The GraphBLAS Standard Effort

Jeremy Kepner (MIT Lincoln Laboratory)

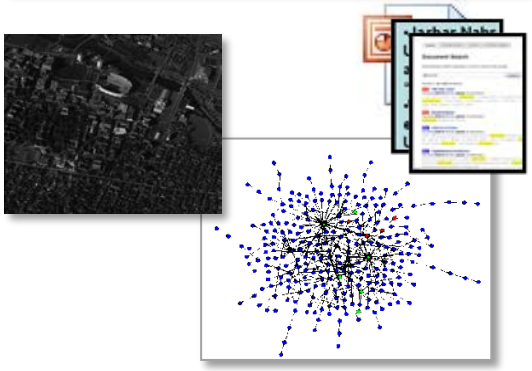
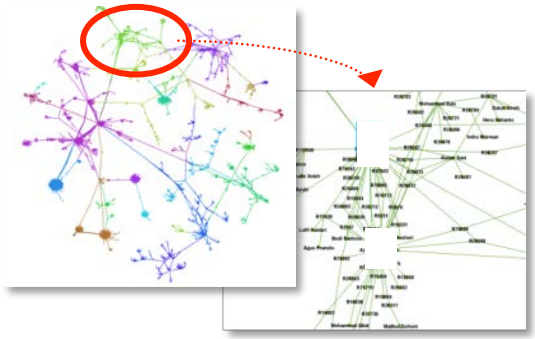
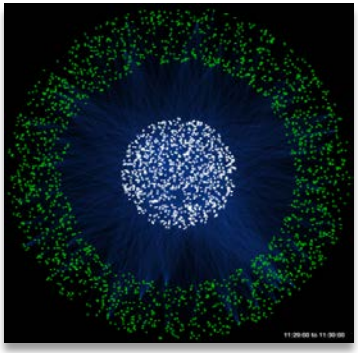
David Bader (GA Tech), Aydın Buluç (LBNL), John Gilbert (UCSB), Joseph Gonzalez (UCB), Tim Mattson (Intel)

*This work is sponsored by the Assistant Secretary of Defense for Research and Engineering under Air Force Contract #FA8721-05-C-0002. Opinions, interpretations, recommendations and conclusions are those of the authors and are not necessarily endorsed by the United States Government.

Outline

- Introduction
- Approach
- GraphBLAS
- Summary

Example Applications of Graph Analytics

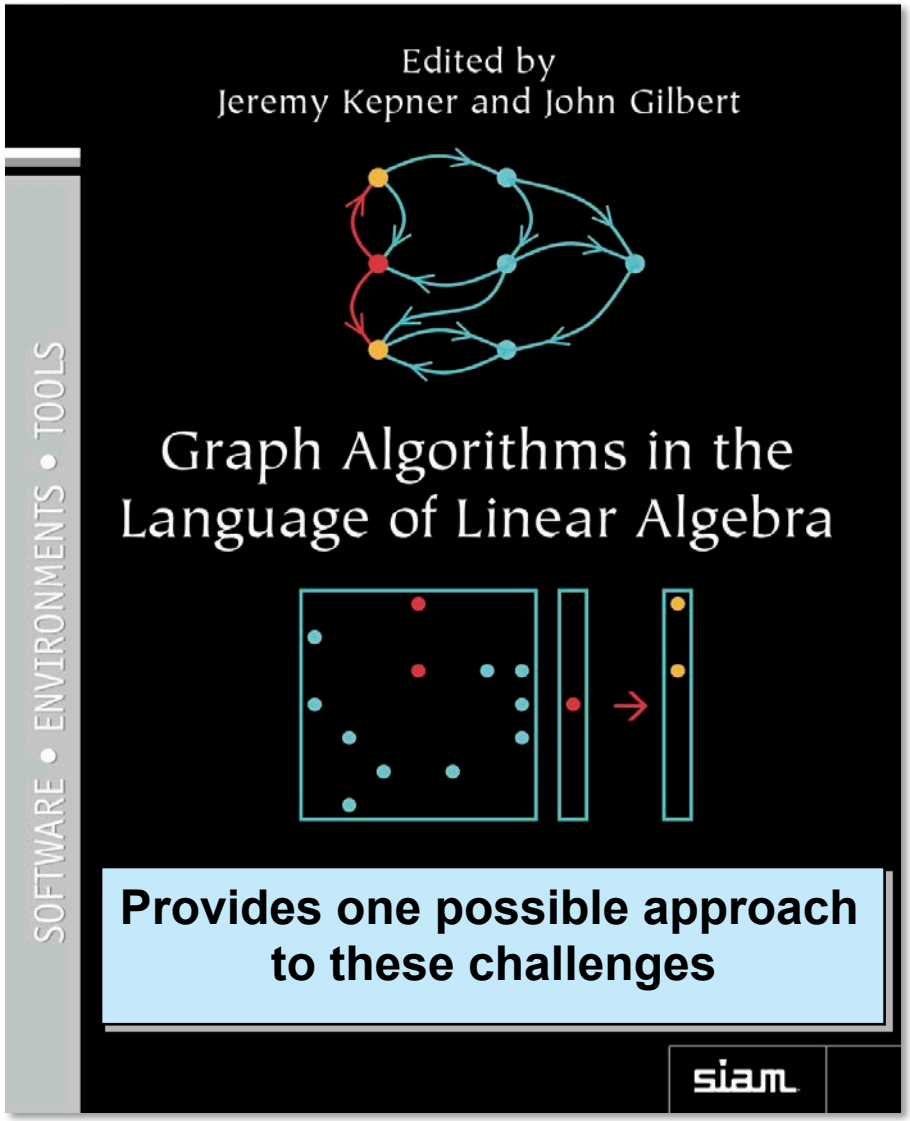
ISR	Social	Cyber
		
• Graphs represent entities and relationships detected through multi-INT sources • 1,000s – 1,000,000s tracks and locations • GOAL: Identify anomalous patterns of life	• Graphs represent relationships between individuals or documents • 10,000s – 10,000,000s individual and interactions • GOAL: Identify hidden social networks	• Graphs represent communication patterns of computers on a network • 1,000,000s – 1,000,000,000s network events • GOAL: Detect cyber attacks or malicious software

Cross-Mission Goal: Detection of subtle patterns in massive graphs

Graph Computation Challenges

- **Software complexity**
  - Typical graph library has hundreds of functions
- **Data Complexity**
  - Graphs often require custom databases or custom schemas
- **Security**
  - Standard graph analysis on encrypted data does currently exist
- **Mathematical complexity**
  - Lack composibility: commutivity, distributivity, and associativity
- **Theory**
  - Standard graph combinatorics has been thoroughly explored
- **Serial processing performance**
  - 1000x less efficient than dense computations
- **Parallel processing performance**
  - Efficiency decreases as $1/\sqrt{P}$

Graph Algorithms in the Language of Linear Algebra (GALA)



- Prof. David Bader (Georgia Tech)
- Ms. Nadya Bliss (Arizona State)
- Mr. Robert Bond (MIT)
- Dr. Aydin Buluc (LBL)
- Dr. Daniel Dunleavy (Sandia)
- Prof. Alan Edelman (MIT)
- Prof. Christos Faloutsos (CMU)
- Prof. Jeremy Fineman (Georgetown)
- Prof. John Gilbert (UCSB)
- Prof. Christine Heitsch (Georgia Tech)
- Dr. Bruce Hendrickson (Sandia)
- Dr. W. Philip Kegelmeyer (Sandia)
- Dr. Jeremy Kepner (MIT)
- Dr. Tammy Kolda (Sandia)
- Prof. Jure Leskovec (Stanford)
- Prof. Kamesh Maduri (Penn State)
- Dr. Sanjeev Mohindra (MIT)
- Mr. Huy Nguyen (MIT)
- Mr. Charlie Rader (MIT)
- Dr. Steve Reinhardt (Yarc)
- Dr. Eric Robinson (Google)
- Dr. Viral Shah (Julia Foundation)

Software/Algorithm Complexity

Algorithm (Problem)	Canonical Complexity	LA-Based Complexity	Critical Path (for LA)
Breadth-first search	$\Theta(m)$	$\Theta(m)$	$\Theta(diameter)$
Betweenness Centrality (unweighted)	$\Theta(mn)$	$\Theta(mn)$	$\Theta(diameter)$
All-pairs shortest-paths (dense)	$\Theta(n^3)$	$\Theta(n^3)$	$\Theta(n)$
Prim (MST)	$\Theta(m+n \log n)$	$\Theta(n^2)$	$\Theta(n)$
Borůvka (MST)	$\Theta(m \log n)$	$\Theta(m \log n)$	$\Theta(\log^2 n)$
Edmonds-Karp (Max Flow)	$\Theta(m^2n)$	$\Theta(m^2n)$	$\Theta(mn)$
Greedy MIS (MIS)	$\Theta(m+n \log n)$	$\Theta(mn+n^2)$	$\Theta(n)$
Luby (MIS)	$\Theta(m+n \log n)$	$\Theta(m \log n)$	$\Theta(\log n)$

Many graph algorithms can be represented with linear algebra (LA) using just a handful of functions.

($n = |V|$ and $m = |E|$)

Sparse Arrays work Naturally with Modern Key/Value Databases (Accumulo)

Input Data

Time	Col1	Col2	Col3
2001-01-01	a		a
2001-01-02	b	b	
2001-01-03		c	c

Accumulo Table: `Ttranspose`

	01-01-2001	02-01-2001	03-01-2001
Col1|a	1		
Col1|b		1	
Col2|b		1	
Col2|c			1
Col3|a	1		
Col3|c			1

Accumulo Table: `T`

	Col1|a	Col1|b	Col2|b	Col2|c	Col3|a	Col3|c
01-01-2001	1				1	
02-01-2001		1	1			
03-01-2001				1		1

- **Tabular data expanded to create many type/value columns**
- **Transpose pairs allows quick look up of either row or column**

D4M 2.0 Schema: A General Purpose High Performance Schema for the Accumulo Database, Kepner et al, IEEE HPEC 2013

Security: Computing on Masked Data

Plaintext Input

log_id	src_ip	dest_ip
001	128.0.0.1	208.29.69.138
002	192.168.1.2	157.166.255.18
003	128.0.0.1	74.125.224.72

Deterministic Encryption DET_{row}

Order Preserving Encryption OPE_{col}

	src_ip|128.0.0.1	src_ip|192.168.1.2	dest_ip|157.166.255.18	dest_ip|208.29.69.138	dest_ip|74.125.224.72
log_id|001	1			1	
log_id|002		1	1		
log_id|003	1			1	1

	BGDJBEAB…	PJDMJPCGG…	QLHNLRJKG…	RSTPWRQQI…	SWVUZZVZJ…
EQKRP…	SKASEMIC			SKASEMIC	
BYZZO…		SKASEMIC	SKASEMIC		
CJYTG…	SKASEMIC			SKASEMIC	SKASEMIC

Masked Data Equivalent

- Sparse table holds data masked by appropriate encryption scheme
- Algebra works with DET or OPE on rows or columns by definition

Computing on Masked Data: a High Performance Method for Improving Big Data Veracity
Gadepally et al, IEEE HPEC 2014

Composable Graph Algorithms

- **Key innovation: mathematical closure**
  - **All array operations return arrays**
  - **Provides algebraic commutativity, associativity, and distributivity**

- **Enables composable mathematical operations**

```
A + B      A - B      A & B      A|B      A*B
```

- **Enables composable query operations via array indexing**

```
A('alice bob ',:)     A('alice ',:)     A('al* ',:)
A('alice : bob ',:)   A(1:2,:)          A == 47.0
```

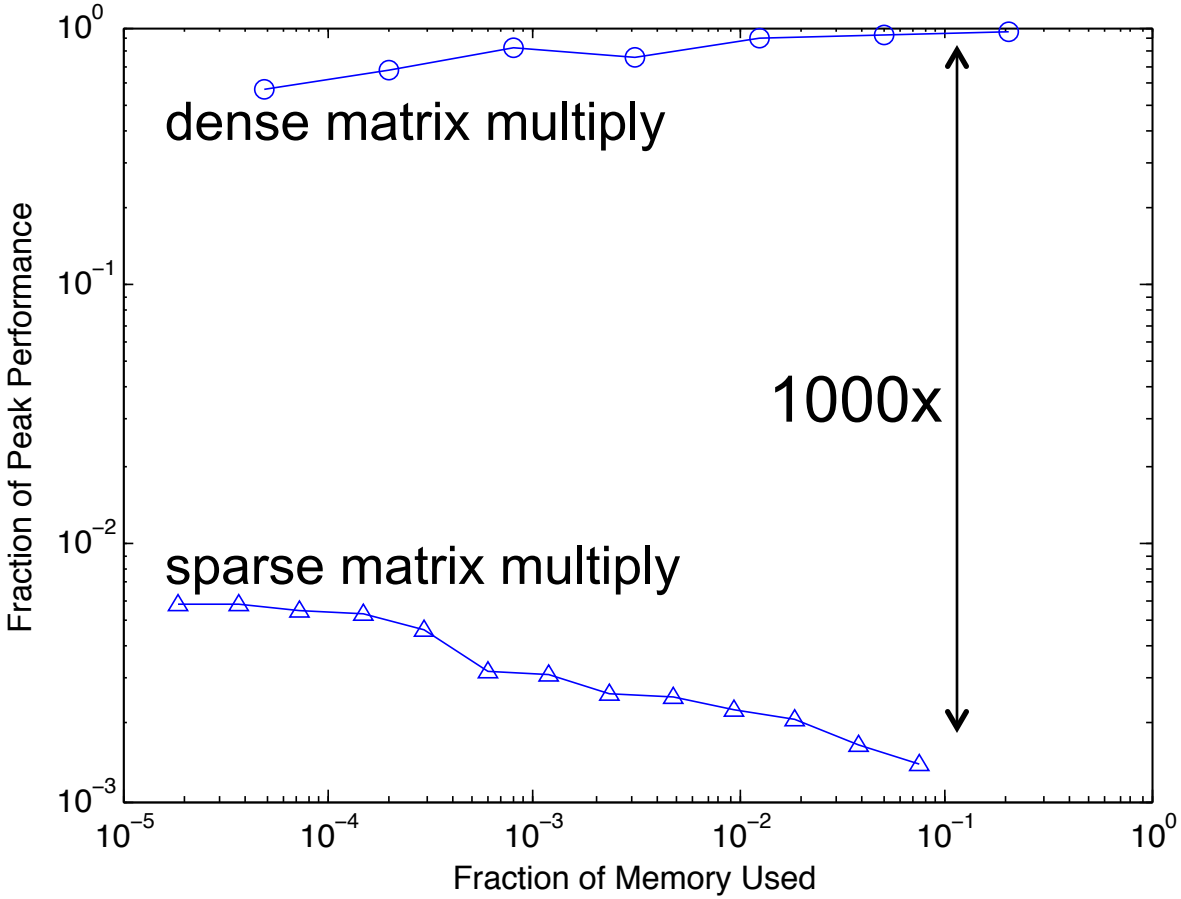
- **Complex queries with ~50x less effort than Java/SQL**

The Abstract Algebra of Big Data and Associative Arrays
Chaidez et al, SIAM Discrete Math 2014

Matrix Based Graph Theory

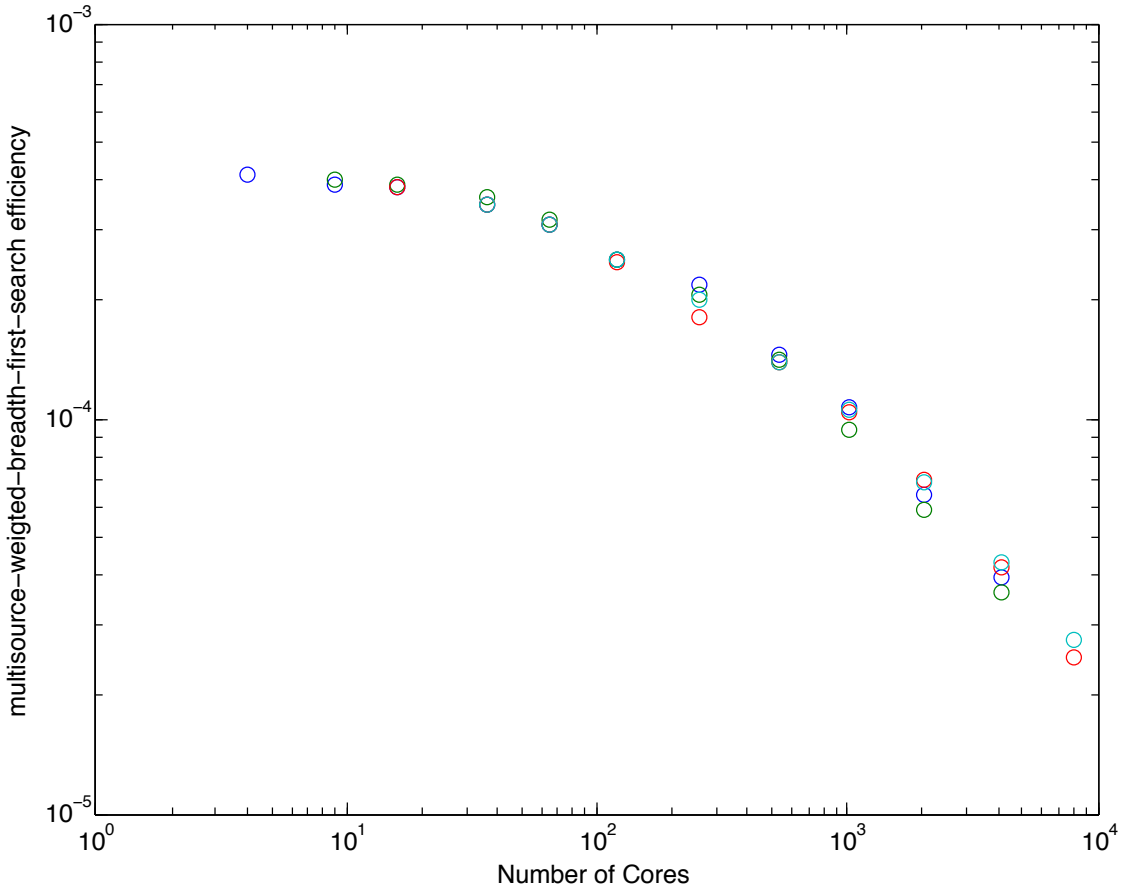
- *Navigating Central Path with Electrical Flows: from Flows to Matchings, and Back*, Madry et al, FOCS 2013, **Best Paper Award**
- *From Graphs to Matrices, and Back: New Techniques for Graph Algorithms, Ph.D. thesis*, Madry, MIT, EECS Department, 2011, **George M. Sprowls Award** (for best MIT doctoral theses in CS).
- *A Polylogarithmic-Competitive Algorithm for the k-Server Problem*, Madry et al, FOCS 2011, **Best Paper Award**
- *Electrical Flows, Laplacian Systems, and Faster Approximation of Maximum Flow in Undirected Graphs*, Madry et al, STOC 2011, **Best Paper Award**
- *An O(log n/log log n)-approximation Algorithm for the Asymmetric Traveling Salesman Problem*, Madry et al, SODA 2010, **Best Paper Award**
- **Older work by some other folks:** *The anatomy of a large-scale hypertextual Web search engine*, Brin & Page, Computer Networks and ISDN Systems 1998, **Most Money Ever Made Award**

Serial Processing Performance



- **Large gap between dense BLAS and graph performance**
- **GraphBLAS provide a good target for processor optimization**

Parallel Processing Performance



- **Parallel graph processing efficiency decreases as $1/\sqrt{P}$**
- **GraphBLAS provide a good target for system optimization**

Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments
Buluc & Gilbert, SIAM Journal on Scientific Computing 2012

Outline

- Introduction
- Approach
- GraphBLAS
- Summary

Can we standardize a "Graph BLAS"?

No, it's not reasonable to define a universal set of building blocks.

Huge diversity in matching graph algorithms to hardware platforms.
No consensus on data structures or linguistic primitives.
Lots of graph algorithms remain to be discovered.
Early standardization can inhibit innovation.

Yes, it *is* reasonable to define a common set of building blocks…
… for graphs as linear algebra.

Representing graphs in the language of linear algebra is a mature field.
Algorithms, high level interfaces, and implementations vary.
But the core primitives are well established.

The Graph BLAS effort

Standards for Graph Algorithm Primitives

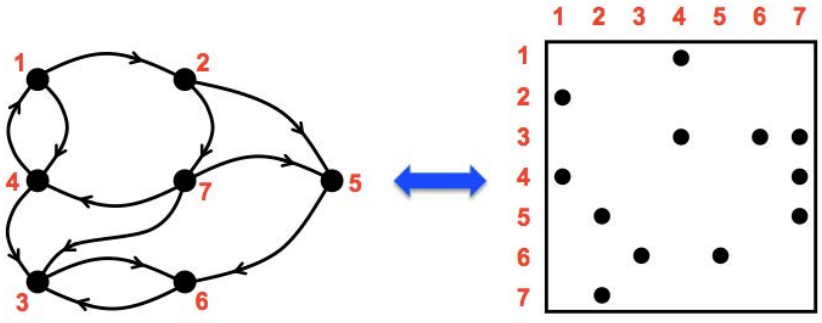
Tim Mattson (Intel Corporation), David Bader (Georgia Institute of Technology), Jon Berry (Sandia National Laboratory), Aydin Buluc (Lawrence Berkeley National Laboratory), Jack Dongarra (University of Tennessee), Christos Faloutsos (Carnegie Melon University), John Feo (Pacific Northwest National Laboratory), John Gilbert (University of California at Santa Barbara), Joseph Gonzalez (University of California at Berkeley), Bruce Hendrickson (Sandia National Laboratory), Jeremy Kepner (Massachusetts Institute of Technology), Charles Leiserson (Massachusetts Institute of Technology), Andrew Lumsdaine (Indiana University), David Padua (University of Illinois at Urbana-Champaign), Stephen Poole (Oak Ridge National Laboratory), Steve Reinhardt (Cray Corporation), Mike Stonebraker (Massachusetts Institute of Technology), Steve Wallach (Convey Corporation), Andrew Yoo (Lawrence Livermore National Laboratory)

Abstract-- It is our view that the state of the art in constructing a large collection of graph algorithms in terms of linear algebraic operations is mature enough to support the emergence of a standard set of primitive building blocks. This paper is a position paper defining the problem and announcing our intention to launch an open effort to define this standard.

- The Graph BLAS Forum: http://istc-bigdata.org/GraphBlas/
- Graph Algorithms Building Blocks (GABB workshop at IPDPS'14): http://www.graphanalysis.org/workshop2014.html

Combinatorial BLAS

http://gauss.cs.ucsb.edu/~aydin/CombBLAS



An extensible distributed-memory library offering a small but powerful set of linear algebraic operations specifically targeting graph analytics.

- **Aimed at graph algorithm designers/programmers who are not expert in mapping algorithms to parallel hardware.**
- **Flexible templated C++ interface; 2D data decomposition**
- **Scalable performance from laptop to 100,000-processor HPC.**
- **Open source software (v1.4.0 released January, 2014)**

Sparse array attribute survey

Function	Graph BLAS	Comb BLAS	Sparse BLAS	STINGER	D4M	SciDB	Tensor Toolbox	Julia	GraphLab
Version		1.3.0	2006	r633	2.5	13.9	2.5	0.2.0	2.2
Language	any	C++	F,C,C++	C	Matlab	C++	Matlab, C++	Julia	C++
Dimension	2	1, 2	2	1, 2, 3	2	1 to 100	2, 3	1,2	2
Index Base	0 or 1	0	0 or 1	0	1	±N	1	1	0
Index Type	uint64	uint64	int	int64	double, string	int64	double	any int	uint64
Value Type	?	user	single, double, complex	int64	logical, double, complex, string	user	logical, double, complex	user	user
Null	0	user	0	0	≤0	null	0	0	int64(-1)
Sparse Format	?	tuple	undef	linked list	dense, csc, tuple	RLE	dense, csc	csc	csr/csc
Parallel	?	2D block	none	block	arbitrary	N-D block, cyclic w/ overlap	none	N-D block, cyclic w/ overlap	Edge based w/ vertex split
+ operations *** operations**	user? user?	user user	+ *	user user	+,*,max,min,∩,∪	user user	+ *	user user	user user

Semirings in graph algorithms

Real field: **(R, +, x)**	Classical numerical linear algebra
Boolean algebra: **({0 1}, |, &)**	Graph traversal
Tropical semiring: **(R U {∞}, min, +)**	Shortest paths
(S, select, select)	Select subgraph, or contract nodes to form quotient graph
(edge/vertex attributes, vertex data aggregation, edge data processing)	Schema for user-specified computation at vertices and edges
(R, max, +)	Graph matching &network alignment
(R, min, times)	Maximal independent set

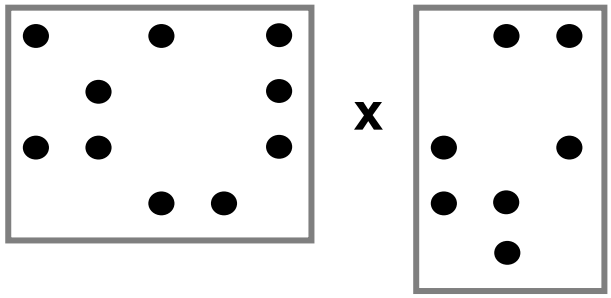
- **Shortened semiring notation: (Set, Add, Multiply)**. Both identities omitted.
- **Add:** Traverses edges, **Multiply:** Combines edges/paths at a vertex
- Neither add nor multiply needs to have an inverse.
- Both **add** and **multiply** are **associative** ,**multiply distributes** over **add**

Outline

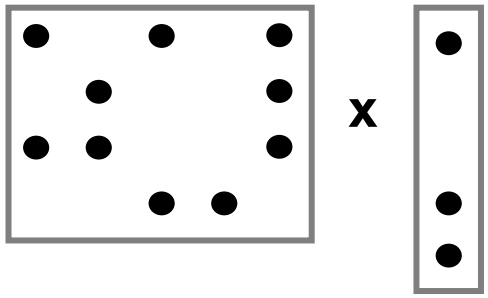
- Introduction
- Approach
- GraphBLAS
- Summary

Linear-algebraic primitives for graphs

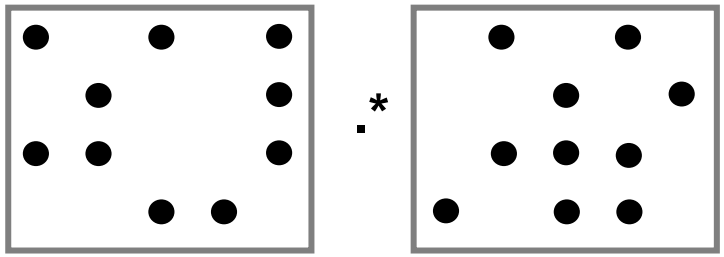
Sparse matrix-sparse matrix multiplication



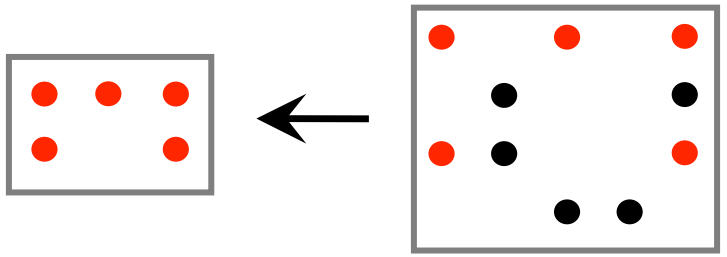
Sparse matrix-sparse vector multiplication



Element-wise operations

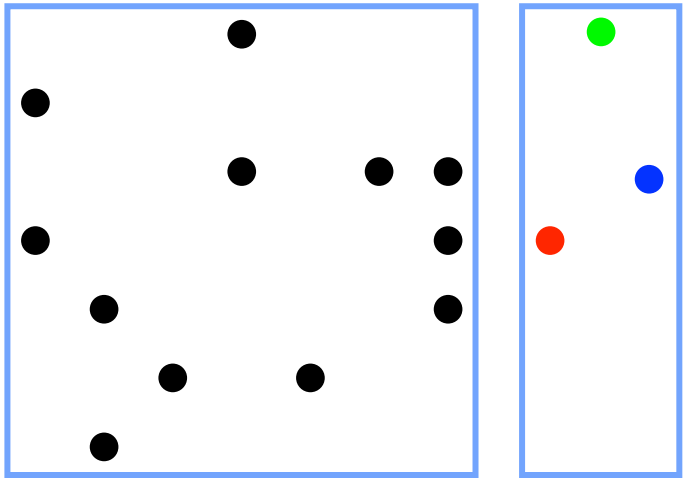


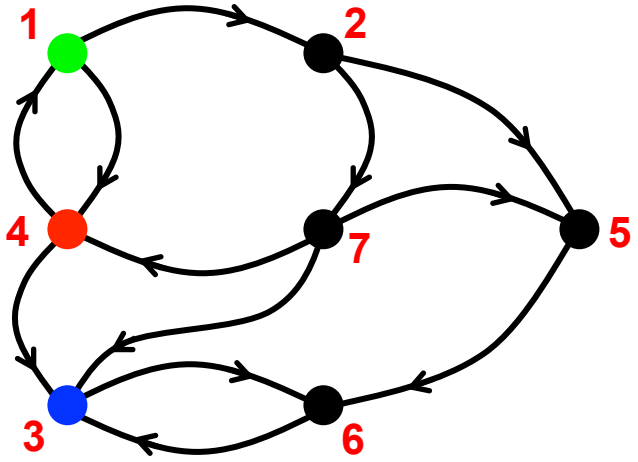
Sparse matrix indexing



The Combinatorial BLAS implements these, and more, on arbitrary semirings, e.g. (×, +), (and, or), (+, min)

Multiple-source breadth-first search

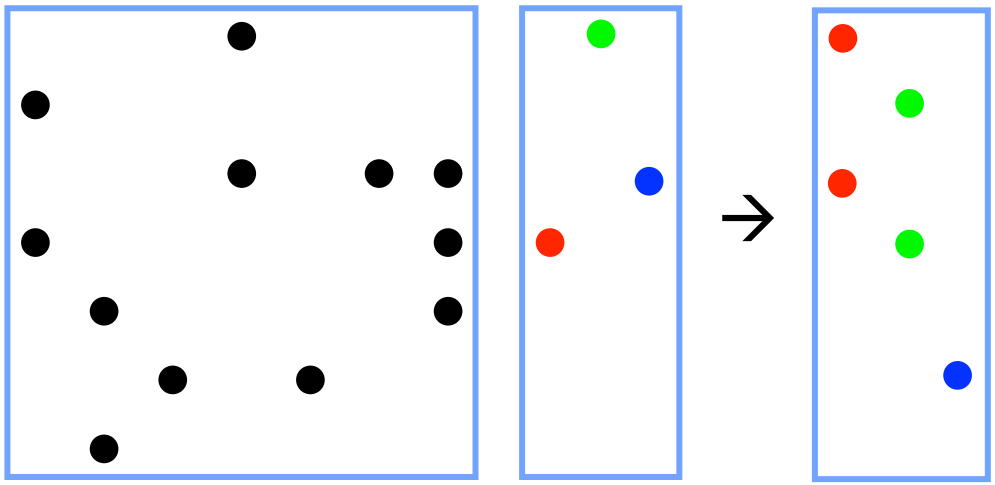


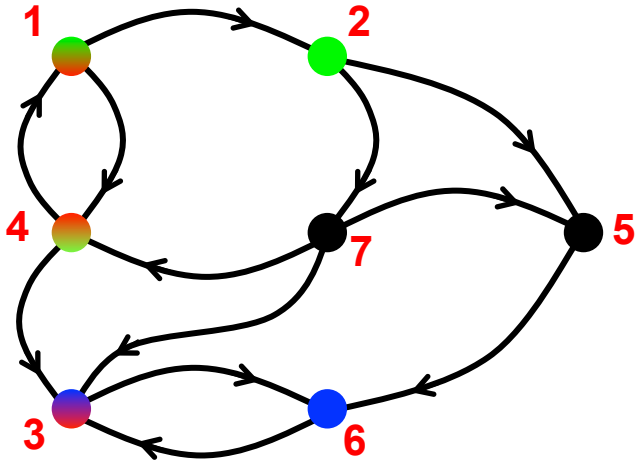


- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

*: A measure of influence in graphs, based on shortest paths

Multiple-source breadth-first search





- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges
- Highly-parallel implementation for Betweenness Centrality*

*: A measure of influence in graphs, based on shortest paths

Matrix times Matrix over semiring

Inputs

matrix **A**: $\mathbb{S}^{MxN}$ (sparse or dense)

matrix **B**: $\mathbb{S}^{NxL}$ (sparse or dense)

Optional Inputs

matrix **C**: $\mathbb{S}^{MxL}$ (sparse or dense)

scalar "add" function $\oplus$

scalar "multiply" function $\otimes$

transpose flags for **A**, **B**, **C**

Outputs

matrix **C**: $\mathbb{S}^{MxL}$ (sparse or dense)

Implements $\mathbf{C} \oplus= \mathbf{A} \oplus.\otimes \mathbf{B}$

for j = 1 : N

$\mathbf{C}(i,k) = \mathbf{C}(i,k) \oplus (\mathbf{A}(i,j) \otimes \mathbf{B}(j,k))$

If input **C** is omitted, implements

$\mathbf{C} = \mathbf{A} \oplus.\otimes \mathbf{B}$

Transpose flags specify operation on $\mathbf{A}^T$, $\mathbf{B}^T$, and/or $\mathbf{C}^T$ instead

Notes

$\mathbb{S}$ is the set of scalars, user-specified

$\mathbb{S}$ defaults to IEEE double float

$\oplus$ defaults to floating-point +

$\otimes$ defaults to floating-point *

Specific cases and function names:

SpGEMM: sparse matrix times sparse matrix

SpMSpV: sparse matrix times sparse vector

SpMV: Sparse matrix times dense vector

SpMM: Sparse matrix times dense matrix

Sparse Matrix Indexing & Assignment

Inputs

matrix **A**: $\mathbb{S}^{MxN}$ (sparse)

matrix **B**: $\mathbb{S}^{|p|x|q|}$ (sparse)

vector $p \subseteq \{1, \ldots, M\}$

vector $q \subseteq \{1, \ldots, N\}$

Optional Inputs

none

Outputs

matrix **A**: $\mathbb{S}^{MxN}$ (sparse)

matrix **B**: $\mathbb{S}^{|p|x|q|}$ (sparse)

SpRef Implements **B** = **A**(p,q)

```
for i = 1 : |p|
  for j = 1 : |q|
    B(i,j) = A(p(i),q(j))
```

SpAsgn Implements **A**(p,q) = **B**

```
for i = 1 : |p|
  for j = 1 : |q|
    A(p(i),q(j)) = B(i,j)
```

Notes

$\mathbb{S}$ is the set of scalars, user-specified

$\mathbb{S}$ defaults to IEEE double float

$|p|$ = length of vector p

$|q|$ = length of vector q

Specific cases and function names

SpRef: get sub-matrix

SpAsgn: assign to sub-matrix

Element-Wise Operations

Inputs

matrix **A**: $\mathbb{S}^{MxN}$ (sparse or dense)

matrix **B**: $\mathbb{S}^{MxN}$ (sparse or dense)

Optional Inputs

matrix **C**: $\mathbb{S}^{MxN}$ (sparse or dense)

scalar "add" function $\oplus$

scalar "multiply" function $\otimes$

Outputs

matrix **C**: $\mathbb{S}^{MxN}$ (sparse or dense)

Implements $\mathbf{C} \oplus= \mathbf{A} \otimes \mathbf{B}$

for i = 1 : M

  for j = 1 : N

    C(i,j) = **C**(i,j) $\oplus$ (**A**(i,j) $\otimes$ **B**(i,j))

If input **C** is omitted, implements

  C = **A** $\otimes$ **B**

Notes

$\mathbb{S}$ is the set of scalars, user-specified

$\mathbb{S}$ defaults to IEEE double float

$\oplus$ defaults to floating-point +

$\otimes$ defaults to floating-point *

Specific cases and function names:

SpEWiseX: matrix elementwise

M=1 or N=1: vector elementwise

Scale: when **A** or **B** is a scalar

Apply/Update

Inputs

matrix **A**: $\mathbb{S}^{MxN}$ (sparse or dense)

Optional Inputs

matrix **C**: $\mathbb{S}^{MxN}$ (sparse or dense)

scalar "add" function $\oplus$

unary function f()

Outputs

matrix **C**: $\mathbb{S}^{MxN}$ (sparse or dense)

Implements $\mathbf{C} \oplus= f(\mathbf{A})$

for i = 1 : M
  for j = 1 : N
    if **A**(i,j) ≠ 0
      C(i,j) = **C**(i,j) $\oplus$ f(**A**(i,j))

If input **C** is omitted, implements
  C = f(**A**)

Notes

$\mathbb{S}$ is the set of scalars, user-specified

$\mathbb{S}$ defaults to IEEE double float

$\oplus$ defaults to floating-point +

Specific cases and function names:

Apply: matrix apply

M=1 or N=1: vector apply

Matrix/Vector Reductions

Inputs

matrix **A**: $\mathbb{S}^{MxN}$ (sparse or dense)

Optional Inputs

vector **c**: $\mathbb{S}^{M}$ or $\mathbb{S}^{N}$ (sparse or dense)

scalar "add" function $\oplus$

dimension d: 1 or 2

Outputs

matrix **c**: $\mathbb{S}^{MxN}$ (sparse or dense)

Implements $\mathbf{c}(i) \oplus= \bigoplus_j \mathbf{A}(i,j)$

for i = 1 : M

  for j = 1 : N

    $\mathbf{c}(i) = \mathbf{c}(i) \oplus \mathbf{A}(i,j)$

If input **C** is omitted, implements

$\mathbf{c}(i) = \bigoplus_j \mathbf{A}(i,j)$

Notes

$\mathbb{S}$ is the set of scalars, user-specified

$\mathbb{S}$ defaults to IEEE double float

$\oplus$ defaults to floating-point +

d defaults to 2

Specific cases and function names:

Reduce (d = 1): reduce matrix to row vector

Reduce (d = 2): reduce matrix to col vector

GraphBLAS initial function list

Function	Parameters	Returns	Math Notation
SpGEMM	- sparse matrices **A** and **B** - unary functors (op)	sparse matrix	**C** = op(**A**) * op(**B**)
SpM{Sp}V **(Sp: sparse)**	- sparse matrix **A** - sparse/dense vector **x**	sparse/dense vector	**y** = **A** * **x**
SpEWiseX	- sparse matrices or vectors - binary functor and predicate	in place or sparse matrix/vector	**C** = **A** .* **B**
Reduce	- sparse matrix **A** and functors	dense vector	**y** = sum(**A**, op)
SpRef	- sparse matrix **A** - index vectors **p** and **q**	sparse matrix	**B** = **A**(**p**,**q**)
SpAsgn	- sparse matrices **A** and **B** - index vectors **p** and **q**	none	**A**(**p**,**q**) = **B**
Scale	- sparse matrix **A** - dense matrix or vector **X**	none	check manual
Apply	- any matrix or vector **X** - unary functor (op)	none	op(**X**)

Summary & Way Forward

Accomplishments

- GraphBLAS effort <1 year since its first meeting (IEEE HPEC 2013)
- General consensus on core functions
- Well defined mathematics with numerous implementations

Next Steps

- Next meeting: IEEE HPEC 2014 Sep 10, Waltham, MA
- Develop meta-spec that codifies mathematics
- Have implementations apply for "blessing"
- Current efforts include:
  - C++: CombBLAS, C: Stinger, Java: GraphLab/Accumulo Graph Library; Matlab/Octave: D4M, Julia, Python: ???
- Evolve into *de facto* language standards targetable by hardware