

Butterfly Enumeration in Bipartite Graphs using GraphBLAS

17 January 2020

Scott McMillan

Emerging Technology Center - PNW
Software Engineering Institute
Carnegie Mellon University



Copyright 2020 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.

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 use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

DM20-0037

GraphBLAS References

Mathematical Foundations of the GraphBLAS

IEEE HPEC 2016

Jeremy Kepner (MIT Lincoln Laboratory Supercomputing Center), Peter Aaltonen (Indiana University), David Bader (Georgia Institute of Technology), Aydın Buluç (Lawrence Berkeley National Laboratory), Franz Franchetti (Carnegie Mellon University), John Gilbert (University of California, Santa Barbara), Dylan Hutchison (University of Washington), Manoj Kumar (IBM), Andrew Lumsdaine (Indiana University), Henning Meyerhenke (Karlsruhe Institute of Technology), Scott McMillan (CMU Software Engineering Institute), Jose Moreira (IBM), John D. Owens (University of California, Davis), Carl Yang (University of California, Davis), Marcin Zalewski (Indiana University), Timothy Mattson (Intel)

Design of the GraphBLAS API for C

IEEE HPEC 2017

Aydın Buluç[†], Tim Mattson[‡], Scott McMillan[§], José Moreira[¶], Carl Yang^{*,†}

[†]*Computational Research Division, Lawrence Berkeley National Laboratory*

[‡]*Intel Corporation* [§]*Software Engineering Institute, Carnegie Mellon University* [¶]*IBM Corporation*

^{*}*Electrical and Computer Engineering Department, University of California, Davis, USA*

GraphBLAS Ecosystem

C++ Algorithms
Repository
Carnegie
Mellon
University

LAGraph
C algorithms
repository

redisgraph
redislabs

Carnegie
Mellon
University
GraphBLAS
Test Framework

pggraphblas
PostgreSQL Wrapper

pygraphblas
Python Wrapper

pyGB
Python Wrapper
around gbl
Carnegie
Mellon
University

GraphBLAS C API, v. 1.3.0
GraphBLAS C++ API (in progress)

Multithreaded
SuiteSparse

IBM-GraphBLAS

Galois GB
TEXAS
The University of Texas at Austin

Distributed gbl
Lawrence Livermore
National Laboratory

gbl
Carnegie
Mellon
University

Pacific Northwest
NATIONAL LABORATORY

Proudly Operated by Battelle Since 1965

GraphBLAST (GPU)

UC DAVIS
UNIVERSITY OF CALIFORNIA

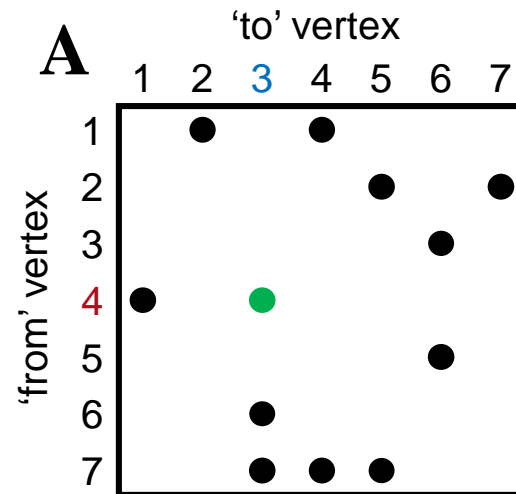
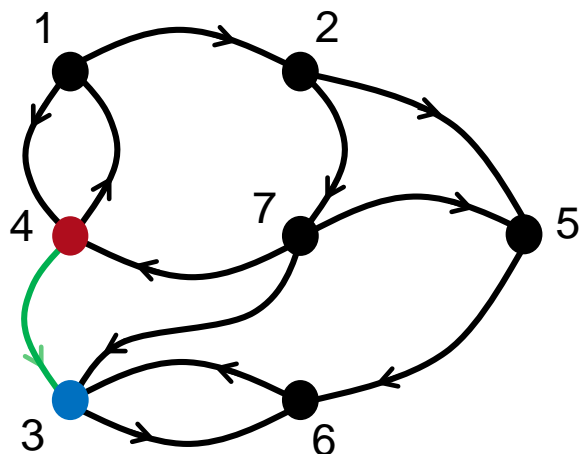


GraphBLAS
on EMU

Carnegie
Mellon
University

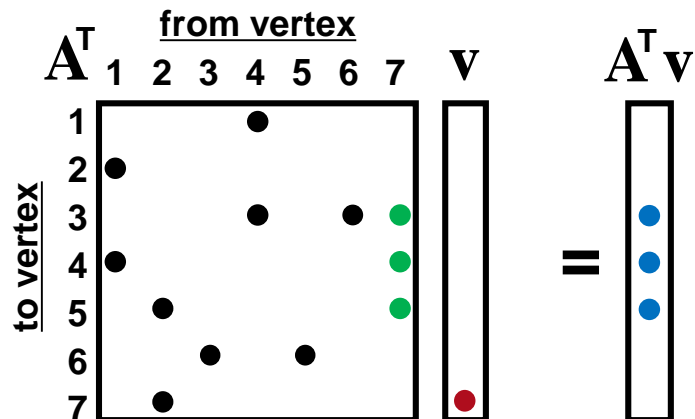
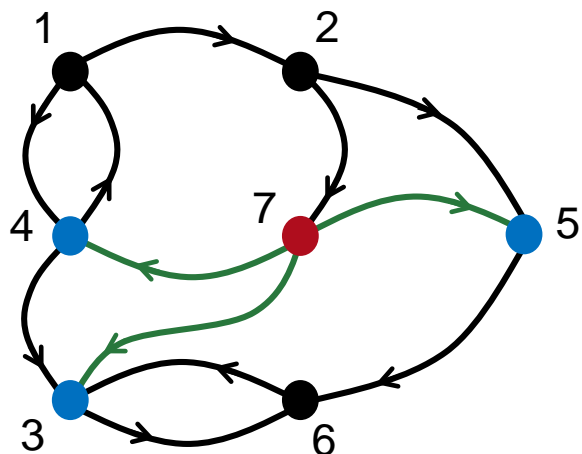
UMBC
UNIVERSITY OF MARYLAND

Graphs as Matrices



Graphs are represented as adjacency matrices that have *irregular* and *sparse* structure.

Graph Operations as Matrix Operations



- Matrix multiply \rightarrow find neighbors (most important primitive)
- Used in breadth-first traversal, shortest paths, and many others
- Sparsity and irregularity of matrix structure is a barrier to high performance

K-truss Enumeration

Find all edges in graph (**A**) that are part of at least $k-2$ (the support) unique triangles.

converged \leftarrow false

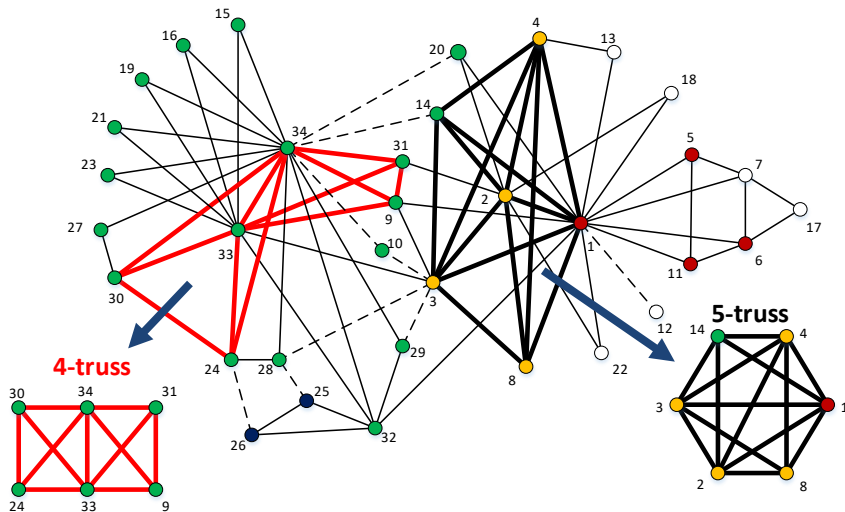
while !*converged* do:

S \langle **A** $\rangle = (\mathbf{A}^T \oplus . \otimes \mathbf{A})$ // 1: support

M = **S** \cdot_{\geq} ($k - 2$) // 2: prune

A \langle **M** $\rangle = \mathbf{M} \circ \mathbf{A}$

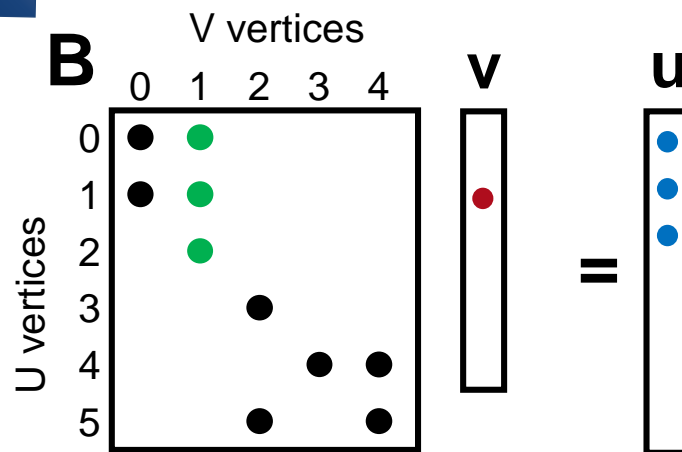
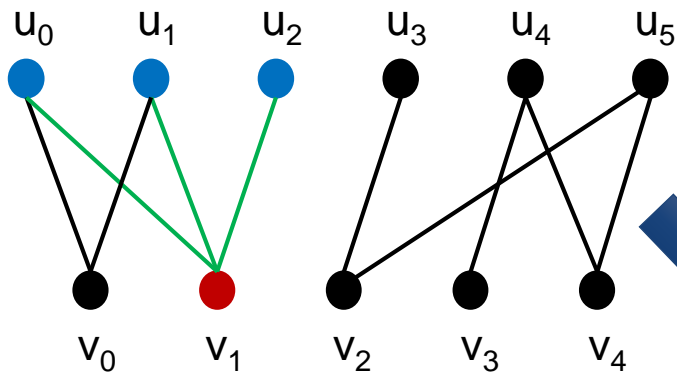
converged \leftarrow isUnchanged(**M**)



Matrix `k_truss(Matrix &A, int k)`

```
{
  Matrix S(A.nrows(), A.ncols());
  mxm(S, A, NoAccum(), ArithmeticSemiring<int>(),
      transpose(A), A);
  apply(M, NoMask(), NoAccum(), GreaterThan<int>(k-2), S);
  apply(A, NoMask(), NoAccum(), Identity<int>(), A);
  return A;
}
```

Bipartite Graphs as Matrices



Butterfly Enumeration

Find all edges in a bipartite graph (\mathbf{B}) that are part of at least $f(k-2)$ unique butterflies.

Given: \mathbf{B} ($|U| \times |V|$ binary sparse matrix), and threshold k .

Step 1: Induce edges within U and V to create triangles

- Create edges in \mathbf{A}_U b/w vertices w/in U that share a common neighbor in V .
- Create edges in \mathbf{A}_V b/w vertices w/in V that share a common neighbor in U .

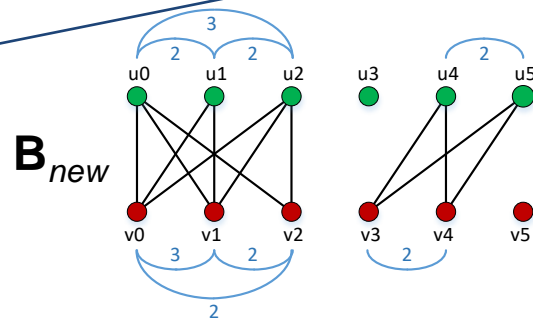
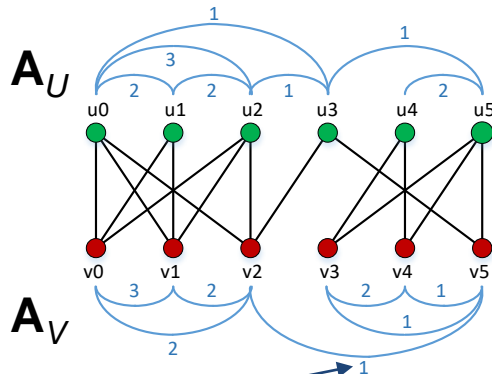
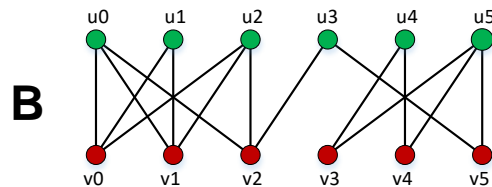
Step 2: Compute the triangle support for edges in \mathbf{A}_U and \mathbf{A}_V

- Hint: support equals the number common neighbors

Step 3: Remove the edges in \mathbf{A}_U and \mathbf{A}_V that do not have “enough support”

Step 4: Compute the new Bipartite Graph (\mathbf{B}_{new}) only consisting of edges that are part of triangles

Repeat the process with \mathbf{B}_{new} larger k until no edges are left.



Butterfly Enumeration

Given: \mathbf{B} ($|U| \times |V|$ binary sparse matrix), and threshold k .

Step 1: Induce edges to create triangles

- $\mathbf{A}_U \langle -I_{|U|} \rangle = \mathbf{B} \oplus \otimes \mathbf{B}^T$ (boolean semiring)
- $\mathbf{A}_V \langle -I_{|V|} \rangle = \mathbf{B}^T \oplus \otimes \mathbf{B}$

Step 2: Compute the triangle support for edges in \mathbf{A}_U and \mathbf{A}_V : \mathbf{S}_U and \mathbf{S}_V

- $\mathbf{S}_U \langle \mathbf{A}_U \rangle = \mathbf{B} \oplus \otimes \mathbf{B}^T$ (arithmetic semiring)
- $\mathbf{S}_V \langle \mathbf{A}_V \rangle = \mathbf{B}^T \oplus \otimes \mathbf{B}$

Step 3: Remove the edges in \mathbf{A}_U and \mathbf{A}_V that do not have “enough support”

- $\mathbf{A}_U \langle \mathbf{M}_U \rangle = \mathbf{A}_U$, where $\mathbf{M}_U = \mathbf{S}_U \geq (k - 2)$
- $\mathbf{A}_V \langle \mathbf{M}_V \rangle = \mathbf{A}_V$, where $\mathbf{M}_V = \mathbf{S}_V \geq (k - 2)$

Step 4: Compute the new Bipartite Graph (\mathbf{B})

- $\mathbf{B}_{new} \langle \mathbf{B} \rangle = (\mathbf{A}_U \oplus \otimes \mathbf{B}) \oplus (\mathbf{B} \oplus \otimes \mathbf{A}_V)$ (boolean semiring)

Repeat with \mathbf{B}_{new} and larger k , until no edges remain.

