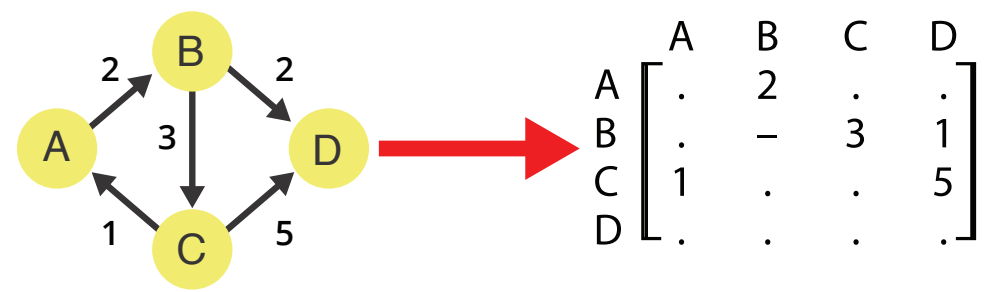


Spiral/AIML: Frontiers of Graph Processing in Linear Algebra

Introduction

Graph algorithms can be expressed as sequences of linear, algebra-like operations through the use of the adjacency matrix. Adjacency matrices are used to represent graphs instead of vertices and edges.



Our work extends the use of linear algebra beyond simple graph traversal.

Writing Graph Algorithms in LA

Many graph primitives can be cast in terms of LA operations.

- Neighbors of a vertex v —vector matrix products.
- Filtering—Hadamard products.
- Semi-rings—to represent Matrix Multiplication like operation (Single-Source Shortest Path uses the min-plus semiring).

This mapping gets rid of the need for “experts” to formulate graph algorithms in LA.

- Formally derived algorithms

Families of Graph Algorithms

Multiple graph algorithms can be enumerated for the same specification.

Our LA approach uses triangle counting:

$$\Delta = \frac{1}{6} \Gamma(A^3)$$

This approach allows us to analyze graph algorithms for individual performance characteristics. The algorithm that best-fit the situation is chosen.

Our **linear algebraic approach** to graph algorithms provides a **flexible framework** that enables high performance code generation for **faster network analysis**.

Writing Graph Algorithms in LA

Operation

Edge Filtering

Linear Algebra

$$A_b = A > \Delta$$

$$A_H = A_b \circ A$$

Vertex bucketing

$$v_{Bi} = (\Delta i \leq tent_{dists} < \Delta(i+1))$$

$$v'_{Bi} = ((\Delta i \leq new_{dist} < \Delta(i+1)) \circ (new_{dist} < old_{dist}))$$

Neighborhood Traversal with Relaxation

$$t_{Bi} = (tent_{dists} \ v_{Bi})$$

$$y_{out}^T = t_{Bi}(\min.+) \begin{bmatrix} . & 2 & . & . \\ . & . & 3 & 2 \\ 1 & . & . & 5 \\ . & . & . & . \end{bmatrix}$$

Busting Myths about Linear Algebraic Approach

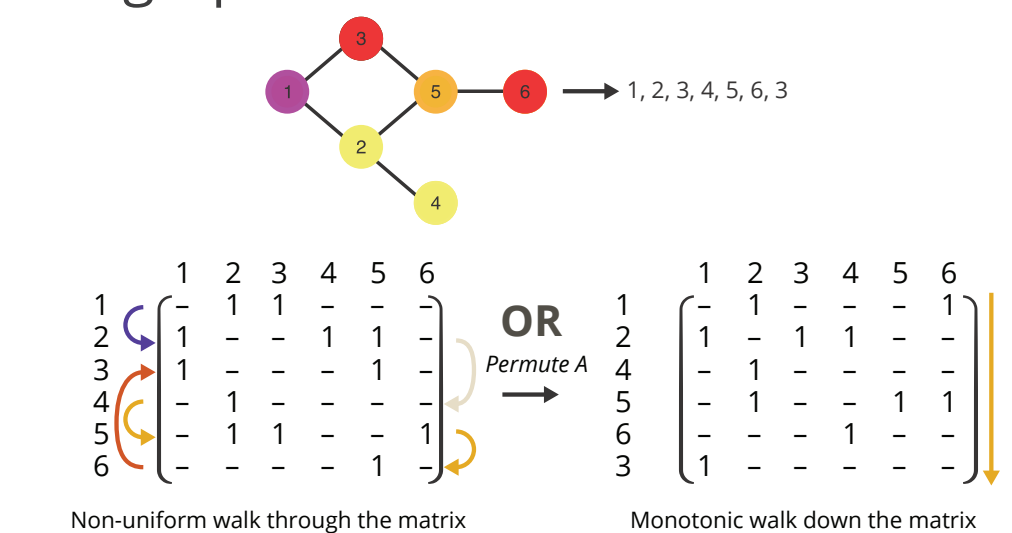
Unifying Edge and Vertex Centric Algorithms

$$\begin{pmatrix} A_{00} & a_{01} & A_{02} \\ * & 0 & a_{T12} \\ * & * & A_{22} \end{pmatrix} \quad \begin{pmatrix} A_{00} & a_{01} & A_{02} \\ * & 0 & a_{T12} \\ * & * & A_{22} \end{pmatrix}$$

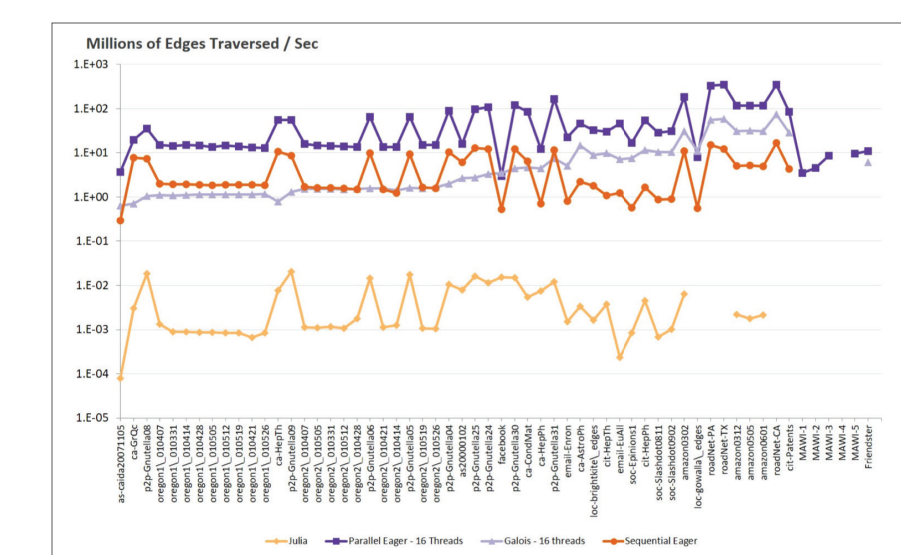
Both classes of algorithms can be expressed using a single linear algebraic framework.

Depth-First Search in Linear Algebra

First look at expressing depth-first traversal of graphs in LA.



Linear Algebraic Approach can Yield Good Performance



2019 Graph Challenge Champion

2018 Graph Challenge Finalist

2017 Graph Challenge Honorable Mention

References

- Low, Tze Meng, et al. “First look: Linear algebra-based triangle counting without matrix multiplication.” 2017 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, 2017.
- Lee, Matthew, and Tze Meng Low. “A family of provably correct algorithms for exact triangle counting.” Proceedings of the First International Workshop on Software Correctness for HPC Applications. ACM, 2017.
- Low, Tze Meng, et al. “Linear Algebraic Formulation of Edge-centric K-truss Algorithms with Adjacency Matrices.” 2018 IEEE High Performance Extreme Computing Conference (HPEC). IEEE, 2018.
- Sridhar, Upasana, et al. “Delta-stepping SSSP: from vertices and edges to GraphBLAS implementations.” 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2019.
- Spampinato, Daniele G., Upasana Sridhar, and Tze Meng Low. 2019. “Linear algebraic depth-first search.” In Proceedings of the 6th ACM SIGPLAN International Workshop on Libraries, Languages and Compilers for Array Programming (ARRAY 2019). ACM, New York, NY, USA, 93-104. DOI: <https://doi.org/10.1145/3315454.3329962>

Copyright 2019 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.

[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

Internal use:* Permission to reproduce this material and to prepare derivative works from this material for internal use is granted, provided the copyright and "No Warranty" statements are included with all reproductions and derivative works.

External use:* This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other external and/or commercial use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

* These restrictions do not apply to U.S. government entities.

Carnegie Mellon® is registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

DM19-1045