Graphs and Sparse Matrices: There and Back Again

John R. Gilbert
University of California, Santa Barbara

SIAM Combinatorial Scientific Computing
October 11, 2016

Support: Intel, Microsoft, DOE Office of Science, NSF

Outline

- Elimination, chordal graphs, and treewidth
- A tale of two matroids
- Magic eigenvectors
- Graphs in the language of linear algebra
- A word about A^TA
- Concluding remarks

Outline

- Elimination, chordal graphs, and treewidth
- A tale of two matroids
- Magic eigenvectors
- Graphs in the language of linear algebra
- A word about A^TA
- Concluding remarks

Combinatorics in the service of linear algebra

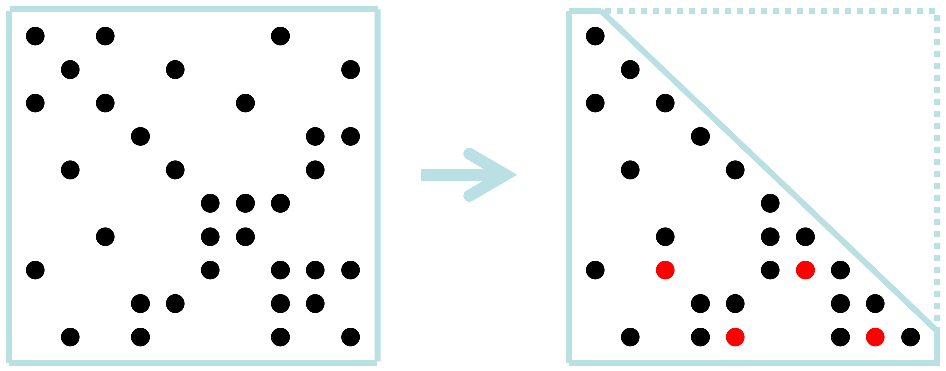
"I observed that most of the coefficients in our matrices were zero; i.e., the nonzeros were 'sparse' in the matrix, and that typically the triangular matrices associated with the forward and back solution provided by Gaussian elimination would remain sparse if pivot elements were chosen with care"

- Harry Markowitz, describing the 1950s work on portfolio theory that won the 1990 Nobel Prize for Economics

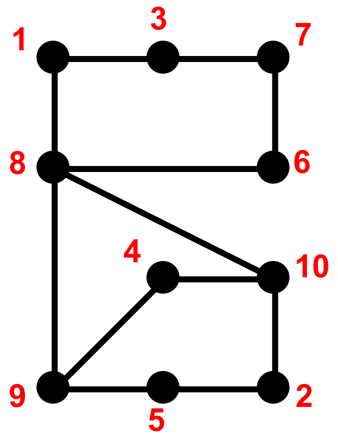
UCSB

Cholesky factorization: $\mathbf{A} = \mathbf{LL}^T$

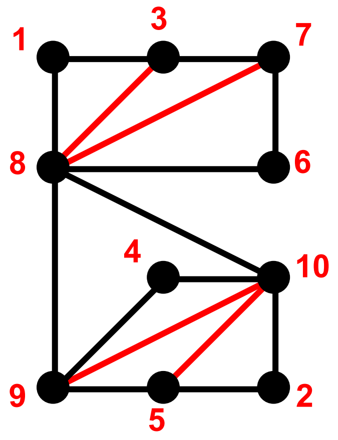
[Parter, Rose]



Fill: new nonzeros in factor



G(A)



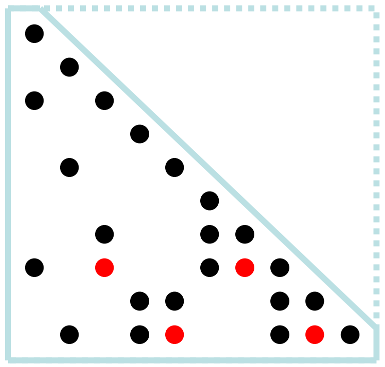
$G^+(A)$
[chordal]

Symmetric Gaussian elimination:

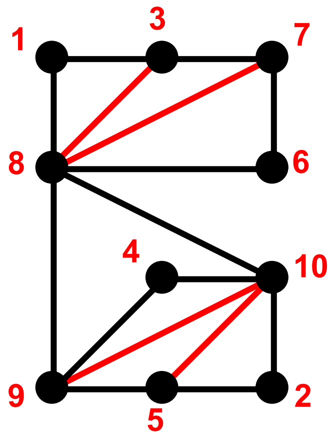
```
for j = 1 to n
    add edges between j's
    higher-numbered neighbors
```

Chordal graphs and trees: Elimination tree

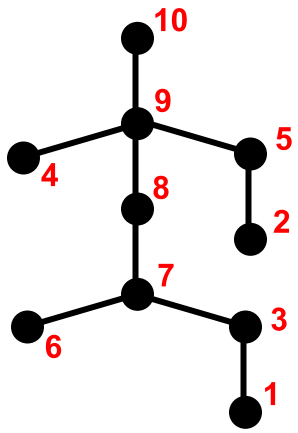
[Duff, Schreiber]



Cholesky factor



$G^+(A)$



T(A)

T(A) : parent(j) = min { i > j : (i, j) in $G^+(A)$ }

parent(col j) = first nonzero row below diagonal in L

- T describes dependencies among columns of factor
- Can compute T from G(A) in almost linear time
- Can compute filled graph $G^+(A)$ easily from T

Sparse Gaussian elimination and chordal completion
[Parter, Rose]

```
Repeat:
        Choose a vertex v and mark it;
        Add edges between unmarked neighbors of v;
Until:  Every vertex is marked
```

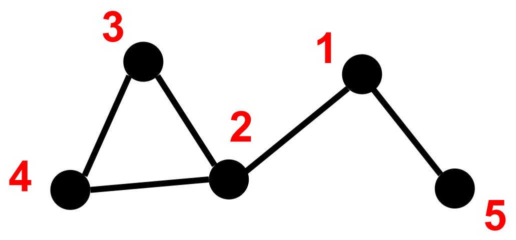
Goal: End up with as few edges as possible.

Equivalently: Add as few edges as possible to make the graph chordal.

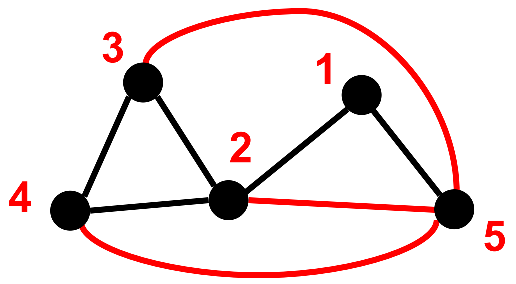
(Note for later: “Fewest edges” is not the only interesting objective.)

Sparse Gaussian elimination and chordal completion

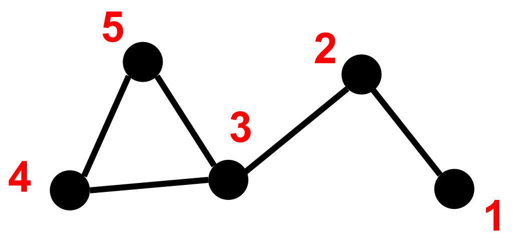
[Parter, Rose]



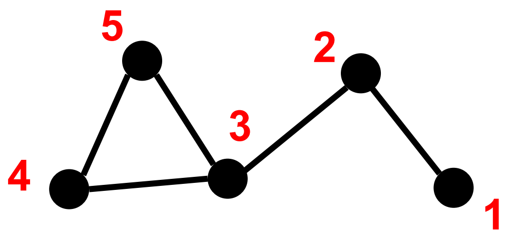
$$Ax = b$$



$$A = L_1 L_1^T$$



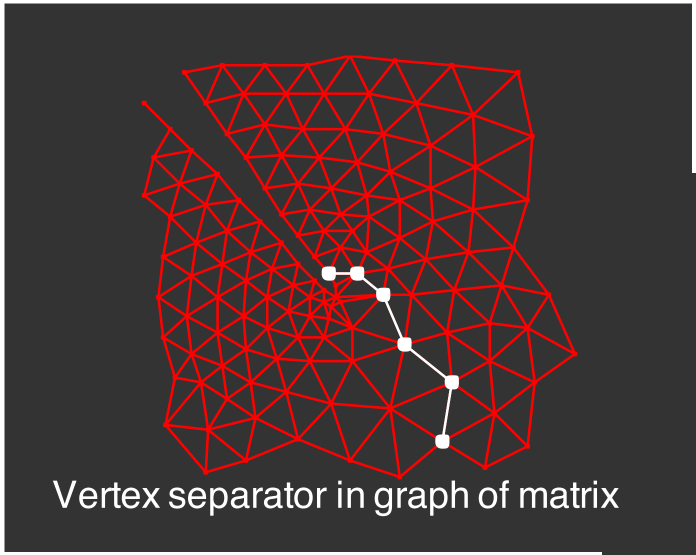
$$(PAP^T)\ (Px) = (Pb)$$

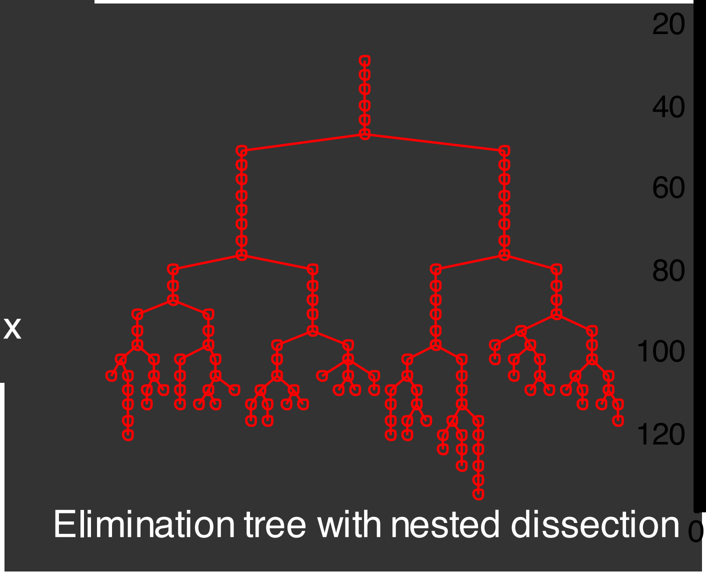


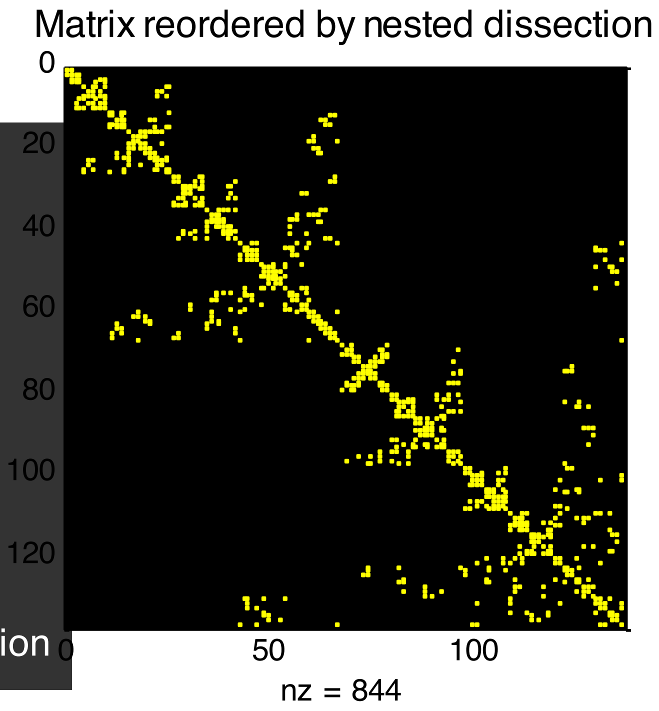
$$PAP^T = L_2 L_2^T$$

Nested dissection and graph partitioning

[George 1973, many extensions]







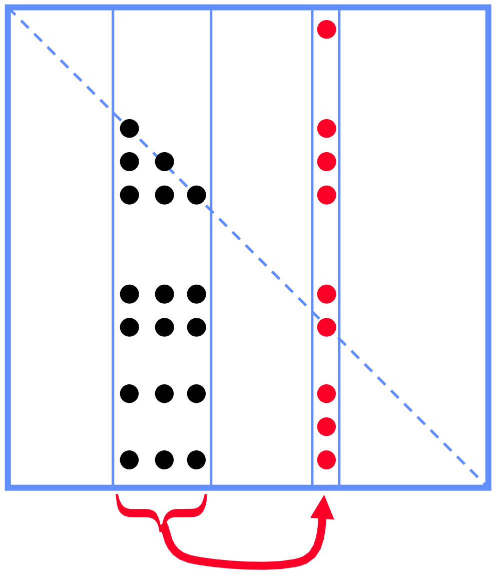
- Minimum-size chordal completion is NP-complete *[Yannakakis]*
- Heuristic: Find small vertex separator, put it last, recurse on subgraphs
- Theory: approx optimal separators => approx optimal fill

Chordal graphs and data movement

- A chordal graph can be represented as a tree of overlapping cliques (complete subgraphs).
- A complete subgraph is a dense submatrix.
- Dense matrix ops do n^3 work for n^2 data movement.
- Most of the ops in Gaussian elimination can be done within dense matrix primitives, esp. DGEMM.

Supernodes for Gaussian elimination

- Supernode = group of adjacent columns of L with same nonzero structure
- Related to clique structure of filled graph $G^+(A)$



- Supernode-column update: k sparse vector ops become

  1 dense triangular solve
  + 1 dense matrix * vector
  + 1 sparse vector add

- Sparse BLAS 1 => Dense BLAS 2
- Supernode-panel or multifrontal updates => Dense BLAS 3

Aside: Matrix structure prediction

- Computing the nonzero structure of Cholesky factor L is much cheaper than computing L itself.
- Computing nnz(L) is nearly linear in nnz(A). *[G, Ng, Peyton]*

UCSB

Aside: Matrix structure prediction

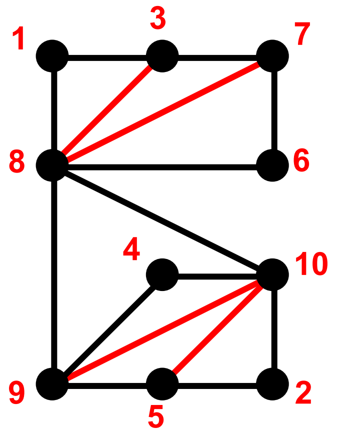
- Computing the nonzero structure of Cholesky factor L is much cheaper than computing L itself.
- Computing nnz(L) is nearly linear in nnz(A). *[G, Ng, Peyton]*
- Not so for sparse matrix product (SpGEMM); computing nnz(B*C) seems to be as hard as computing B*C.

Aside: Matrix structure prediction

- Computing the nonzero structure of Cholesky factor L is much cheaper than computing L itself.
- Computing nnz(L) is nearly linear in nnz(A). *[G, Ng, Peyton]*
- Not so for sparse matrix product (SpGEMM); computing nnz(B*C) seems to be as hard as computing B*C.
- Can *estimate* nnz(B*C) accurately in time linear in nnz(B, C)! *[E. Cohen 1998]*
- Lots of cool recent work on sampling algorithms to estimate statistics of matrix functions.

Complexity measures for chordal completion

$G^+(A)$



Elimination degree:

d_j = # higher neighbors of j in G^+

$d = (2, 2, 2, 2, 2, 2, 1, 2, 1, 0)$

- Nonzeros = edges = $\Sigma_j\, d_j$ (moment 1)
- Work = flops = $\Sigma_j\, (d_j)^2$ (moment 2)
- Front size = treewidth = $\max_j\, d_j$ (moment ∞)

Treewidth shows up in lots of other graph algorithms…

UCSB

Treewidth and Fixed-Parameter-Tractable problems

[Downey/Fellows, Arnborg/Proskurowski, Sullivan, many others]

- Many NP-hard problems on graphs have low-order polynomial complexity on graphs of bounded treewidth.
- Coloring, feedback vertex set, co-path set, …
- Algebraic geometry: Solving polynomial equations by chordal elimination & Gröbner bases *[Cifuentes/Parrilo]*
- Bounded treewidth implies linear-time Ax = b, pretty rare!
- Treewidth of some graph classes:
  - Planar: O(sqrt(n))
  - Random Erdos-Renyi (connected): O(n)
  - Random hyperbolic power law (>=3): O(log n) or O(1)

Network graph decompositions

- Empirical measurements of "treelike structure" in real graphs: *[Adcock/Sullivan/Mahoney 2016 etc.]*
  - Treewidth
  - Hyperbolicity
  - Core-periphery decomposition etc.
- "Bounded expansion": Contracting some disjoint low-diameter subgraphs leaves all subgraphs sparse *[Demaine et al. 2014]*
- Question: What can new ways of looking at the structure of network graphs tell us about efficient linear solvers and eigensolvers?

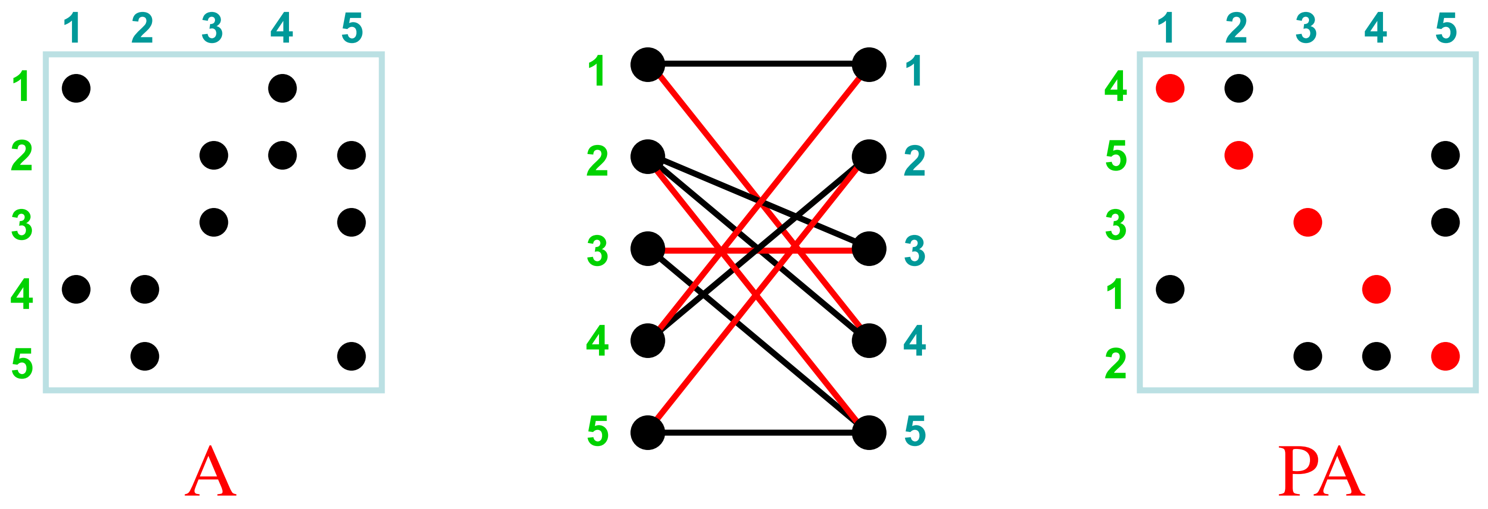
Outline

- Elimination, chordal graphs, and treewidth
- A tale of two matroids
- Magic eigenvectors
- Graphs in the language of linear algebra
- A word about A^TA
- Concluding remarks

Matroid: Abstraction of "independent sets" and rank

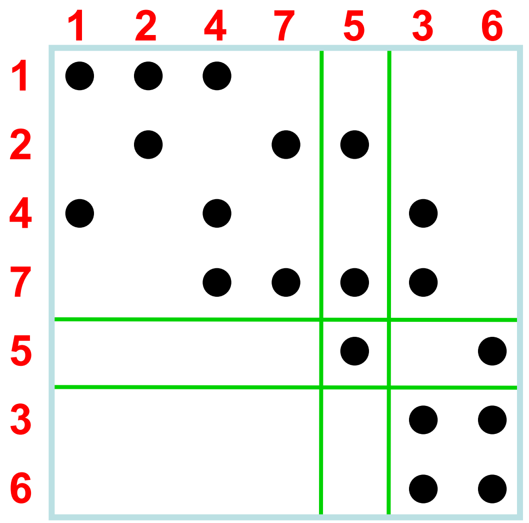
- Linear matroid:
  - Rows of a matrix
  - Independent set = linearly independent vectors.
  - Matlab "rank"
- Matching matroid:
  - Vertices of bipartite graph
  - Independent set = vertices with row-complete matching.
  - Matlab "sprank"
- Often the same (under a no-cancellation assumption).
- The matching matroid is less expensive to compute with!

Bipartite matching

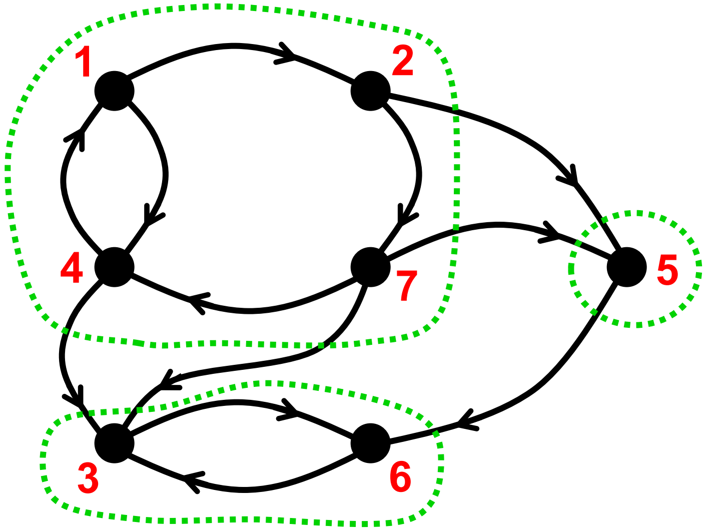


- <span>Perfect matching:</span> set of edges that hits each vertex exactly once
- Matrix permutation to place nonzeros (or heavy elements) on diagonal
- Efficient sequential algorithms based on augmenting paths
- No known work/span efficient parallel algorithms

Strongly connected components



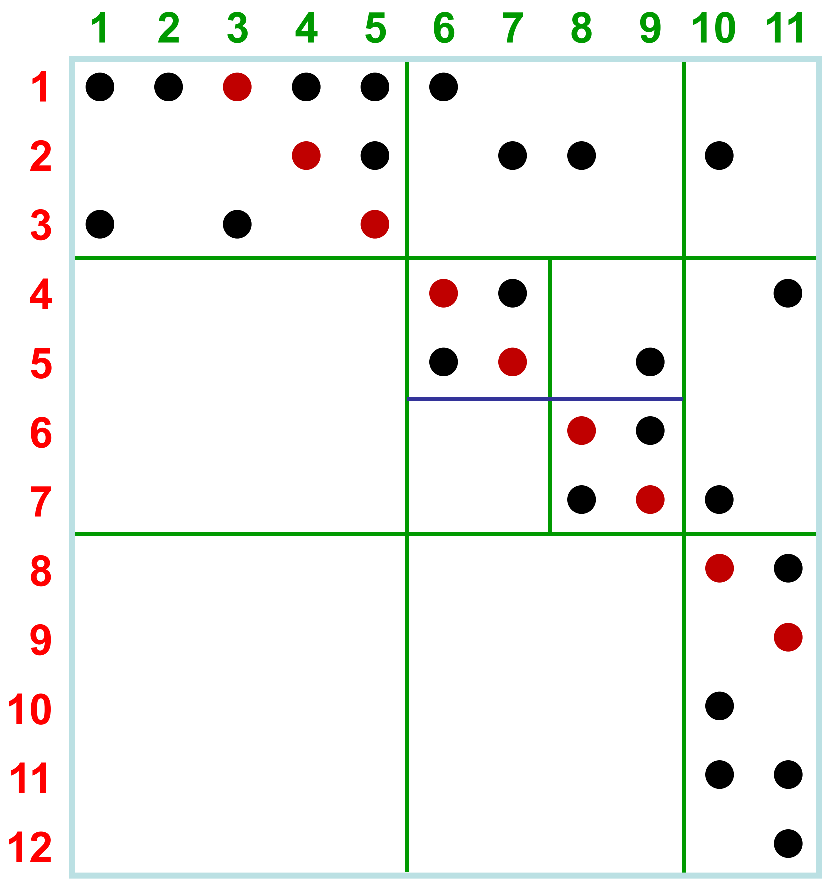
PAP^T

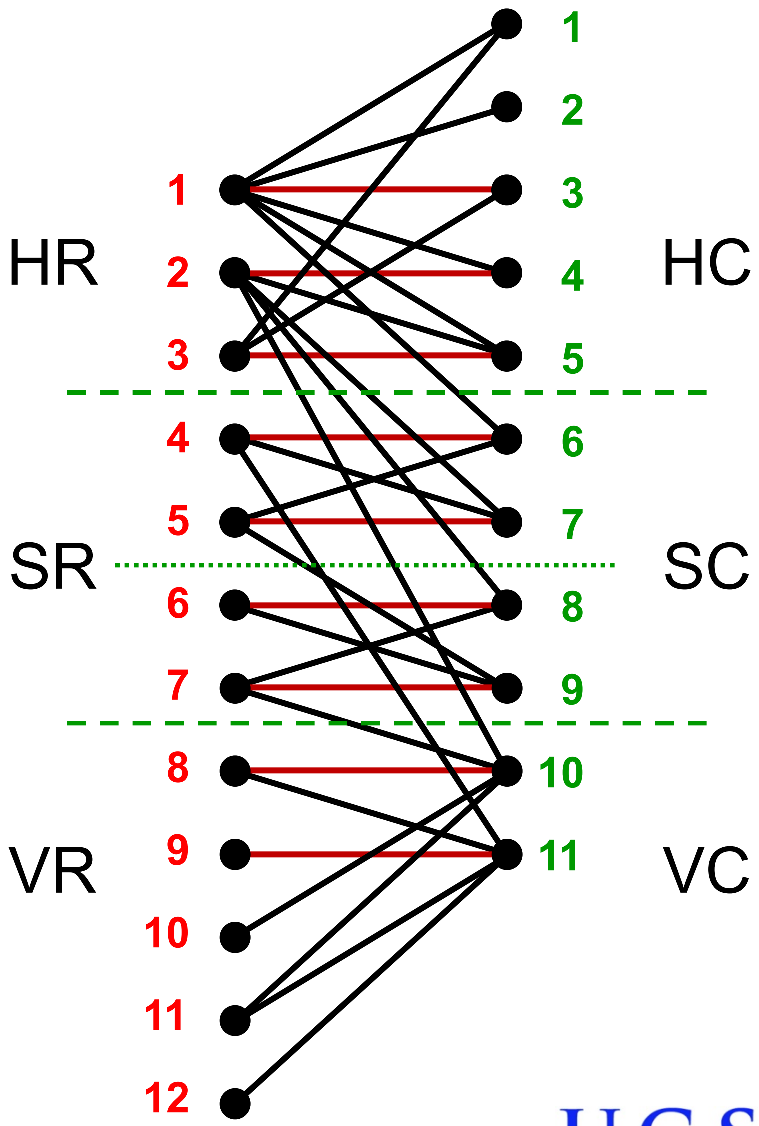


G(A)

- Symmetric permutation to block triangular form
- Diagonal blocks are strong Hall (irreducible / strongly connected)
- Sequential: linear time by depth-first search *[Tarjan]*
- Parallel: divide & conquer, work and span depend on input *[Fleischer, Hendrickson, Pinar]*

Dulmage-Mendelsohn decomposition

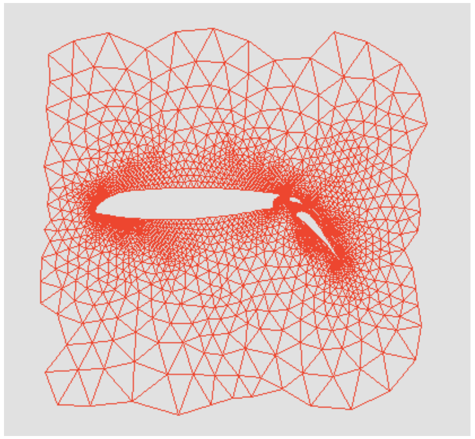


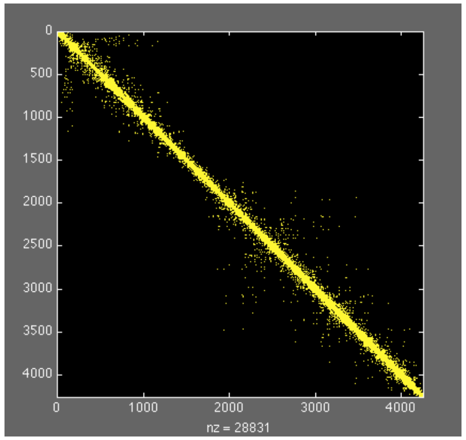


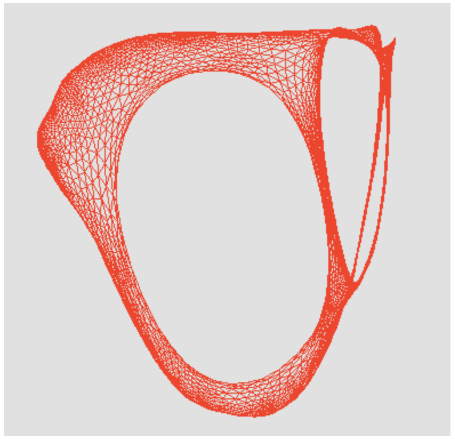
Outline

- Elimination, chordal graphs, and treewidth
- A tale of two matroids
- Magic eigenvectors
- Graphs in the language of linear algebra
- A word about A^TA
- Concluding remarks

Laplacian matrix of a graph







- Symmetric, positive semidefinite matrix.
- Eigenvectors for partitioning and embedding
  [Fiedler, Pothen/Simon, Spielman/Teng, many others]
- Laplacian paradigm: Use Ax = b as a subroutine in graph algorithms
  [Kelner, Teng, many others]
- See tomorrow morning's talks by Gary Miller, Elisabetta Bergamini, Kevin Deweese, Tristan Konolige

Graph algorithms in sparse matrix computation

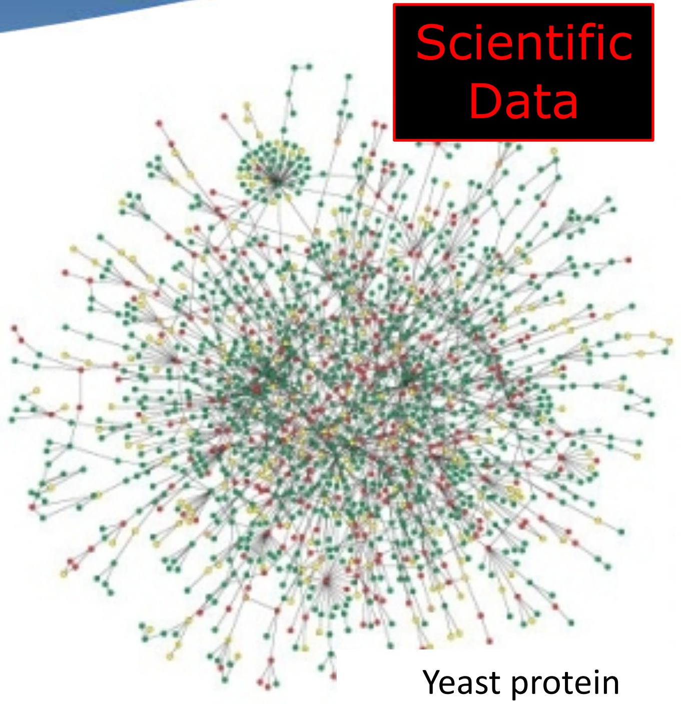
Many, many graph algorithms have been used, invented, implemented at large scale for sparse matrix computation:

- Symmetric problems: elimination tree, nonzero structure prediction, sparse triangular solve, sparse matrix-matrix multiplication, min-height etree, …
- Nonsymmetric problems: sparse triangular solve, bipartite matching (weighted and unweighted), Dulmage-Mendelsohn decomposition / strong components, …
- Iterative methods: graph partitioning again, independent set, low-stretch spanning trees, …

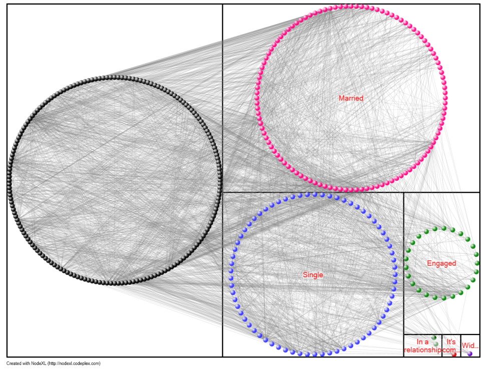
Outline

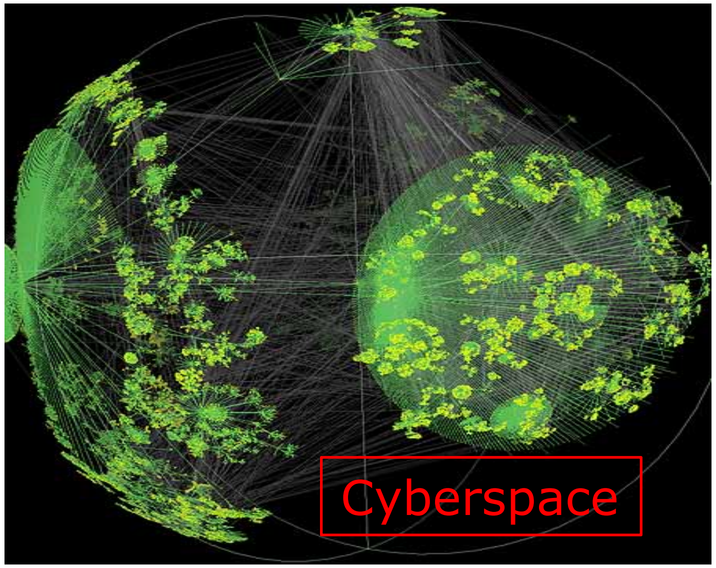
- Elimination, chordal graphs, and treewidth
- A tale of two matroids
- Magic eigenvectors
- Graphs in the language of linear algebra
- A word about A^TA
- Concluding remarks

Large graphs are everywhere…



Yeast protein interactions, courtesy H. Jeong





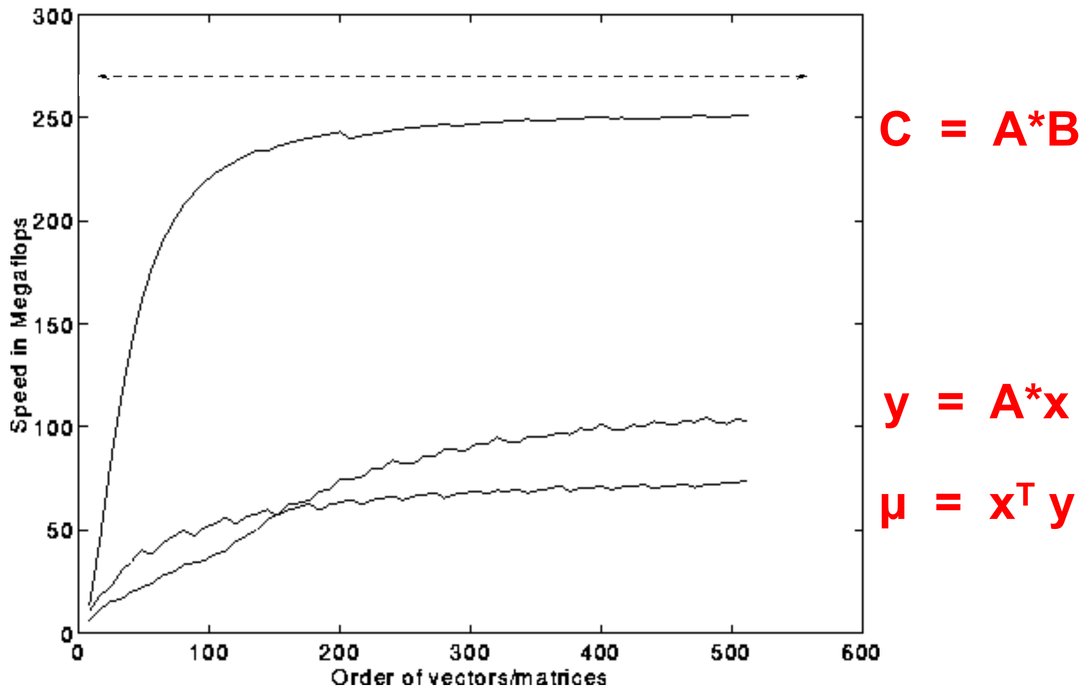
WWW snapshot, courtesy Y. Hyun

The middleware challenge for graph analysis

- By analogy to numerical scientific computing. . .

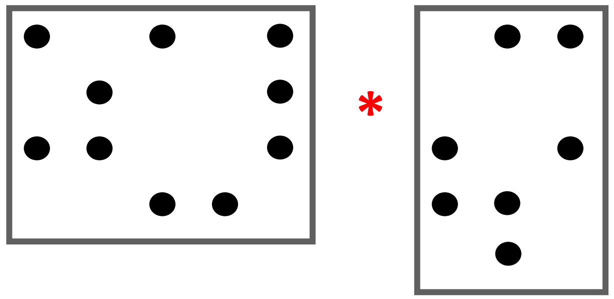
- What should the combinatorial BLAS look like?

Basic Linear Algebra Subroutines (BLAS):
Ops/Sec vs. Matrix Size

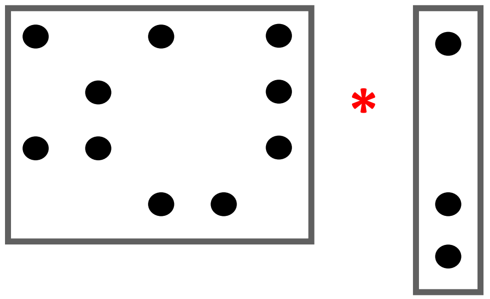


Sparse array primitives for graphs

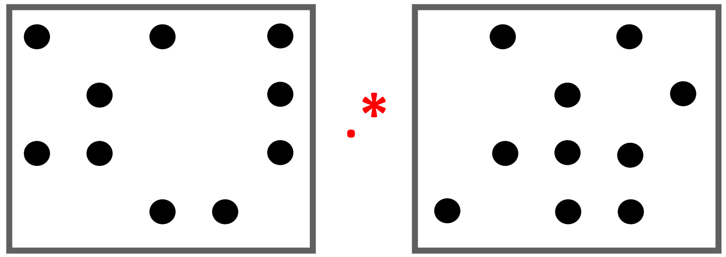
Sparse matrix-sparse matrix multiplication



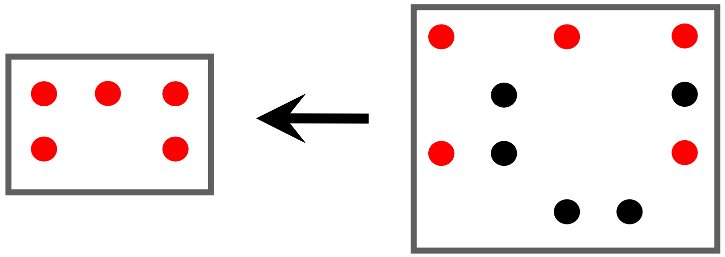
Sparse matrix-sparse vector multiplication



Element-wise operations

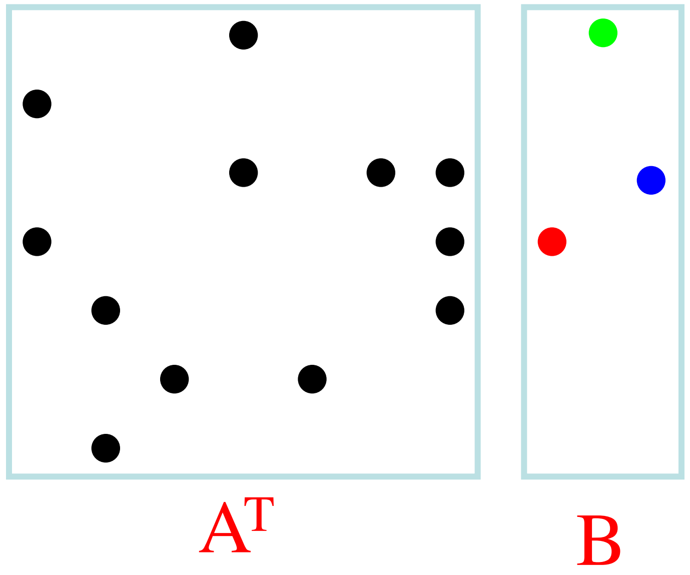


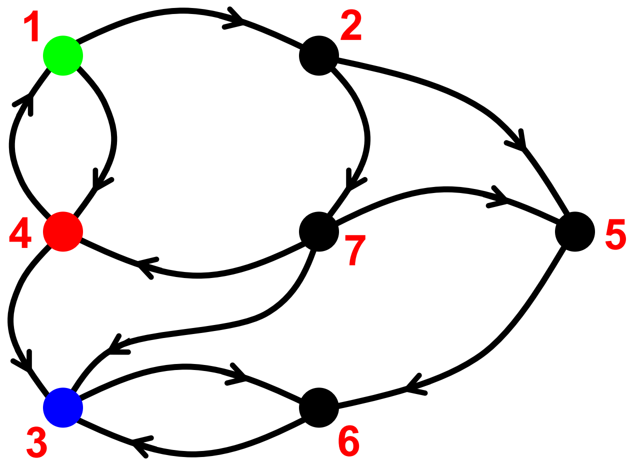
Sparse matrix indexing



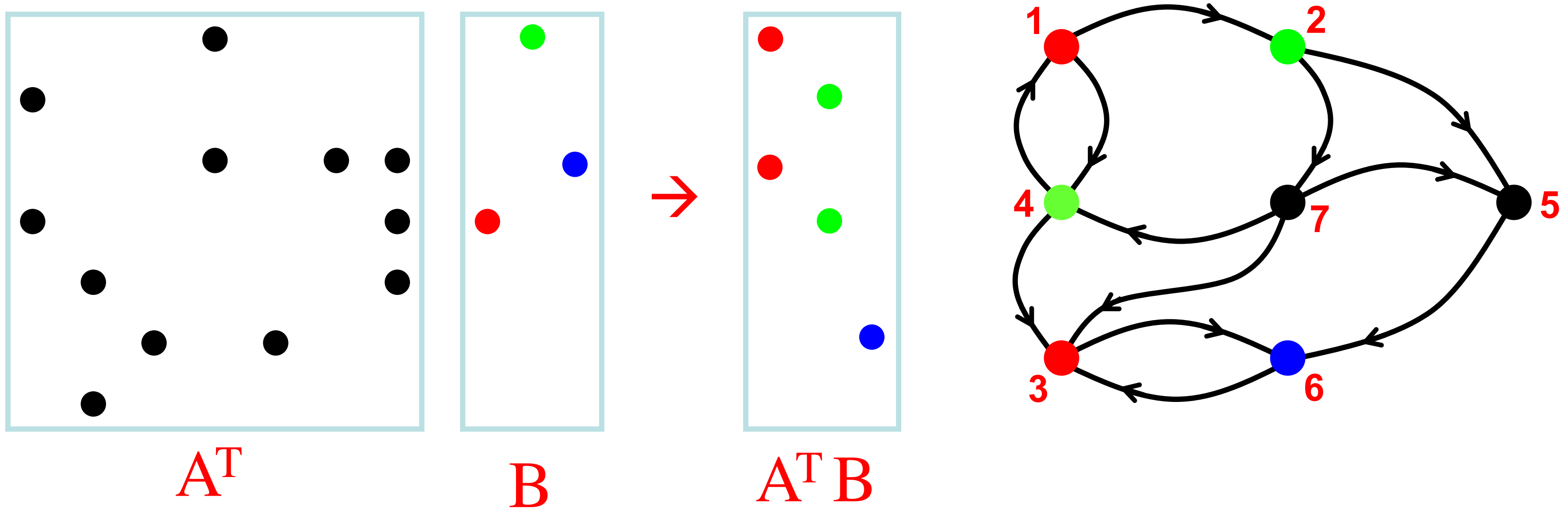
Matrices over various semirings: (+, ·), (and, or), (min, +), …

Multiple-source breadth-first search





Multiple-source breadth-first search



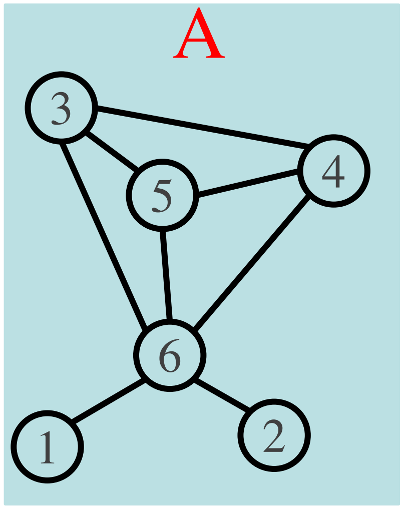
- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Three possible levels of parallelism: searches, vertices, edges

UCSB

Examples of semirings in graph algorithms

("values": edge/vertex attributes, "add": vertex data aggregation, "multiply": edge data processing)	General schema for user-specified computation at vertices and edges
Real field: (R, +, *)	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

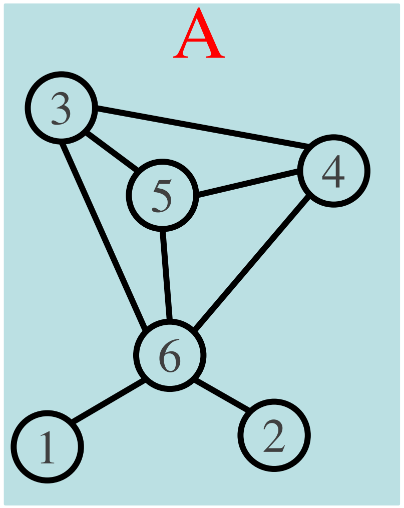
Counting triangles (clustering coefficient)



Clustering coefficient:

- Pr (wedge i-j-k makes a triangle with edge i-k)
- 3 * # triangles / # wedges
- 3 * **4** / **19** = **0.63** in example
- may want to compute for each vertex j

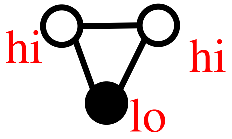
Counting triangles (clustering coefficient)



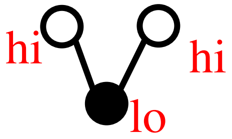
Clustering coefficient:

- Pr (wedge i-j-k makes a triangle with edge i-k)
- 3 * # triangles / # wedges
- 3 * **4** / **19** = **0.63** in example
- may want to compute for each vertex j

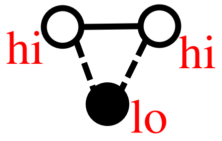
"Cohen's" algorithm to count triangles:



- Count triangles by lowest-degree vertex.

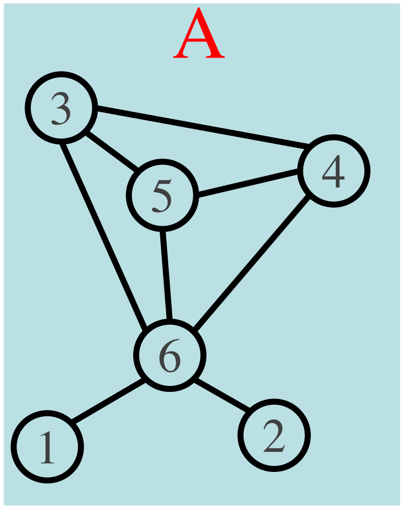


- Enumerate "low-hinged" wedges.



- Keep wedges that close.

Counting triangles (clustering coefficient)

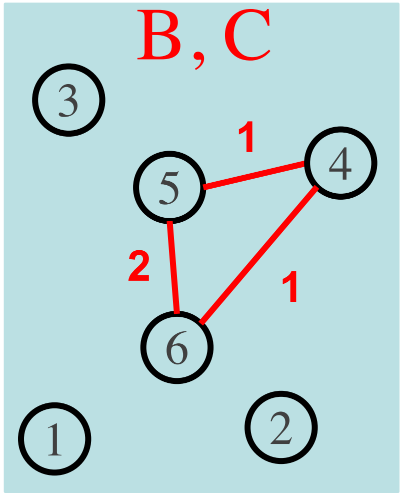


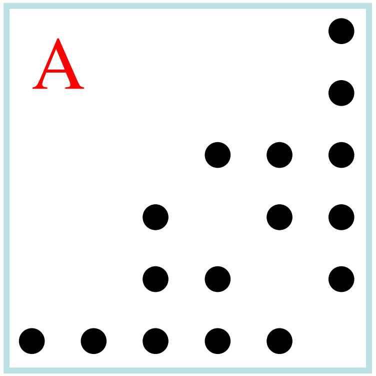
$A = L + U$ (hi->lo + lo->hi)

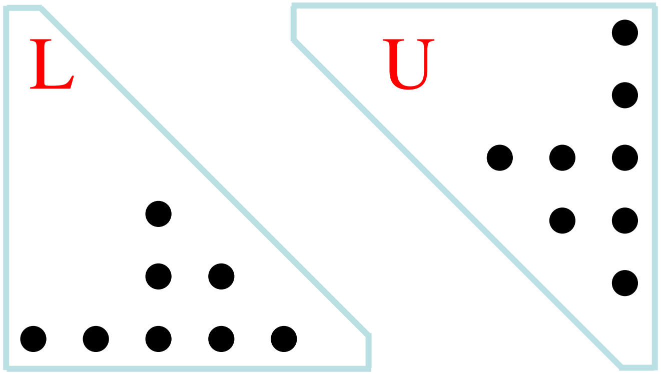
$L \times U = B$ (wedge, low hinge)

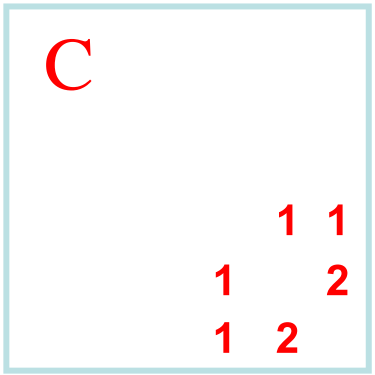
$A \wedge B = C$ (closed wedge)

sum(C)/2 = **4 triangles**





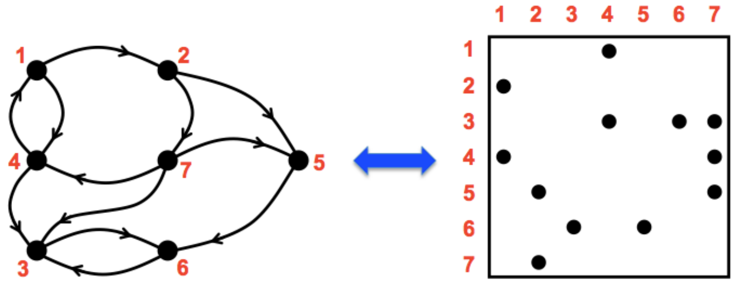




Combinatorial BLAS

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

[Azad, Buluc, G, Lugowski, …]



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.
- Scalable performance from laptop to 100,000-processor HPC.

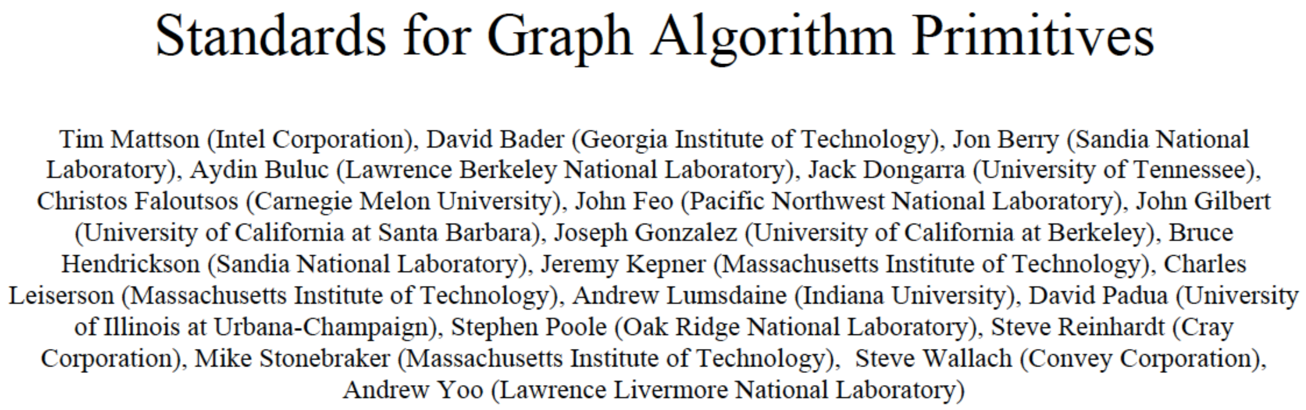
- Open source software.
- Version 1.5.0 released January, 2016.

Combinatorial BLAS "users" (Jan 2016)

- IBM (T.J.Watson, Zurich, & Tokyo)
- Intel
- Cray
- Microsoft
- Stanford
- MIT
- UC Berkeley
- Carnegie-Mellon
- Georgia Tech
- Purdue
- Ohio State
- U Texas Austin
- NC State
- UC San Diego
- UC Merced
- UC Santa Barbara
- Berkeley Lab
- Sandia Labs
- Columbia
- U Minnesota
- Duke
- Indiana U
- Mississippi State
- SEI
- Paradigm4
- Mellanox
- IHPC (Singapore)
- Tokyo Inst of Technology
- Chinese Academy of Sciences
- U Canterbury (New Zealand)
- King Fahd U (Saudi Arabia)
- Bilkent U (Turkey)
- U Ghent (Belgium)

The Graph BLAS effort

- Manifesto, HPEC 2013:



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.

- Workshops at IPDPS, HPEC, SC
- Periodic working group telecons and meetings
- Graph BLAS Forum: http://graphblas.org

GraphBLAS Base Operations

Operation	Math	Out	Inputs
mxm	$\mathbf{C}(\neg\mathbf{M}) \oplus= \mathbf{A}^{\mathbf{T}} \oplus.\otimes \mathbf{B}^{\mathbf{T}}$	**C**	$\neg$, **M**, $\oplus$, **A**, **T**, $\oplus.\otimes$, **B**, **T**
mxv (vxm)	$\mathbf{c}(\neg\mathbf{m}) \oplus= \mathbf{A}^{\mathbf{T}} \oplus.\otimes \mathbf{b}$	**c**	$\neg$, **m**, $\oplus$, **A**, **T**, $\oplus.\otimes$, **b**
eWiseMult	$\mathbf{C}(\neg\mathbf{M}) \oplus= \mathbf{A}^{\mathbf{T}} \otimes \mathbf{B}^{\mathbf{T}}$	**C**	$\neg$, **M**, $\oplus$, **A**, **T**, $\otimes$, **B**, **T**
eWiseAdd	$\mathbf{C}(\neg\mathbf{M}) \oplus= \mathbf{A}^{\mathbf{T}} \oplus \mathbf{B}^{\mathbf{T}}$	**C**	$\neg$, **M**, $\oplus$, **A**, **T**, $\oplus$, **B**, **T**
reduce (row)	$\mathbf{c}(\neg\mathbf{m}) \oplus= \oplus_j \mathbf{A}^{\mathbf{T}}(:,j)$	**c**	$\neg$, **m**, $\oplus$, **A**, **T**, $\oplus$
apply	$\mathbf{C}(\neg\mathbf{M}) \oplus= f(\mathbf{A}^{\mathbf{T}})$	**C**	$\neg$, **M**, $\oplus$, **A**, **T**, *f*
transpose	$\mathbf{C}(\neg\mathbf{M}) \oplus= \mathbf{A}^{\mathbf{T}}$	**C**	$\neg$, **M**, $\oplus$, **A (T)**
extract	$\mathbf{C}(\neg\mathbf{M}) \oplus= \mathbf{A}^{\mathbf{T}}(\mathbf{i},\mathbf{j})$	**C**	$\neg$, **M**, $\oplus$, **A**, **T**, **i**, **j**
assign	$\mathbf{C}(\neg\mathbf{M})\ (\mathbf{i},\mathbf{j}) \oplus= \mathbf{A}^{\mathbf{T}}$	**C**	$\neg$, **M**, $\oplus$, **A**, **T**, **i**, **j**
buildMatrix	$\mathbf{C}(\neg\mathbf{M}) \oplus= \mathbb{S}^{mxn}(\mathbf{i},\mathbf{j},\mathbf{v},\oplus)$	**C**	$\neg$, **M**, $\oplus$, $\oplus$, m, n, **i**, **j**, **v**
extractTuples	$(\mathbf{i},\mathbf{j},\mathbf{v}) = \mathbf{A}(\neg\mathbf{M})$	**i,j,v**	$\neg$, **M**, **A**

Notation: **i,j** – index arrays, **v** – scalar array, **m** – 1D mask, **other bold-lower** – vector (column), **M** – 2D mask, **other bold-caps** – matrix, **T** – transpose, **¬** - structural complement, $\oplus$ monoid/binary function, $\oplus.\otimes$ semiring, **blue** – optional parameters, **red** – optional modifiers (using Descriptors)

Breadth-first search with Graph BLAS (draft C API spec)

```
GrB_info BFS(GrB_Vector *v, GrB_Matrix A, GrB_index s)
/*
 * Given a boolean n x n adjacency matrix A and a source vertex s, performs a BFS traversal
 * of the graph and sets v[i] to the level in which vertex i is visited (v[s] == 1).
 * If i is not reacheable from s, then v[i] = 0. (Vector v should be empty on input.)
 */
{
  GrB_index n;
  GrB_Matrix_nrows(&n,A);                          // n = # of rows of A

  GrB_Vector_new(v,GrB_INT32,n);                   // Vector<int32_t> v(n)
  GrB_assign(v,0);                                 // v = 0

  GrB_Vector q;                                    // vertices visited in each level
  GrB_Vector_new(&q,GrB_BOOL,n);                   // Vector<bool> q(n)
  GrB_assign(&q,false);
  GrB_assign(&q,true,s);                           // q[s] = true, false everywhere else

  GrB_Space Boolean;                               // Boolean space <bool,bool,bool,||,&&,false,true>
  GrB_Space_new(&Boolean,GrB_BOOL,GrB_BOOL,GrB_BOOL,GrB_LOR,GrB_LAND,false,true);

  GrB_Descriptor desc;                             // Descriptor for vxm
  GrB_Descriptor_new(&desc);
  GrB_Descriptor_add(desc,GrB_ARG1,GrB_NOP);       // no operation on the vector
  GrB_Descriptor_add(desc,GrB_ARG2,GrB_NOP);       // no operation on the matrix
  GrB_Descriptor_add(desc,GrB_MASK,GrB_LNOT);      // invert the mask

  /*
   * BFS traversal and label the vertices.
   */
  int32_t d = 1;                                   // d = level in BFS traversal
  bool succ = false;                               // succ == true when some successor found
  do {
    GrB_assign(v,d,q);                             // v[q] = d
    GrB_vxm(&q,Boolean,q,A,*v,desc);               // q[!v] = q ||.&& A ; finds all the unvisited
                                                   // successors from current q
    GrB_reduce(&succ,q,GrB_LOR);                   // succ = ||(q)
    d++;                                           // next level
  } while (succ);                                  // if there is no successor in q, we are done.

  GrB_free(q);                                     // q vector no longer needed
  GrB_free(Boolean);                               // Boolean semiring no longer needed
  GrB_free(desc);                                  // descriptor no longer needed

  return GrB_SUCCESS;
}
```

Outline

- Elimination, chordal graphs, and treewidth
- A tale of two matroids
- Magic eigenvectors
- Graphs in the language of linear algebra
- A word about $A^T A$
- Concluding remarks

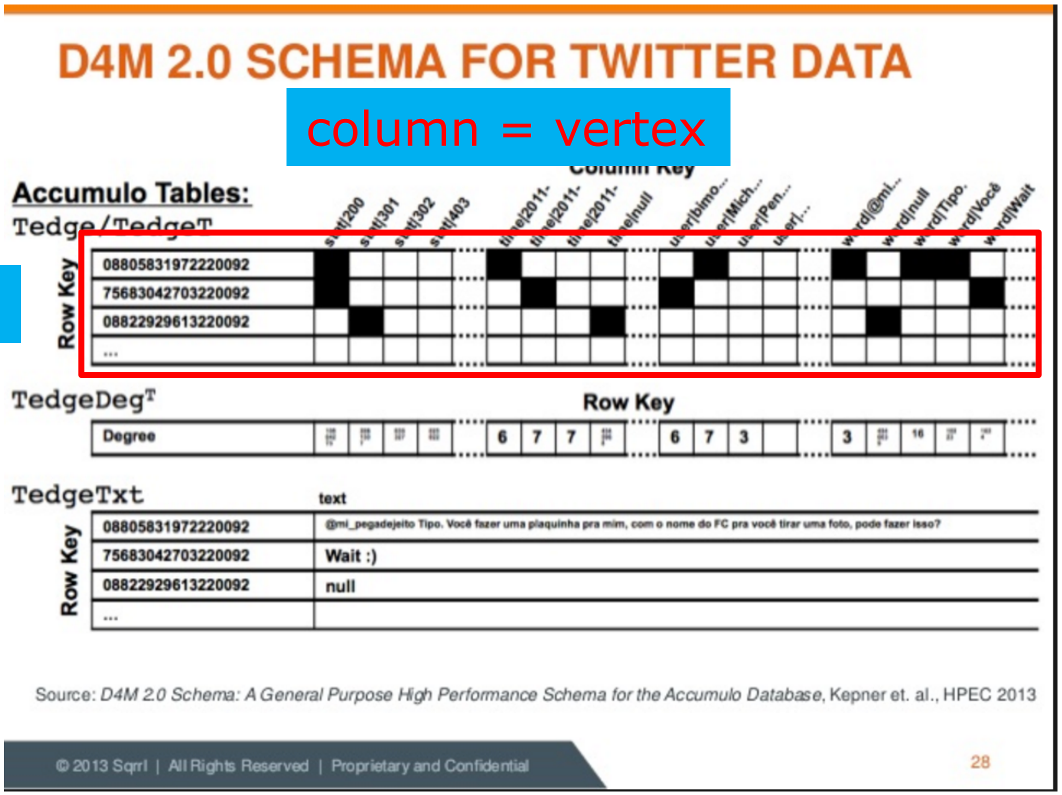
Storing A, operating implicitly on A^TA

- CombBLAS represents graphs as <u>adjacency</u> matrices.
- D4M *[Kepner et al.]* represents graphs as <u>incidence</u> matrices; matrix A represents $G(A^TA)$:

A

row = hyperedge

column = vertex



D4M 2.0 SCHEMA FOR TWITTER DATA

Accumulo Tables:

Tedge/TedgeT

Column Key

Row Key	stat|200	stat|301	stat|302	stat|403	....	time|2011-	time|2011-	time|2011-	time|null	....	user|bimo...	user|Mich...	user|Pen...	user|...	....	word|@mi...	word|null	word|Tipo.	word|Você	word|Wait	....
08805831972220092	■				....	■				....		■			....	■		■	■		....
75683042703220092	■				....		■			....	■				....					■	....
08822929613220092		■			....				■	....					....		■				....
...					....					....					....						....

TedgeDeg^T

Row Key

Degree	[illegible]	[illegible]	[illegible]	[illegible]	....	6	7	7	[illegible]	....	6	7	3		....	3	[illegible]	16	[illegible]	[illegible]	....

TedgeTxt

Row Key	text
08805831972220092	@mi_pegadejeito Tipo. Você fazer uma plaquinha pra mim, com o nome do FC pra você tirar uma foto, pode fazer isso?
75683042703220092	Wait :)
08822929613220092	null
...	

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

© 2013 Sqrrl | All Rights Reserved | Proprietary and Confidential 28

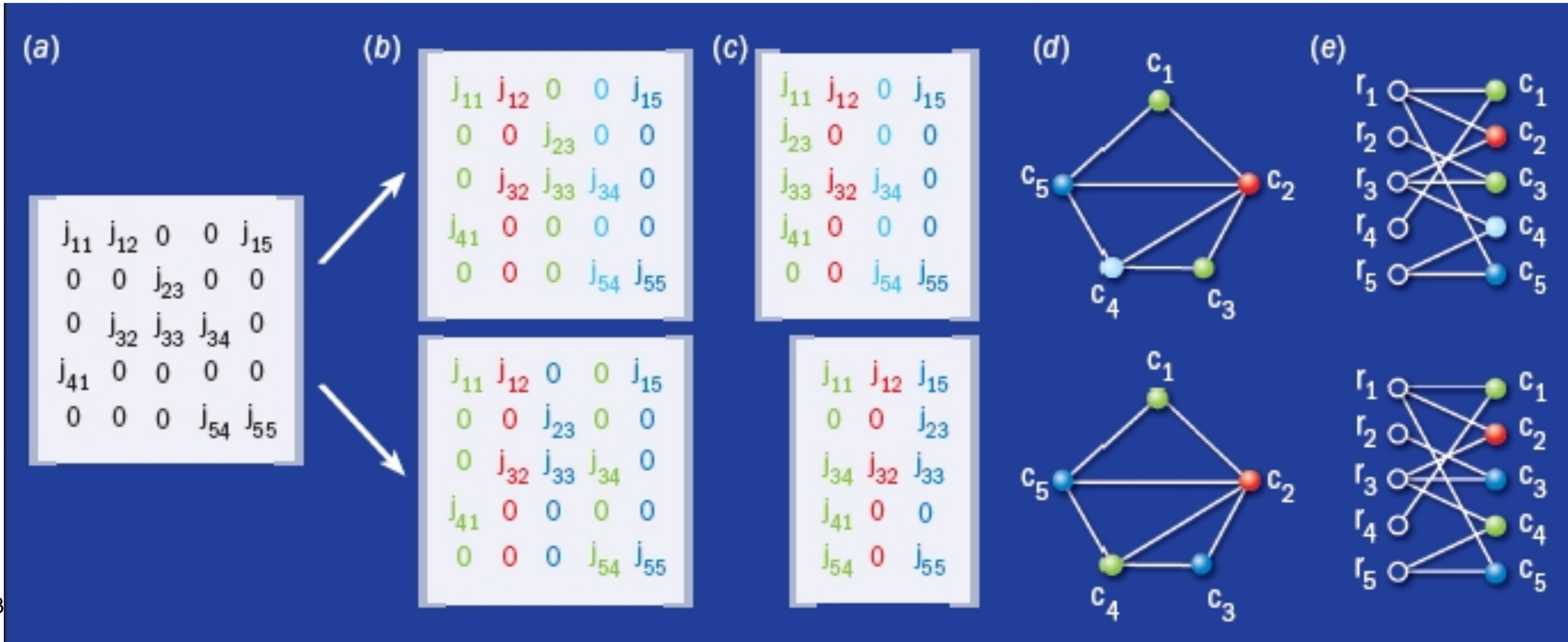
UCSB

Distance-2 coloring for sparse Jacobians

[Gebremedhin/Manne/Pothen etc.]

- Goal: compute (sparse) matrix J of partial derivatives, $J(i,j) = \partial y_i / \partial x_j$
- Nonoverlapping columns can be computed together.
- Method: find a coloring of $G(J^T J)$ without forming it from J.

Diagram courtesy CSCAPES


(a)
(b)
(c)
(d)
(e)
j_{11} j_{12} 0 0 j_{15}
0 0 j_{23} 0 0
0 j_{32} j_{33} j_{34} 0
j_{41} 0 0 0 0
0 0 0 j_{54} j_{55}
c_1 c_2 c_3 c_4 c_5
r_1 r_2 r_3 r_4 r_5

Storing A, operating implicitly on $A^T A$

- Many, many other cases:
  - Optimization: KKT systems, interior point methods.
  - Linear equations: QR factorization, structure prediction for LU factorization with partial pivoting.

- Question: What can you do fast on $G(A^T A)$ just from $G(A)$?

Structure prediction for PA = LU and A = QR

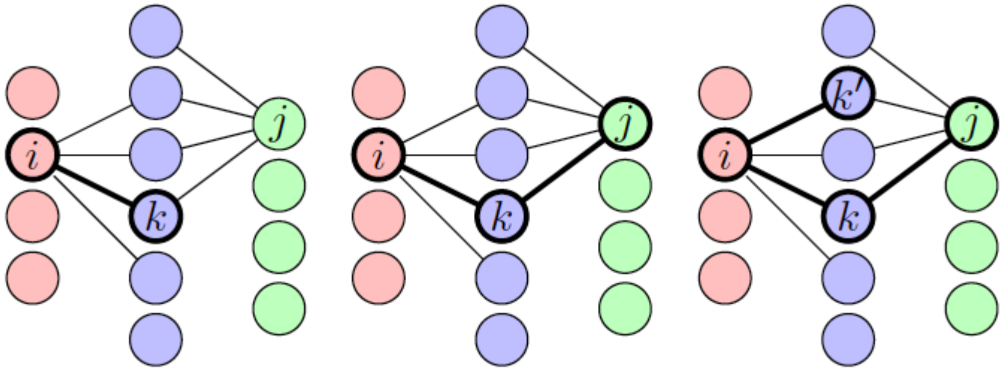
Given the nonzero structure of (nonsymmetric) A,
one can find . . .

- column nested dissection or min degree permutation
- column elimination tree $T(A^TA)$
- row and column counts for filled graph $G^+(A^TA)$
- supernodes of $G^+(A^TA)$
- nonzero structure of $G^+(A^TA)$

. . . efficiently, without forming A^TA.

Statistics for A^TA itself are harder!

- nnz(A^TA) seems to be as hard as computing A^TA.
  - but randomized estimate is possible *[Cohen 1998]*
- Sampling algorithms are possible too, e.g. diamond sampling for k largest elements of A^TA (or B*C in general) *[Ballard/Kolda/Pinar/Seshadri 2015]*



(a) Sample $(k, i) \propto w_{ki}$ (b) Sample $j \in \mathcal{N}_k^B$ (c) Sample $k' \in \mathcal{N}_i^A$

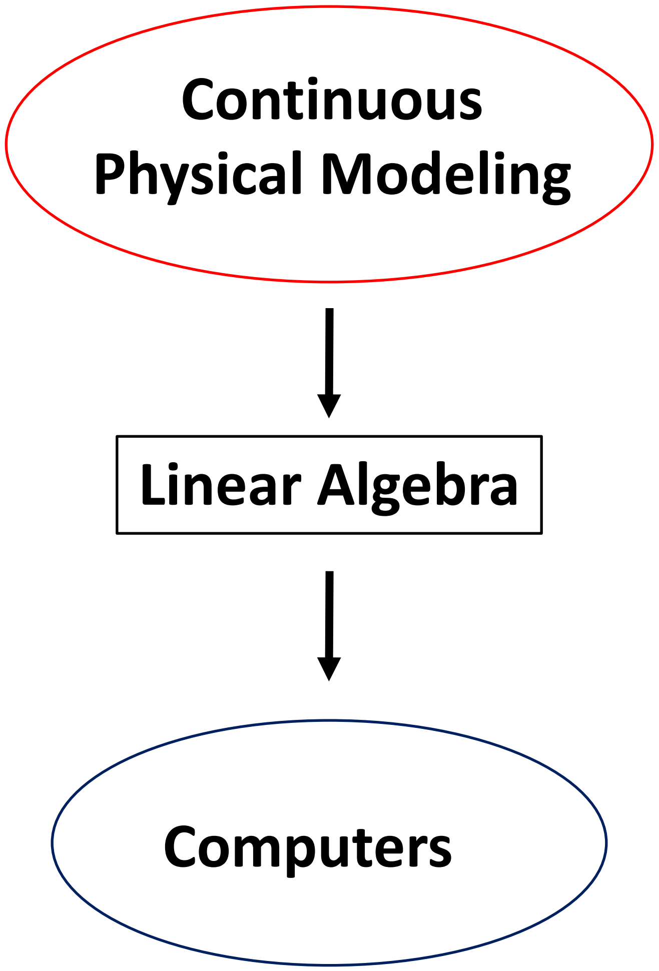
Ballard et al. ICDM 2015

UCSB

Outline

- Elimination, chordal graphs, and treewidth
- A tale of two matroids
- Magic eigenvectors
- Graphs in the language of linear algebra
- A word about A^TA
- Concluding remarks

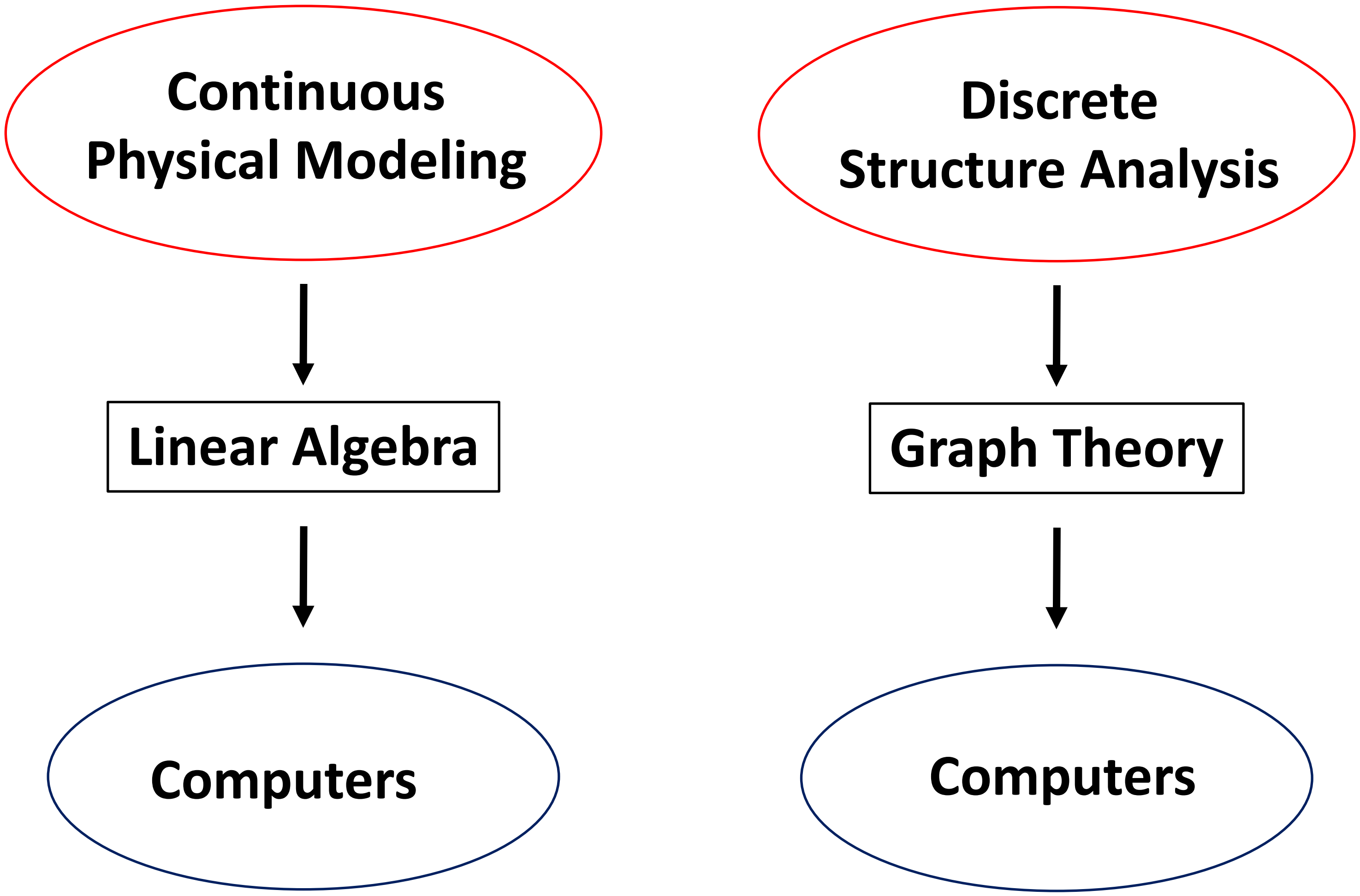
Past 50 Years



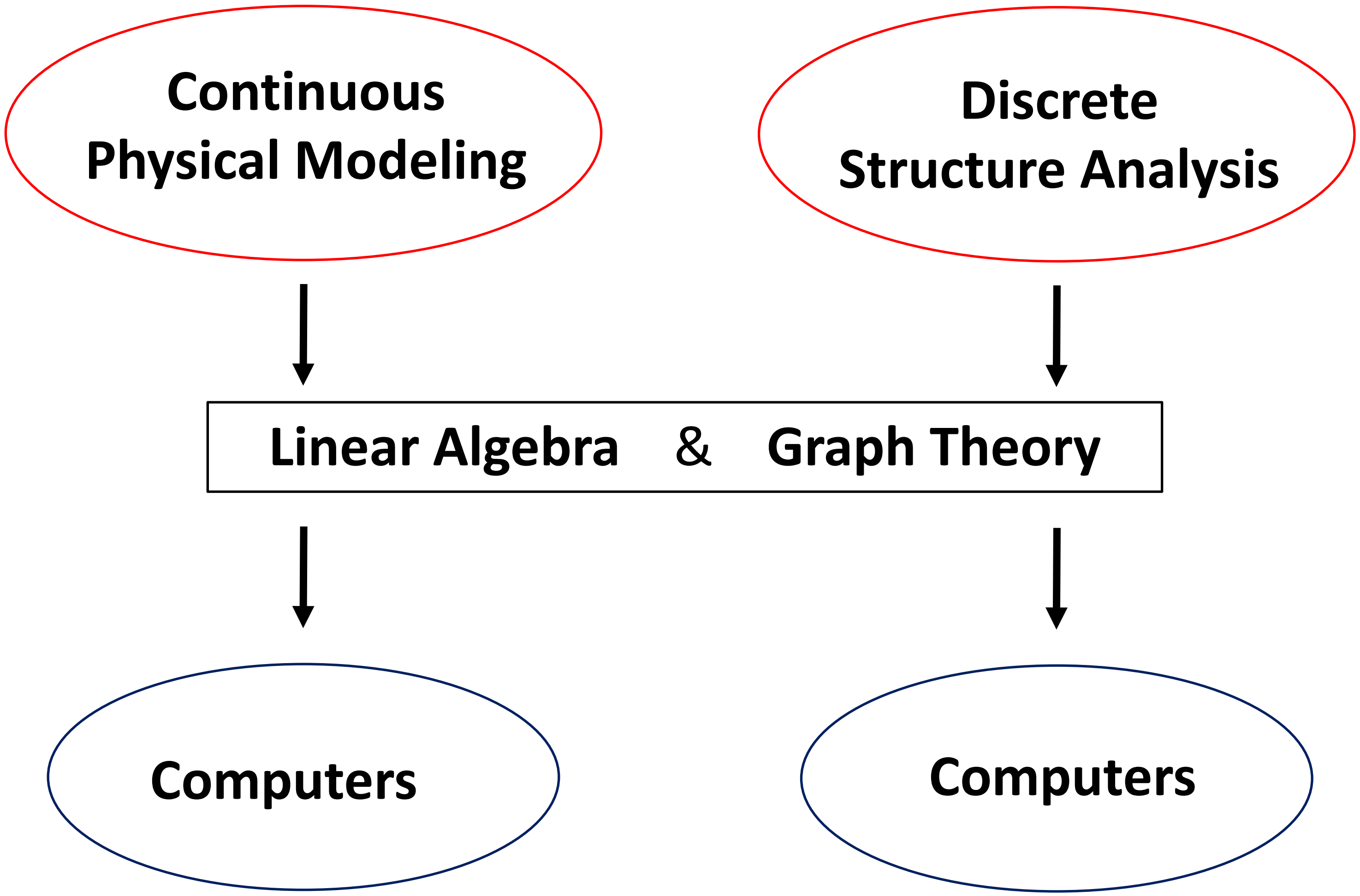
As the "middleware" of scientific computing, linear algebra has given us:

- Mathematical tools
- High-level primitives
- High-quality software libraries
- High-performance kernels for computer architectures
- Interactive environments

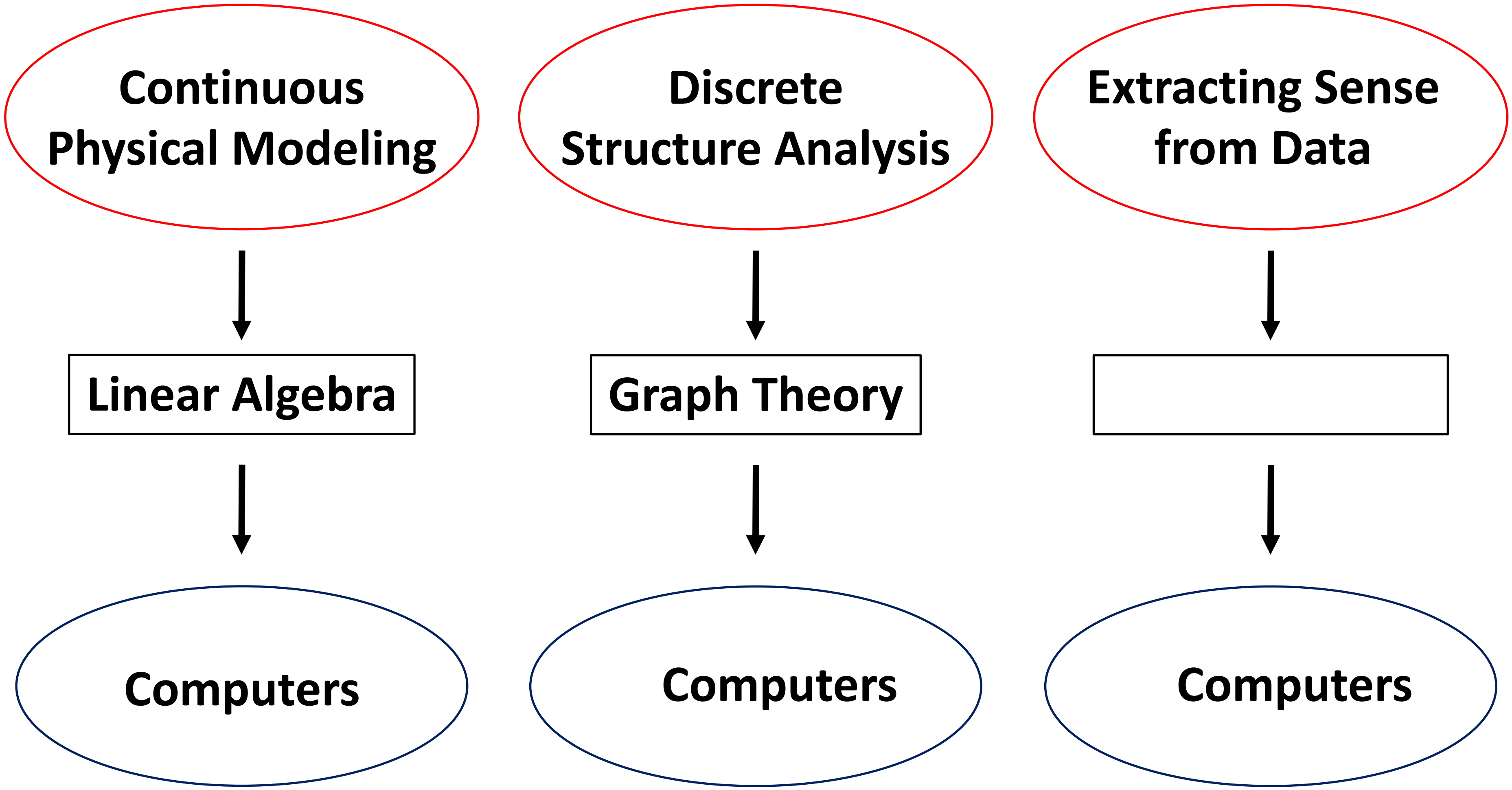
Today


Continuous Physical Modeling
Linear Algebra
Computers
Discrete Structure Analysis
Graph Theory
Computers

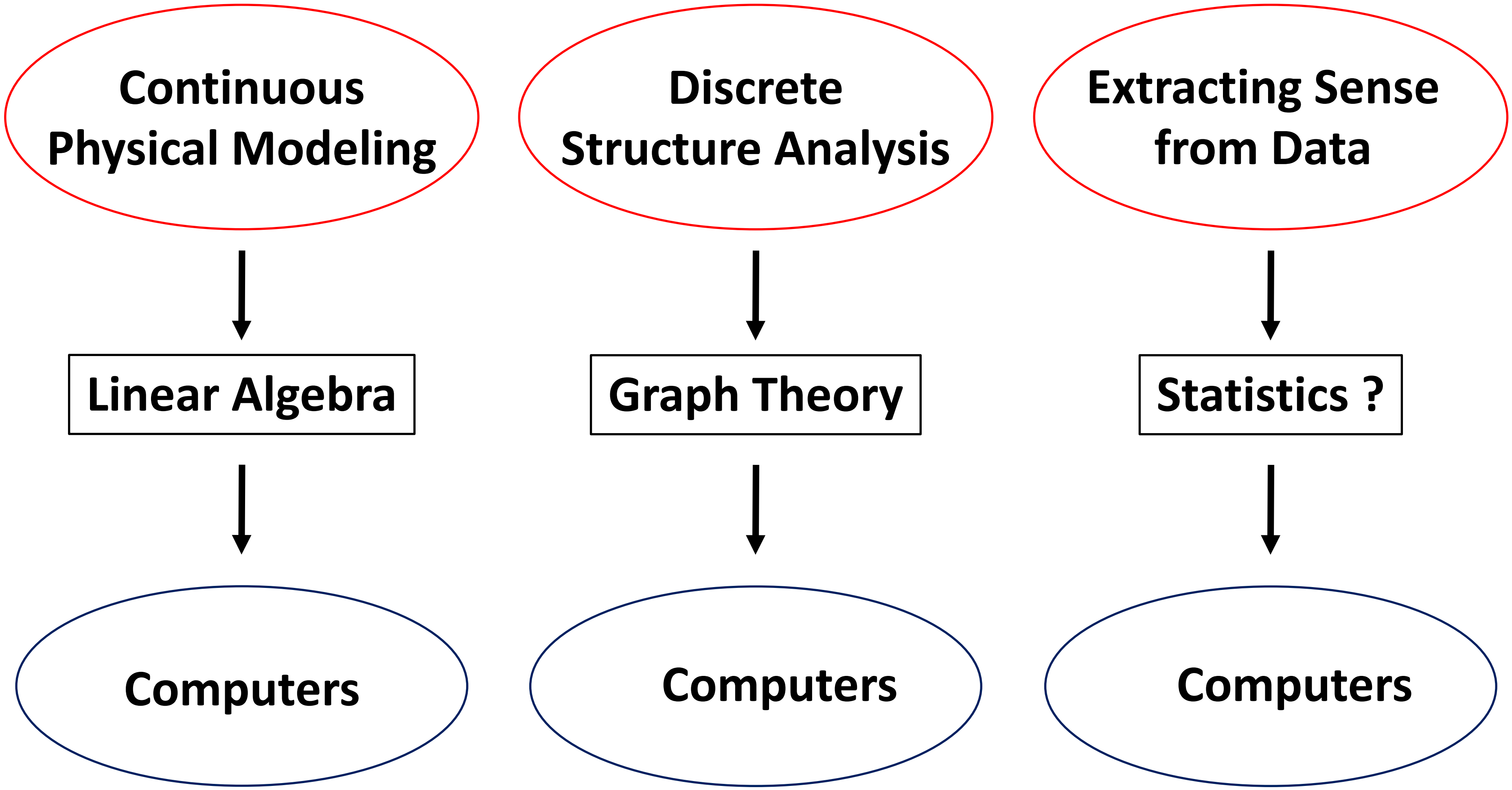
Today


Continuous Physical Modeling
Discrete Structure Analysis
Linear Algebra & Graph Theory
Computers
Computers

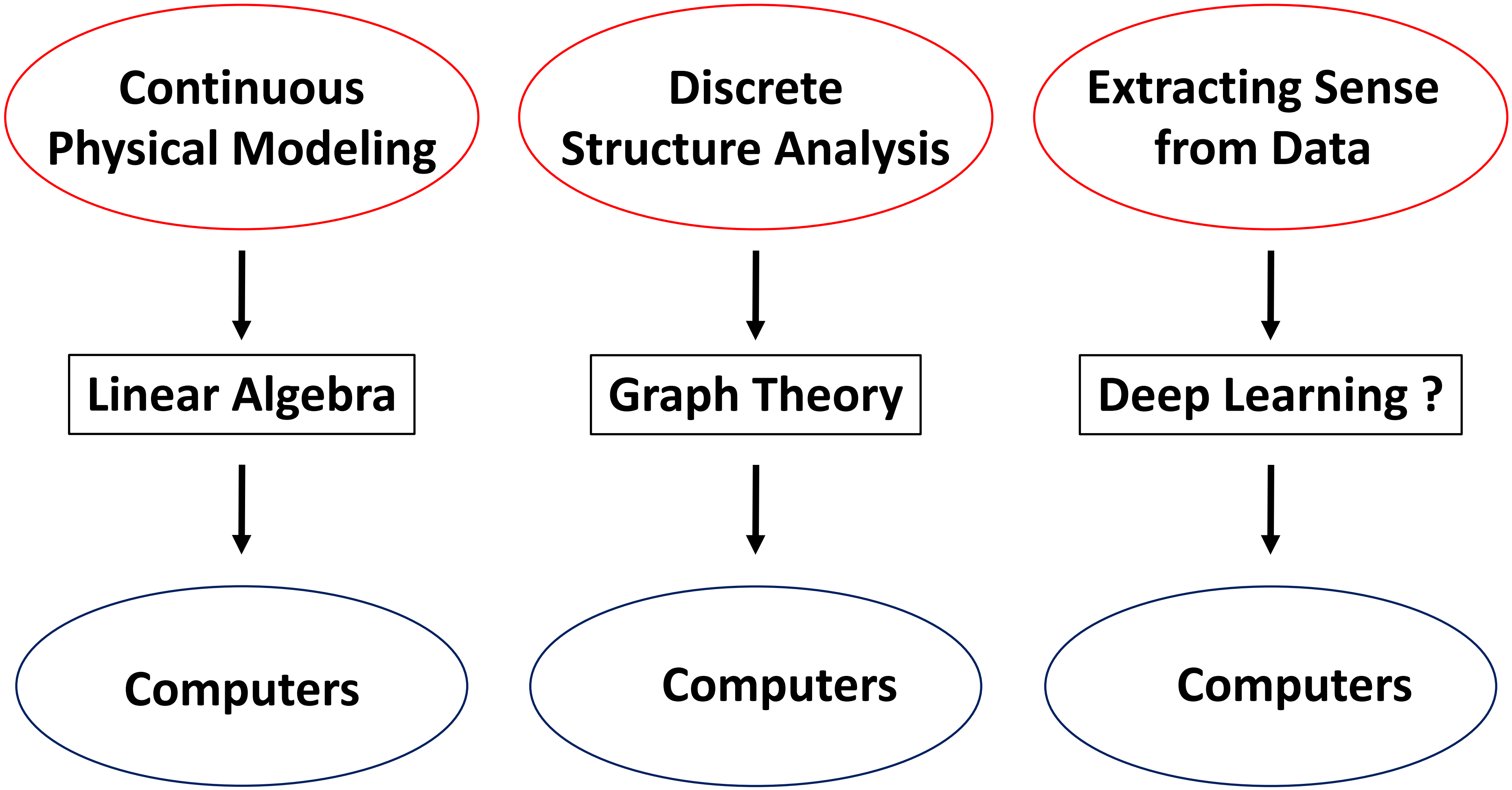
Tomorrow?


Continuous Physical Modeling
Discrete Structure Analysis
Extracting Sense from Data
Linear Algebra
Graph Theory
Computers
Computers
Computers

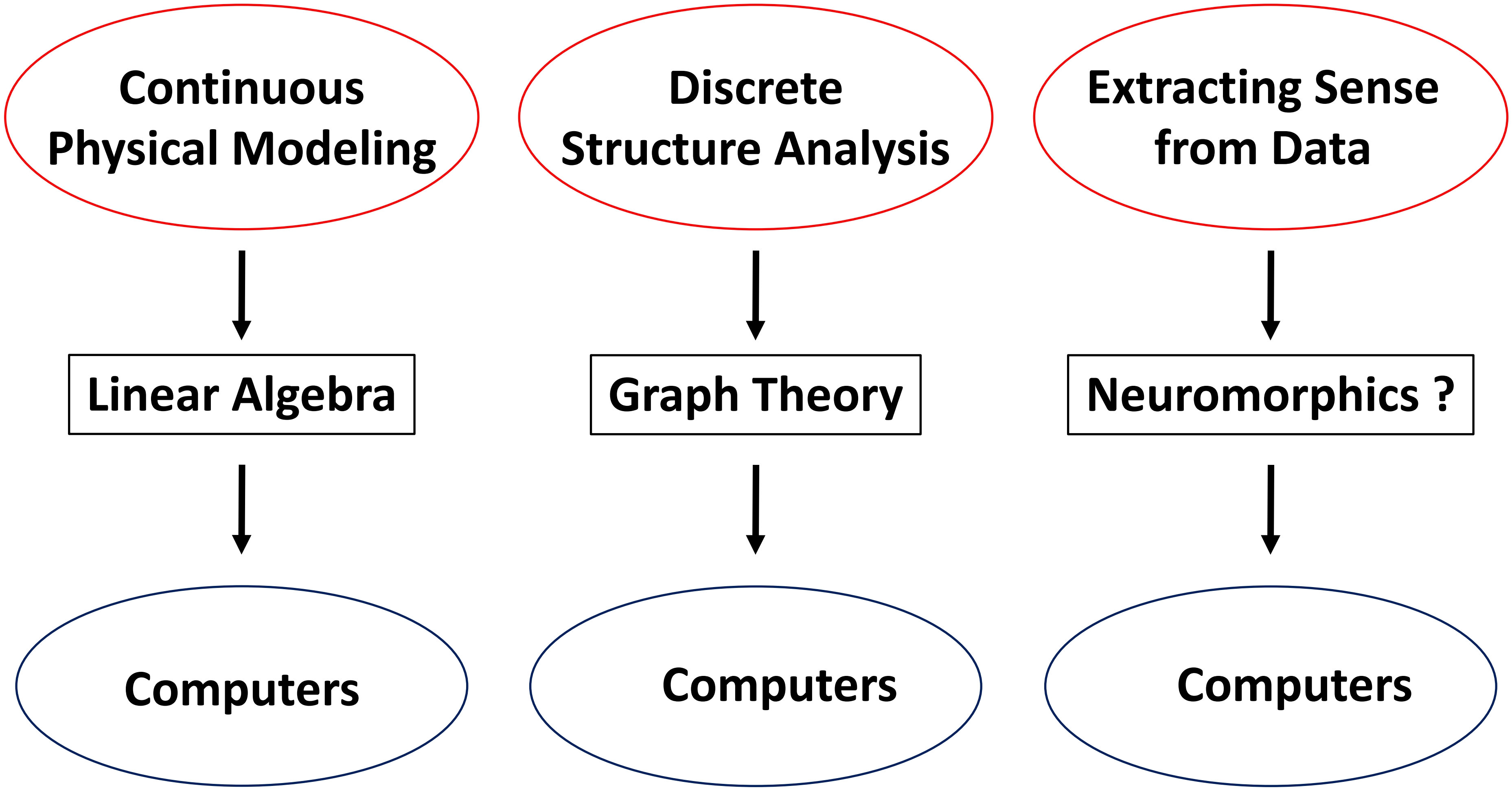
Tomorrow?


Continuous Physical Modeling
Discrete Structure Analysis
Extracting Sense from Data
Linear Algebra
Graph Theory
Statistics ?
Computers
Computers
Computers

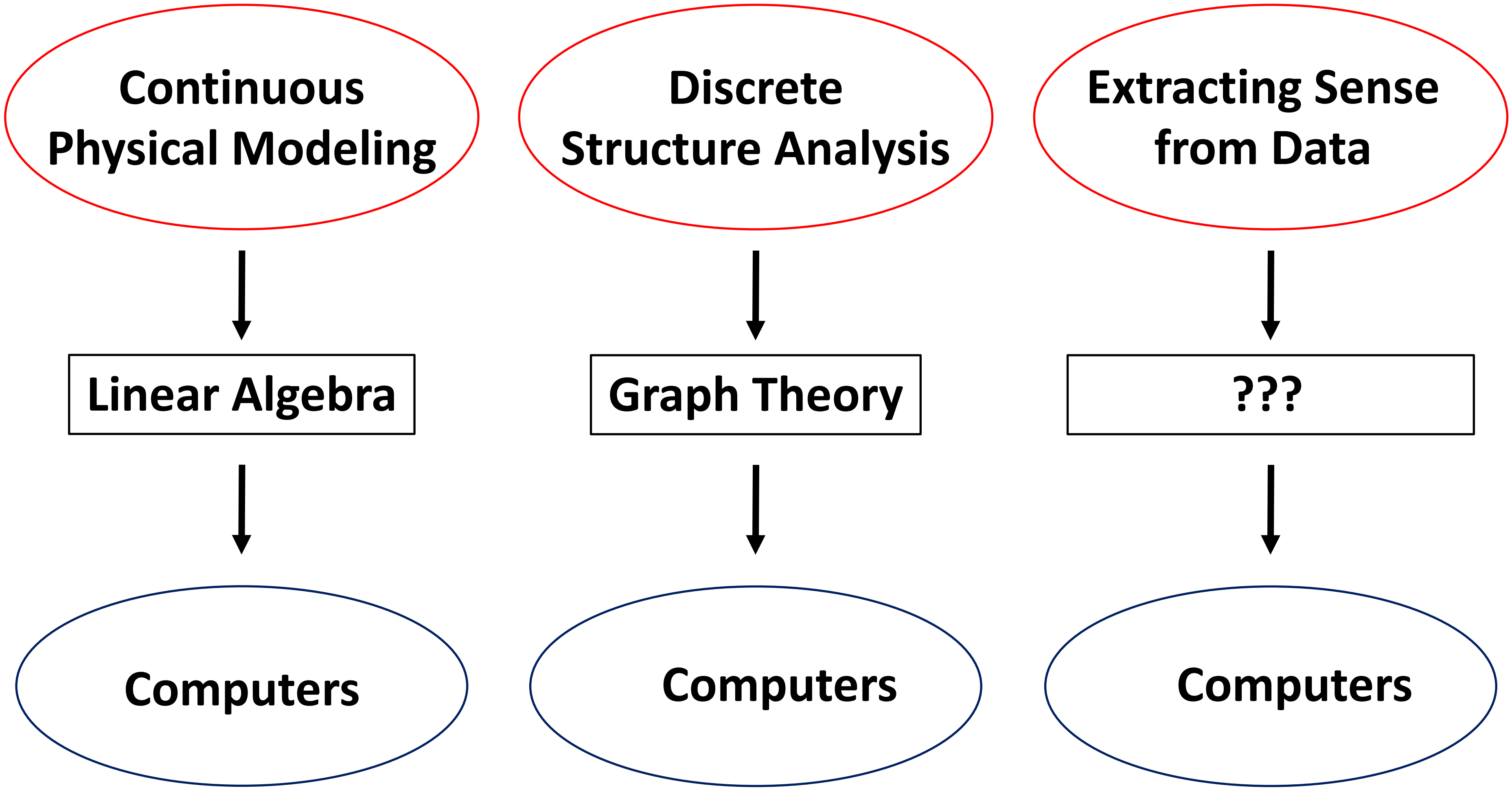
Tomorrow?


Continuous Physical Modeling
Discrete Structure Analysis
Extracting Sense from Data
Linear Algebra
Graph Theory
Deep Learning ?
Computers
Computers
Computers

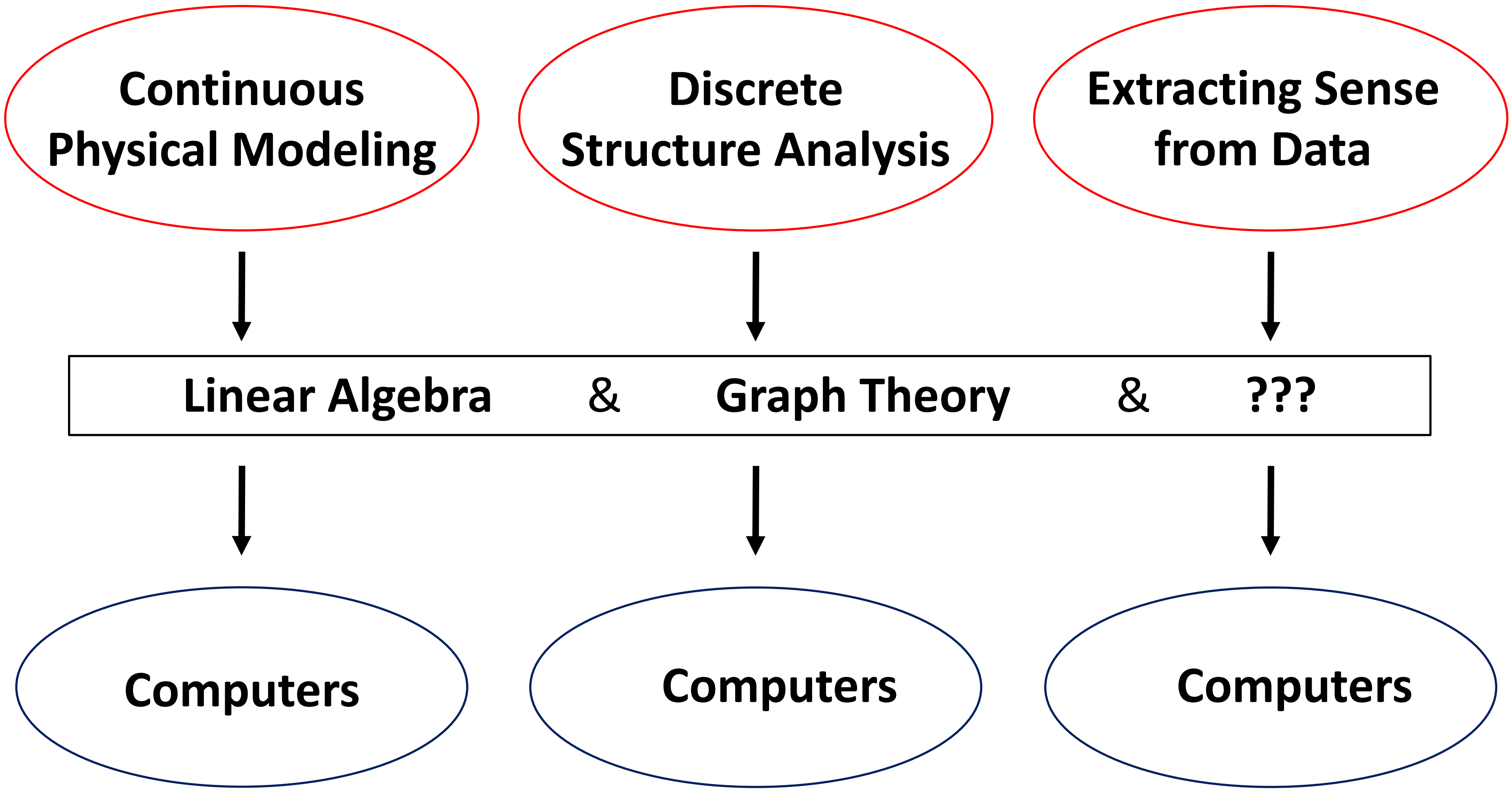
Tomorrow?


Continuous Physical Modeling
Linear Algebra
Computers
Discrete Structure Analysis
Graph Theory
Computers
Extracting Sense from Data
Neuromorphics ?
Computers

Tomorrow?


Continuous Physical Modeling
Discrete Structure Analysis
Extracting Sense from Data
Linear Algebra
Graph Theory
???
Computers
Computers
Computers

Tomorrow?


Continuous Physical Modeling
Discrete Structure Analysis
Extracting Sense from Data
Linear Algebra & Graph Theory & ???
Computers
Computers
Computers

This is a great time to be doing research in CSC…

- What can new ways of looking at the structure of network graphs tell us about $Ax = b$ and $Ax = \lambda x$?
- What else can you do fast on $G(A^TA)$ just from $G(A)$? (Especially with sampling & approximation)
- What will be the middleware for making sense of big data?

UCSB

This is a great time to be doing research in CSC…

- What can new ways of looking at the structure of network graphs tell us about $Ax = b$ and $Ax = \lambda x$?
- What else can you do fast on $G(A^TA)$ just from $G(A)$? (Especially with sampling & approximation)
- What will be the middleware for making sense of big data?

Matrix computation is beginning to repay a 60-year debt to graph algorithms.

It helps to look at things from two directions.

Thanks …

Ariful Azad, David Bader, Lucas Bang, Jon Berry, Eric Boman, Aydin Buluc, Ben Chang, John Conroy, Kevin Deweese, Erika Duriakova, Assefaw Gebremedhin, Shoaib Kamil, Jeremy Kepner, Tammy Kolda, Tristan Konolige, Manoj Kumar, Adam Lugowski, Tim Mattson, Scott McMillan, Henning Meyerhenke, Jose Moreira, Esmond Ng, Lenny Oliker, Weimin Ouyang, Ali Pinar, Alex Pothen, Carey Priebe, Steve Reinhardt, Lijie Ren, Eric Robinson, Viral Shah, Veronika Strnadova-Neely, Blair Sullivan, Shang-Hua Teng, Yun Teng, Sam Williams

… and Intel, Microsoft, NSF, DOE Office of Science

UCSB