Push-Pull on Graphs is Column- and Row-based SpMV Plus Masks

Carl Yang[1,2], Aydın Buluç[2,3] and John D. Owens[1]

[1] Department of Electrical and Computer Engineering, UC Davis
[2] Computational Research Division, LBNL
[3] Department of Electrical Engineering and Computer Sciences, UC Berkeley

Objectives

- Investigate the generalizability of direction-optimized BFS.
- Investigate how direction-optimized BFS can be expressed using linear algebra.

Introduction

Push-pull, also known as direction-optimized breadth-first-search (DOBFS), is a key optimization for making breadth-first-search (BFS) run efficiently. Linear algebra-based frameworks have advantages in conciseness, performance and portability. However, there is no work in literature describing how to implement it within a linear algebra-based framework. Our work shows that DOBFS fits well within the linear algebra-based framework.

Traversal is Matvec

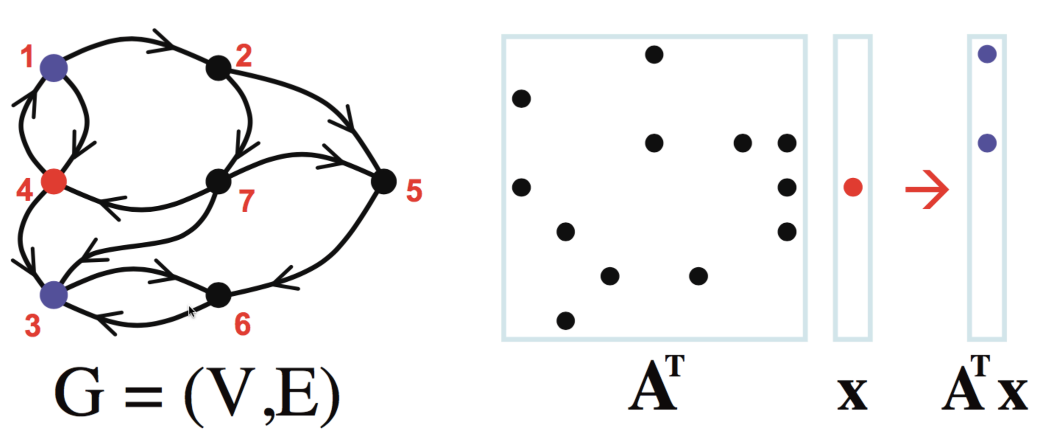


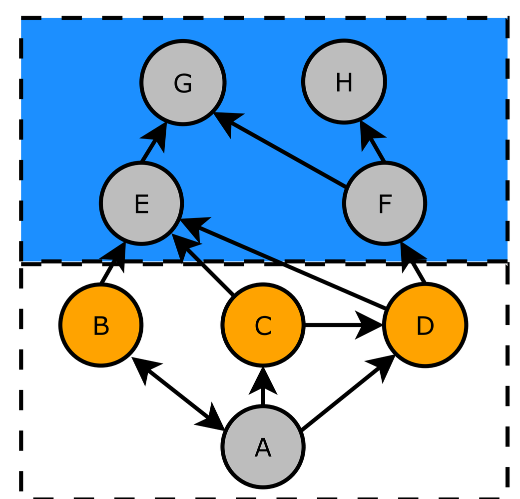
Figure: Matrix-graph duality. The adjacency matrix $\mathbf{A}$ is the dual of graph G. The matvec is the dual of the BFS graph traversal. Figure is based on Kepner and Gilbert's book.

Complexity Results

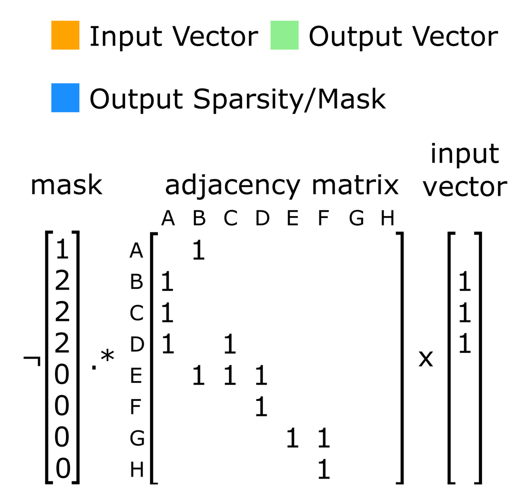
Operation		Expected Cost
Row-based	unmasked	$\mathcal{O}(dM)$
	masked	$\mathcal{O}(d\, nnz(\mathbf{m}))$
Column-based	unmasked	$\mathcal{O}(d\, nnz(\mathbf{f}) \log M)$
	masked	$\mathcal{O}(d\, nnz(\mathbf{f}) \log M)$

Table: Four sparse matvec variants and their associated cost, measured in terms of number of expected memory accesses into the sparse matrix $\mathbf{A}$ required.

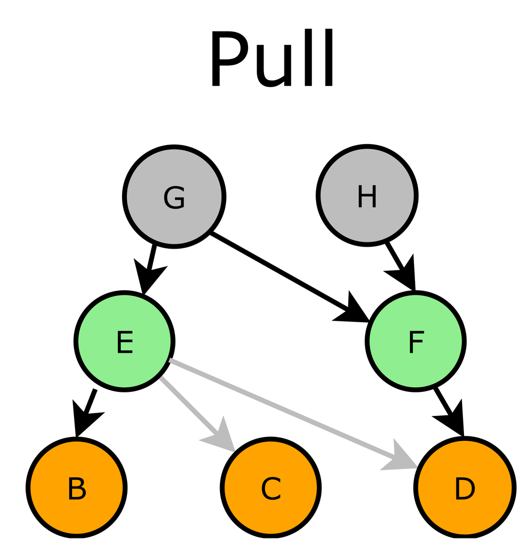
Direction-optimized BFS



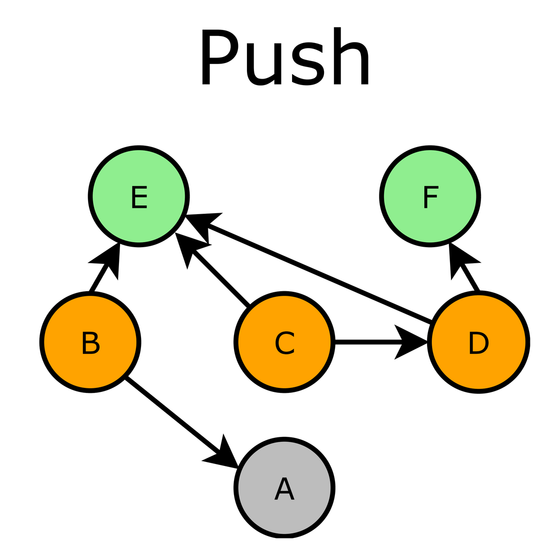
(a) Graphical representation.



(b) Linear algebra.



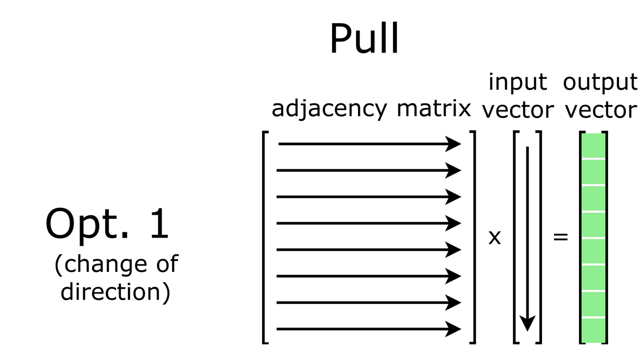
(c) Pull iteration.



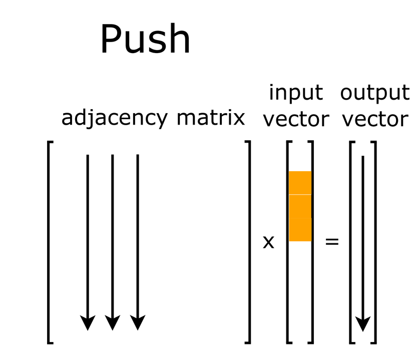
(d) Push iteration.

Figure: Simple example showing BFS traversal from the 3 nodes marked orange. There is a one-to-one correspondence between the graphical representation of both traversal strategies and their respective matvec equivalents on the figure to the right.

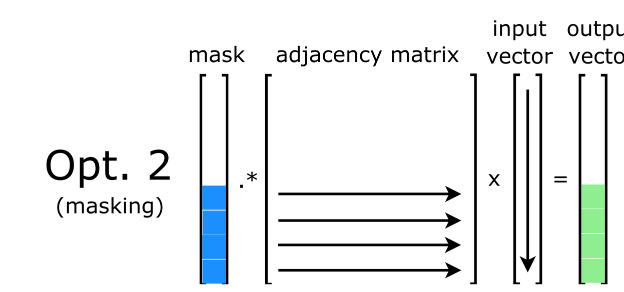
Optimizations



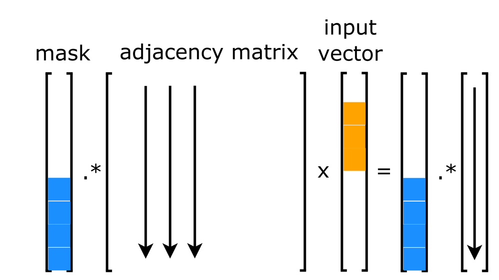
(a) Row-based matvec cannot use input or output sparsity.



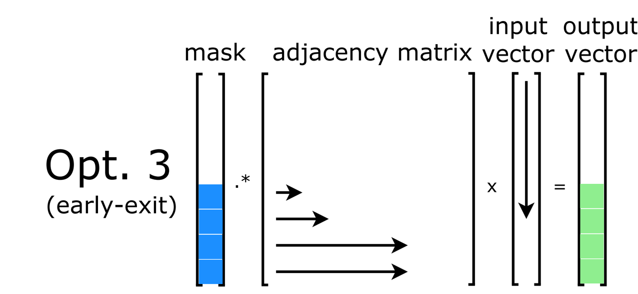
(b) Column-based matvec can use input sparsity.



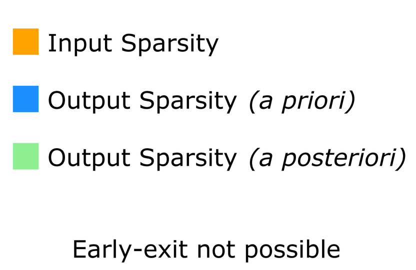
(c) Row-based masked matvec can use output sparsity.



(d) Column-based masked matvec can use input sparsity.



(e) Row-based masked matvec can use output sparsity and early-exit.



(f) Column-based masked matvec cannot early-exit.

Figure: The three optimizations known as "direction-optimized" BFS.

Experimental Results

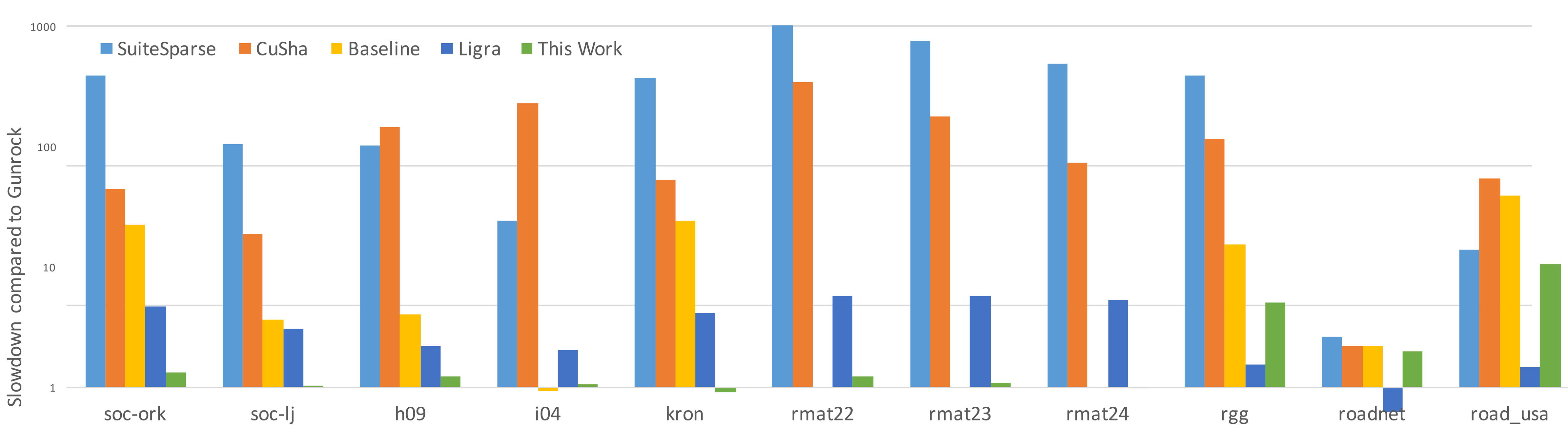
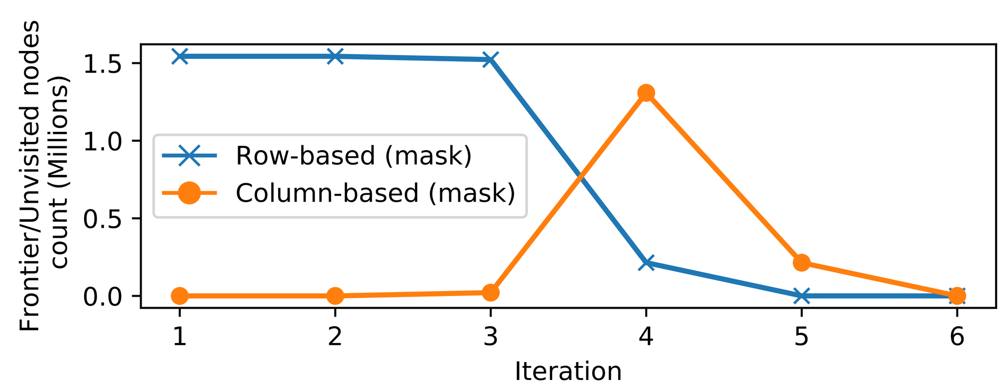


Figure: Comparison of our work to other graph libraries (SuiteSparse, CuSha, a baseline push-based BFS, Ligra, and Gunrock) implemented on 1× Intel Xeon 4-core E5-2637 v2 CPU and 1× NVIDIA Tesla K40c GPU.

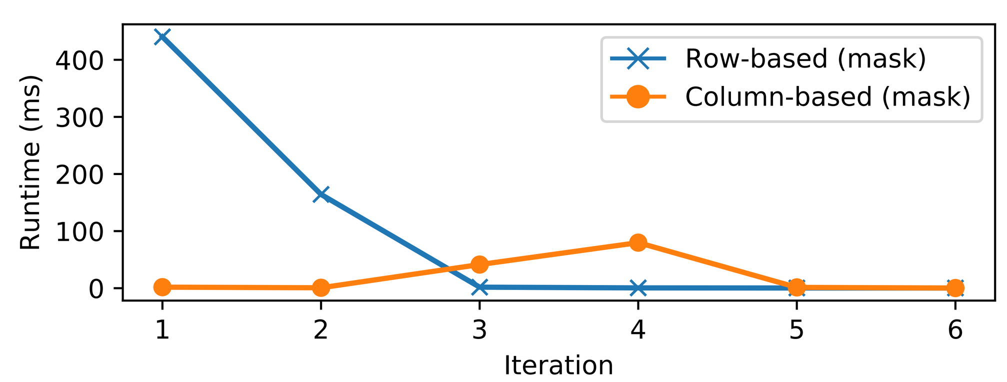
Conclusion

In this paper we demonstrate that push-pull corresponds to the concept of column- and row-based masked matvec. A possible future research direction would be to extend masking to other applications such as triangle counting and enumeration, adaptive PageRank, batched betweenness centrality, maximal independent set, and convolutional neural networks.

Direction-optimized Runtime



(a) Frontier count and unvisited node count.



(b) Push and pull runtime.

Figure: Breakdown of edge types in frontier during BFS traversal of Kronecker scale-21 graph (2M vertices, 182M edges).

Contact Information

- Web: http://www.ece.ucdavis.edu/∼ctcyang/
- Email: ctcyang@ucdavis.edu
- Phone: +1 (916) 802-8178