Linear Algebra is the Right Way to Think About Graphs
Supercomputing Conference 2018: Doctoral Showcase

Carl Yang$^{1,3}$, Aydı̈n Buluç$^{2,3}$ and John D. Owens$^1$

$^1$ University of California, Davis
$^2$ University of California, Berkeley
$^3$ Lawrence Berkeley National Laboratory

November 13, 2018
Overview

1. Problem
2. Metric: Expressibility
3. Metric: Performance
4. Conclusion
Overview

1 Problem

2 Metric: Expressibility

3 Metric: Performance

4 Conclusion
Graph frameworks

CPU: Pregel, GraphLab, Cyclops, GraphX, Powergraph, GoFFish, Blogel, Gremlin, Haloop, Apache Giraph, Apache Hama, GPS, Mizan, Giraphx, Seraph, GiraphUC, Pregel+, Pregelix, Apache Tinkerpop, LFGraph, Gelly, Trinity, Ligra

GPU: Gunrock, Medusa, Totem, Frog, VertexAPI2, MapGraph, CuSha
What are the right primitives?

1. Portable
2. Concise
3. Expressible
4. High-performance
Linear algebra is the right way to think about graph algorithms.

Graph primitives based in linear algebra are superior to ones based on existing vertex-centric graph frameworks in portability, conciseness, expressibility and performance.
Graph traversal is sparse matrix multiplication

$G = (V, E)$

$A^T x \rightarrow A^T x$

\(^1\text{Denes Konig, 1931}\)
Ingredient 1: Operations

(a) eWiseAdd

(b) eWiseMult

(c) mxv

(d) mxm
Ingredient 2: Operators

Semiring notation: (Add, Multiply, Domain)
### Ingredient 2: Operators

Semiring notation: (Add, Multiply, Domain)

<table>
<thead>
<tr>
<th>Name</th>
<th>Semiring</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real field</td>
<td>${+, \times, \mathbb{R}}$</td>
<td>Classical numerical linear algebra</td>
</tr>
<tr>
<td>Boolean</td>
<td>${</td>
<td>, &amp;, {0, 1}}$</td>
</tr>
<tr>
<td>Tropical</td>
<td>${\min, +, \mathbb{R} \cup {\infty}}$</td>
<td>Shortest path</td>
</tr>
<tr>
<td>Max-plus</td>
<td>${\max, +, \mathbb{R}}$</td>
<td>Graph matching</td>
</tr>
<tr>
<td>Min-times</td>
<td>${\min, \times, \mathbb{R}}$</td>
<td>Maximal independent set</td>
</tr>
</tbody>
</table>
Operations + Operators = GraphBLAS

- **Miscellaneous:** connectivity, traversal (BFS), independent sets (MIS), graph matching
- **Centrality:** (PageRank, betweenness, closeness)
- **Graph clustering:** (Markov cluster, peer pressure, spectral, local)
- **Shortest paths:** (all-pairs, single-source, temporal)

GraphBLAS primitives in increasing arithmetic intensity

---

1. [http://graphblas.org](http://graphblas.org)
Metric: Portability

Open standard with clear C API specification ensures portability.
## Metric: Conciseness

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Ligra</th>
<th>Gunrock</th>
<th>Gunrock-GrB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Breadth-first-search</td>
<td>29</td>
<td>2732</td>
<td>25</td>
</tr>
<tr>
<td>Single-source shortest-path</td>
<td>55</td>
<td>760</td>
<td>25</td>
</tr>
<tr>
<td>Local graph clustering</td>
<td>84</td>
<td>875</td>
<td>45</td>
</tr>
</tbody>
</table>

**Table:** Lines of C++ application code counted by ‘cloc’.
Problem: Existing implementations not performant enough

Overview

1. Problem

2. Metric: Expressibility

3. Metric: Performance

4. Conclusion
Breadth First Search (BFS)

Breadth first search is a very important building block for other graph algorithms such as bipartite matching, maximum flow, strongly connected components, betweenness centrality, etc.
Example: Push BFS
Example: Push BFS
Problem: Many active vertices $\Rightarrow$ Slow

![Graph showing active vertices count over iterations](image)
Direction-optimized BFS

\[1\] Scott Beamer, Krste Asanovic, and David Patterson. ‘Direction-Optimizing Breadth-First Search’. SC ’12
Push is column-based mxv

Push

E → B → A
B → C → D
C → F
D → F
A → E
Push is column-based mxv

![Diagram of a graph with nodes A, B, C, D, E, and F connected by arrows.]

The adjacency matrix is used in the context of linear algebra to represent the connections between nodes in a graph. The equation $x = \begin{bmatrix} a \end{bmatrix}$ represents the output vector, where $x$ is the result of multiplying the adjacency matrix by the input vector $x$.
Pull is row-based mxv
Pull is row-based mxv

![Diagram of a graph with nodes G, H, E, F, B, C, and D connected by arrows.]

Mathematical representation:

\[
\begin{bmatrix}
\text{adjacency matrix}
\end{bmatrix}
\begin{bmatrix}
\text{input vector}
\end{bmatrix}
=
\begin{bmatrix}
\text{output vector}
\end{bmatrix}
\]
Where the speedup comes from

- Push-only baseline: 1.0x
- Structure only: 1.62x
- Change of direction: 1.08x
- Masking: 2.58x
- Early exit: 4.02x
- Operand reuse: 2.68x

Performance (GTEPS)
Overview

1 Problem

2 Metric: Expressibility

3 Metric: Performance

4 Conclusion
Experimental setup

Hardware:
- CPU: Intel 4-core E5-2637 v2 Xeon CPU @ 3.50GHz, 556GB RAM
- GPU: NVIDIA K40c, 12GB RAM

Datasets:
- scale-free graphs up to 16.8M vertices, 520M edges
- road network graphs up to 23.9M vertices, 577M edges
Comparable performance to state-of-the-art on scale-free graphs
GraphBLAS has closed the 10× performance gap
Overview

1. Problem
2. Metric: Expressibility
3. Metric: Performance
4. Conclusion
Conclusion

- Our work is the first GraphBLAS implementation on the GPU that shows comparable performance to state-of-the-art
  - 10× to 1000× speed-up compared to other GraphBLAS implementations
- Linear algebra-based graph frameworks are the way to go:
  - Just as expressible and performant as their vertex-centric counterparts
  - More concise and portable
- Future work: Move more decisions to backend
- Future work: Distributed memory implementation
Acknowledgments

- My advisors, John D. Owens and Aydı̇n Buluç
- The Gunrock team
- Scott McMillan
- NVIDIA
  - for their generous hardware support
- Funding support
  - NSF Award # CCF-1629657
  - DARPA XDATA program under US Army award W911QX-12-C-0059
  - DARPA HIVE program under agreement number FA8650-18-2-7836
  - LBNL’s DOE contract DE-AC02-05CH11231
  - DOE’s OASCR contract DE-AC02-05CH11231
  - NNSA and DOE’s Office of Science under the Exascale Computing Project 17-SC-20-SC
Questions?

Open-source (Apache 2.0) code:
https://github.com/gunrock/gunrock-grb