Parallel Algorithms for Updating Dynamic Networks using Graph Sparsification

Sriram Srinivasan

University of Nebraska at Omaha, USA

Introduction

Algorithms for analyzing dynamic networks help study how properties of complex systems evolve with time. Since the networks are very large, parallel algorithms are essentiall. However, because the data is highly unstructured and exhibit poor locality of access, designing scalable dynamic algorithms is very challenging [1,2]. We present a **framework for creating parallel algorithms for updating dynamic network**, using graph sparsification [3].

Key Ideas:

1. Traversal is the most expensive operation in network algorithms. Divide network into subgraphs for faster traversal

2. Sparsification tree[4], localizes the subgraph. Need only to search height of tree for updates.

3. Only a small fraction of edges are essential for update. Prioritize analyzing these key edges

Contributions:

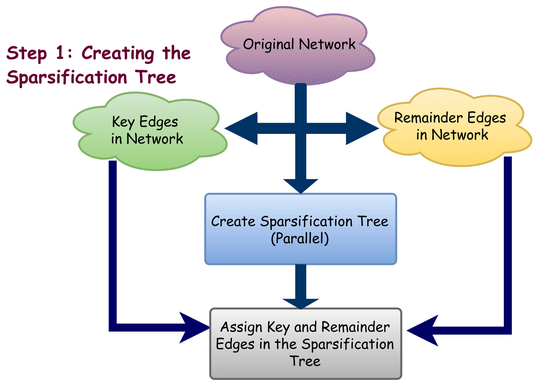
1. First parallel (shared memory) implementation of graph sparsification
2. First framework for updating dynamic network algorithms
3. Scalable algorithms for connected paths, minimum weighted spanning tree (MST)
4. Edge mapping algorithm based on longest paths for faster computation

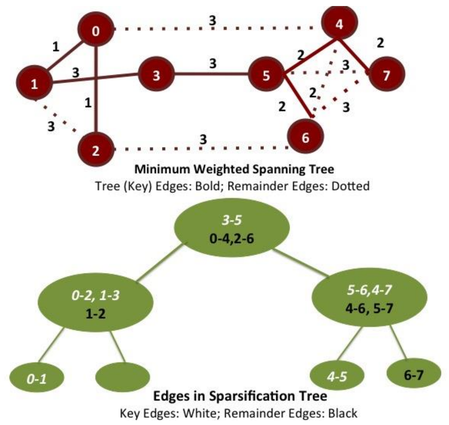
References

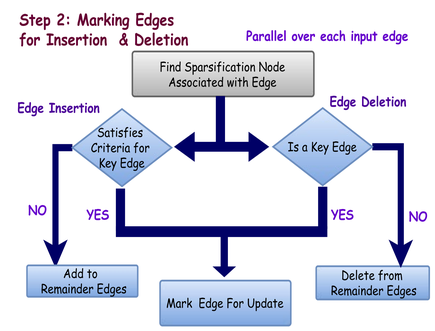
[1]. D. A. Bader, J. Berry, A. Amos-Binks, D. Chavarría-Miranda, C. Hastings, K. Madduri, and S. C. Poulos, "Stinger: Spatio-temporal interaction networks and graphs (sting) extensible representation," *Ga. Inst. Technol. Tech Rep*, 2009.
[2] J. Plimpton and T. Shead "Streaming data analytics via message passing with application to graph algorithms". Technical Report. http://wsga.sandia.gov/docs/Plimpton_stream.pdf.
[3]. D. Eppstein, Z. Galil, G. F. Italiano, and A. Nissenzweig, "Sparsification—
a technique for speeding up dynamic graph algorithms," *J. ACM JACM*, vol. 44, no. 5, pp. 669–696, 1997.
[4]. S. Srinivasan, S. Bhowmick, and S. Das, "Application of Graph Sparsification in Developing Parallel Algorithms for Updating Dynamic Networks," Graph Algorithms Building Blocks Workshop at IPDPS 2016.
[5]. D. Chakrabarti, Y. Zhan, and C. Faloutsos. "R-MAT: A recursive modelfor graph mining". SDM, 2004.
[6]. A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for
testing community detection algorithms. Phys. Rev. E, 78(046110), 2008.

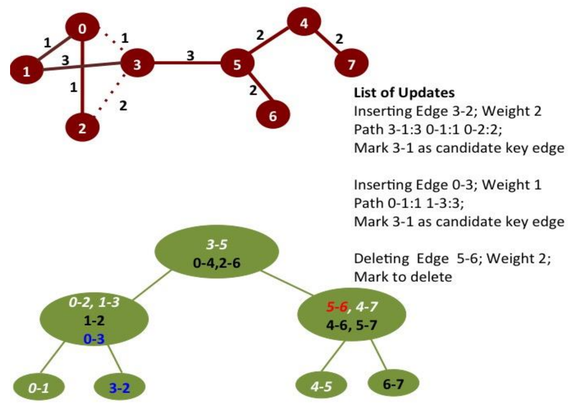
Acknowledgements: Funded by NSF-CCF; Award number 1533881.

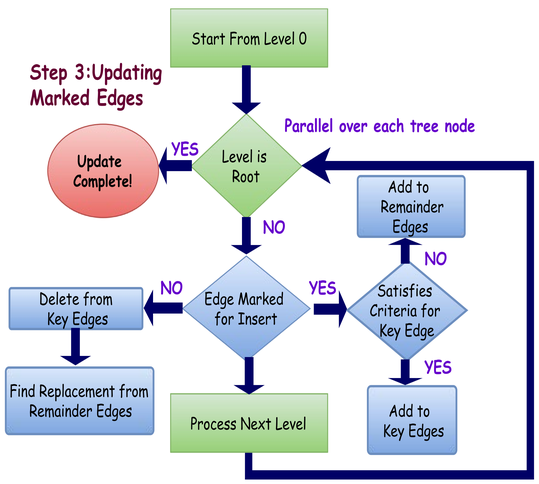
Updating Networks Using Graph Sparsification

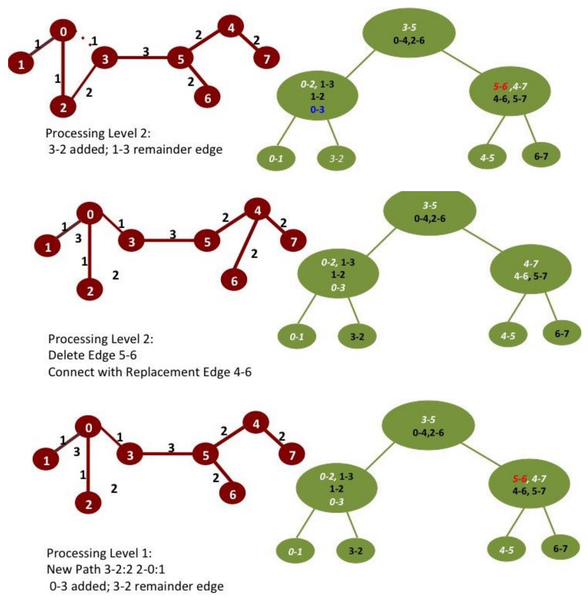












Parallel code Sequential code Code for I/O

Experimental Results

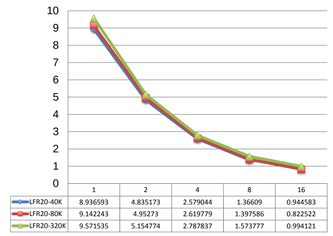
Datasets:
Random Networks (RMAT [5])
Community Rich Networks (LFR[6])

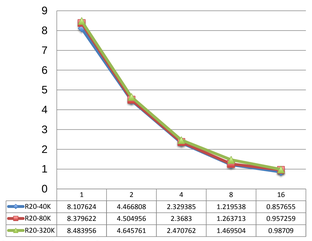
Updates:
Combination of Inserts and Deletes without repeats (40K, 80K, 320K)

Machine:
Tusker at Holland Computing Center.
6,784 cores 523TB Lustre storage

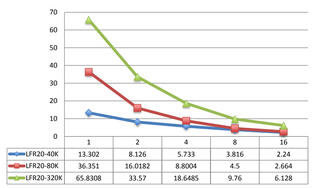
Scalability Results

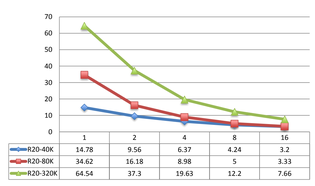
Network Size:
RMAT: ~ 1 million vertices X8 edges
LFR: ~ 1 million vertices X10 edges.
X-axis: Time to update network (in seconds) Y-axis: Number of threads





Results for Connected Paths (log-scale)





Results for MST: only insertions (log-scale)

Edge Mapping

More scalability is obtained if edges are in the lower tree nodes. We developed, *path ordering*, a vertex ordering technique where the unbranched paths in the tree are numbered first. Thus longer paths are in the lower nodes, significantly reducing the runtime of Step 2.

Percentage of Edges in Top Layer with Random Ordering is 75%
Percentage of Edges in Top layer with Path Ordering reduces to 35%

Time to execute Step 2 for MST (update: 320K edges, vertices~1 million) is 2.1 seconds with path ordering. The step takes over 6 minutes with random ordering

Scan QR code to visit my graph sparsification website
https://graphsparsification.herokuapp.com