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

Carl Yang\textsuperscript{1,2}, Aydin Buluç\textsuperscript{2,3} and John D. Owens\textsuperscript{1}

\textsuperscript{1} Department of Electrical and Computer Engineering, UC Davis
\textsuperscript{2} Computational Research Division, LBNL
\textsuperscript{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

Direction-optimized BFS

Optimizations

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.

Complexity Results

<table>
<thead>
<tr>
<th>Operation</th>
<th>Expected Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row- unmasked</td>
<td>$O(dM)$</td>
</tr>
<tr>
<td>based masked</td>
<td>$O(d m n z(m))$</td>
</tr>
<tr>
<td>Column- unmasked</td>
<td>$O(d m n z(f) \log m n z(f))$</td>
</tr>
<tr>
<td>based masked</td>
<td>$O(d m n z(f) \log m n z(f))$</td>
</tr>
</tbody>
</table>

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

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.

Contact Information

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