's main data structure called a low hop emulator and am currently exploring different algorithms to leverage it in the SSSP calculation. In addition, I plan to explore some of the low hop emulator's applications, including in metric space embedding and graph decomposition.

[1]Tim Baer, Raghavendra Kanakagiri, and Edgar Solomonik. *Parallel Minimum Spanning Forest Computation using Sparse Matrix Kernels.* In submission to PP22 (available on arXiv).