Combinatorial Scientific Computing: Tutorial, Experiences, and Challenges

John R. Gilbert
University of California, Santa Barbara

Workshop on Algorithms for Modern Massive Datasets
June 15, 2010

Support: NSF, DARPA, DOE, SGI

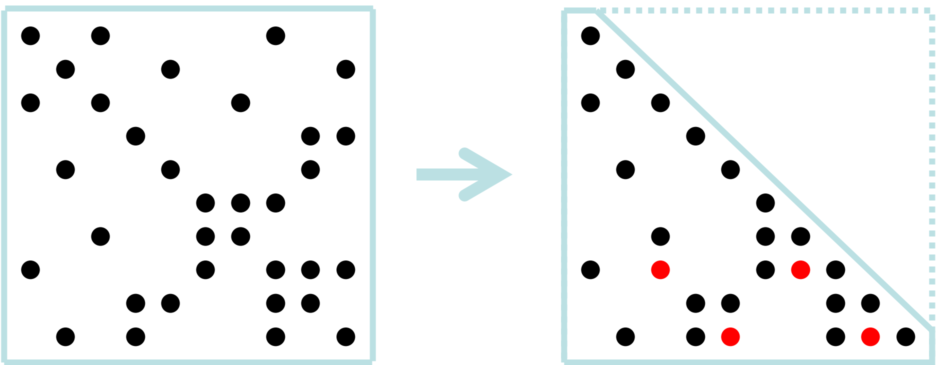
Combinatorial Scientific Computing

"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"

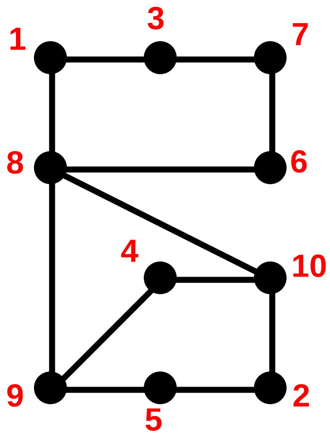
- Harry Markowitz, describing the 1950s work on portfolio theory that won the 1990 Nobel Prize for Economics

UCSB

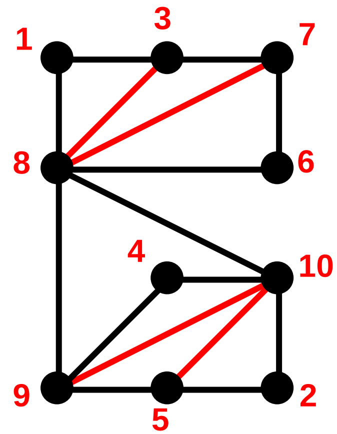
Graphs and Sparse Matrices: Cholesky factorization



Fill: new nonzeros in factor



G(A)

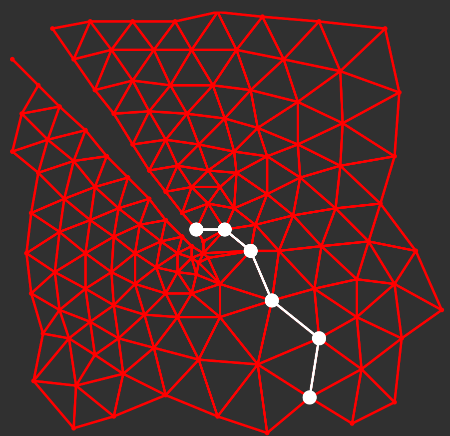


$G^{+}(A)$
[chordal]

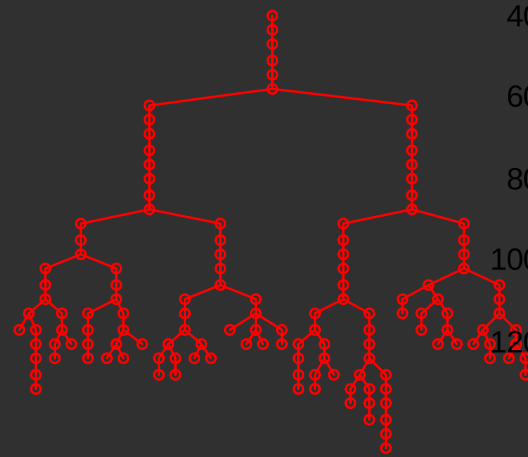
Symmetric Gaussian elimination:

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

Fill-reducing matrix permutations

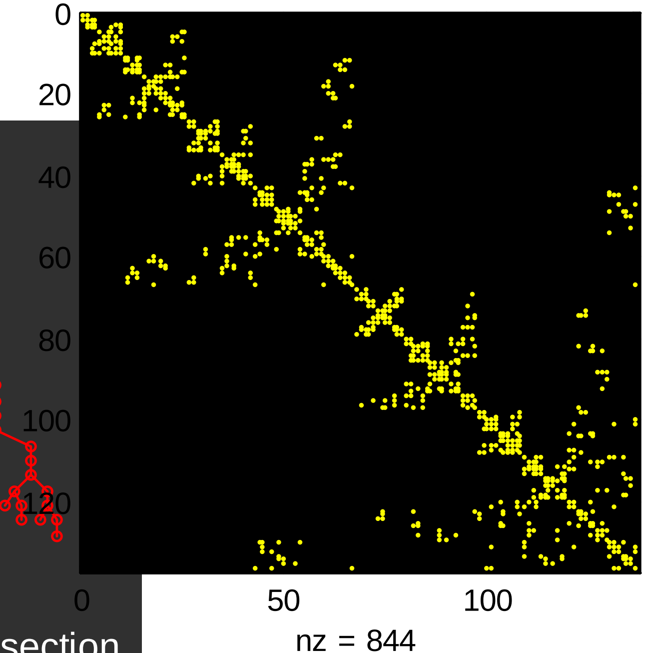


Vertex separator in graph of matrix



Elimination tree with nested dissection

Matrix reordered by nested dissection

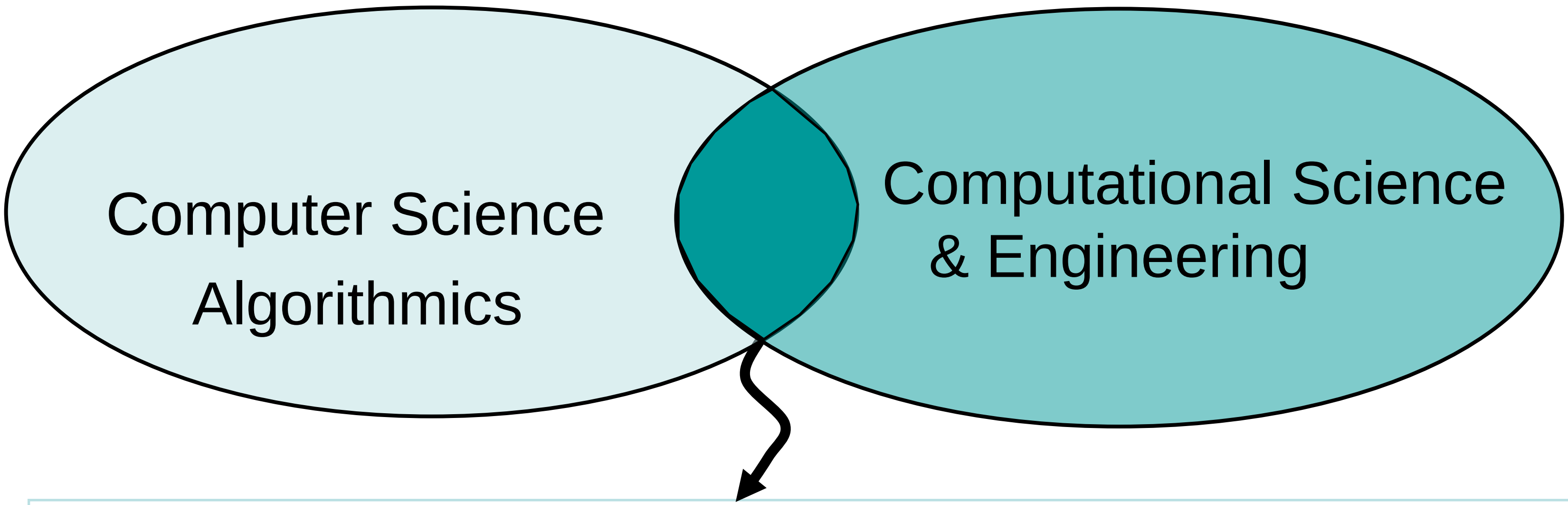


- <u>Theory:</u> approx optimal separators => approx optimal fill and op count
- <u>Orderings:</u> nested dissection, minimum degree, hybrids
- <u>Graph partitioning:</u> spectral, geometric, multilevel

UCSB

Combinatorial Scientific Computing

What's in the intersection?



Combinatorial Scientific Computing:

The development, application and analysis of combinatorial algorithms to enable scientific and engineering computations

CSC: What's in a name?

- Deeper origins in …
  - Sparse direct methods community (1950s & onward)
  - Statistical physics: graphs and Ising models (1940s & 50s)
  - Chemical classification (1800s, Cayley)
  - Common esthetic, techniques, and goals among researchers who were far apart in traditional scientific taxonomy

UCSB

CSC: What's in a name?

- Deeper origins in …
  - Sparse direct methods community (1950s & onward)
  - Statistical physics: graphs and Ising models (1940s & 50s)
  - Chemical classification (1800s, Cayley)
  - Common esthetic, techniques, and goals among researchers who were far apart in traditional scientific taxonomy
- "Combinatorial scientific computing" chosen in 2002
  - After lengthy email discussion among ~ 30 people.
  - Now > 70,000 Google hits for "combinatorial scientific computing"

UCSB

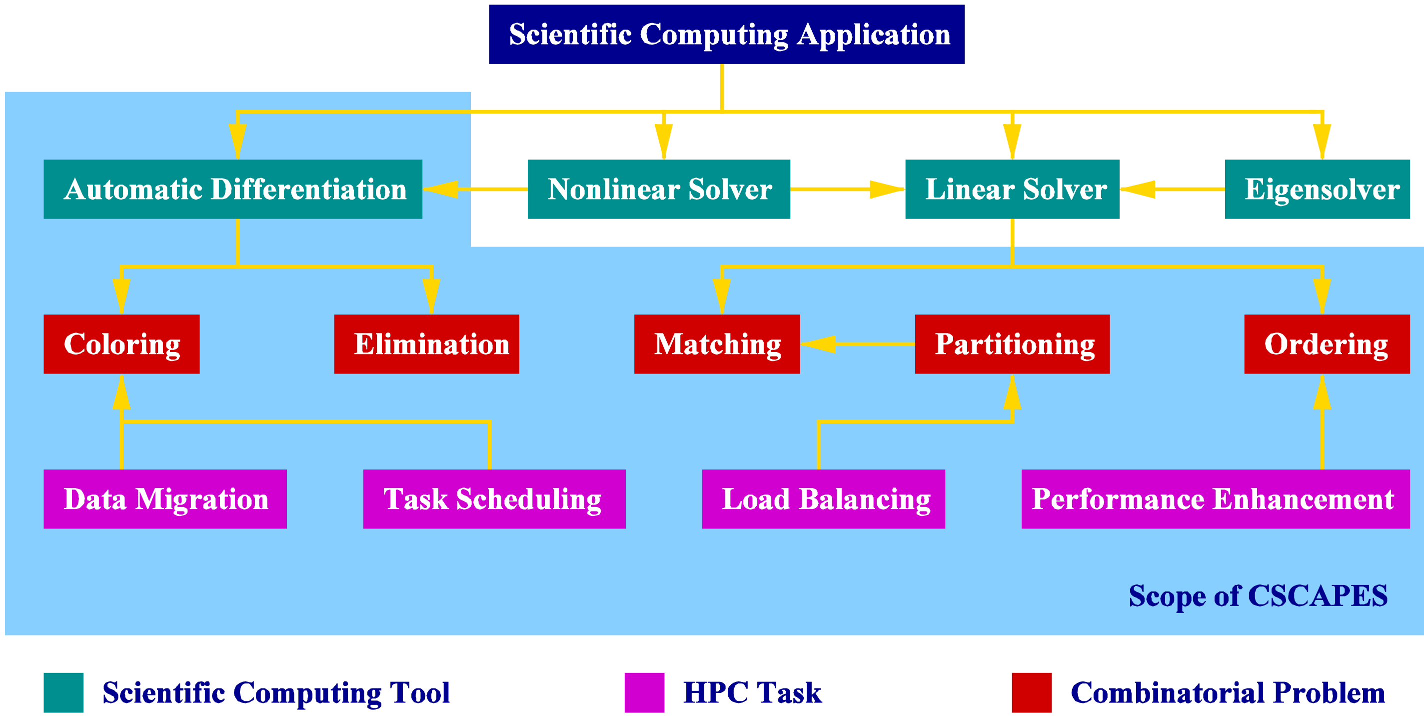
CSC: What's in a name?

- Deeper origins in …
  - Sparse direct methods community (1950s & onward)
  - Statistical physics: graphs and Ising models (1940s & 50s)
  - Chemical classification (1800s, Cayley)
  - Common esthetic, techniques, and goals among researchers who were far apart in traditional scientific taxonomy
- "Combinatorial scientific computing" chosen in 2002
  - After lengthy email discussion among ~ 30 people.
  - Now > 70,000 Google hits for "combinatorial scientific computing"
- Recognition by scientific community & funding agencies
  - 5 major CSC workshops since 2004, plus many minisymposia
  - DOE "CSCapes" institute formed 2006

UCSB

DOE CSCapes Institute

www.cscapes.org

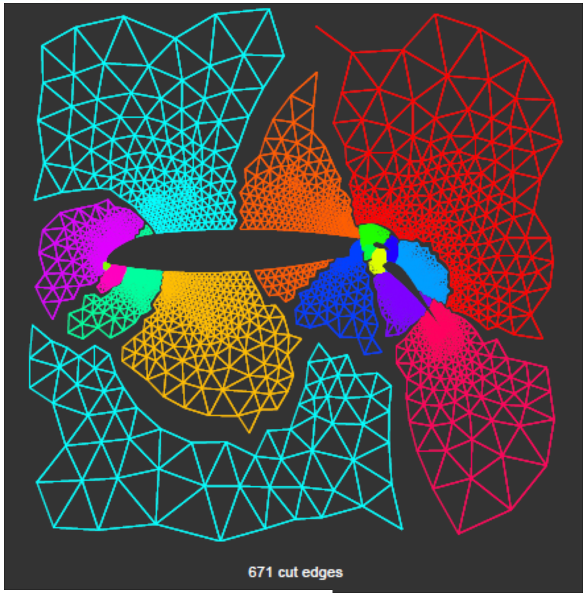


Lead Principal Investigator: Alex Pothen, Purdue

A Brief Tour of Applications

Partitioning data for parallel computation

- <u>Goals</u>: balance load, reduce data movement
- <u>Approaches</u>: geometric, spectral, multilevel
- Geometric meshes are easier than general sparse structures!
- Graphs => hypergraphs, more detailed models
- Partitioning in parallel: how to bootstrap?



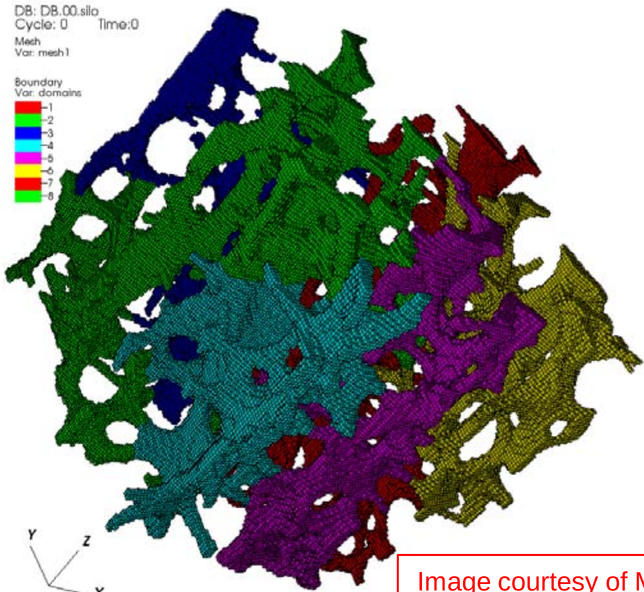
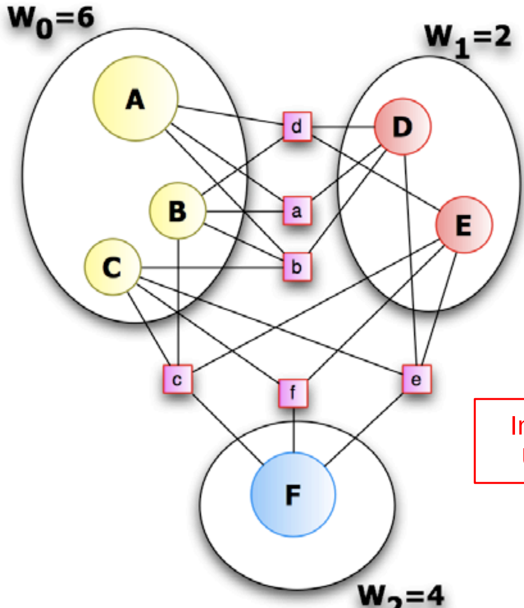


Image courtesy of Mark Adams



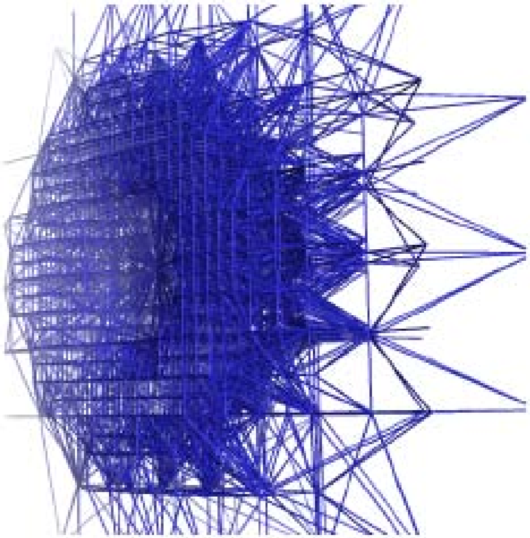
$c_j = 1$ for all edges

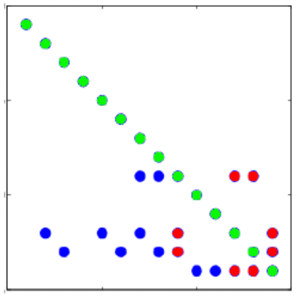
Comm Vol in App = 3 x (2-1) + 3 x (3-1) = 9

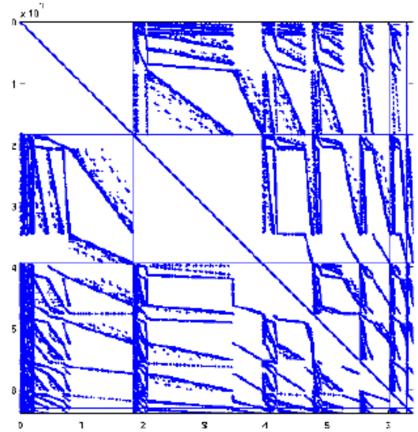
Image courtesy of Umit Catalyurek

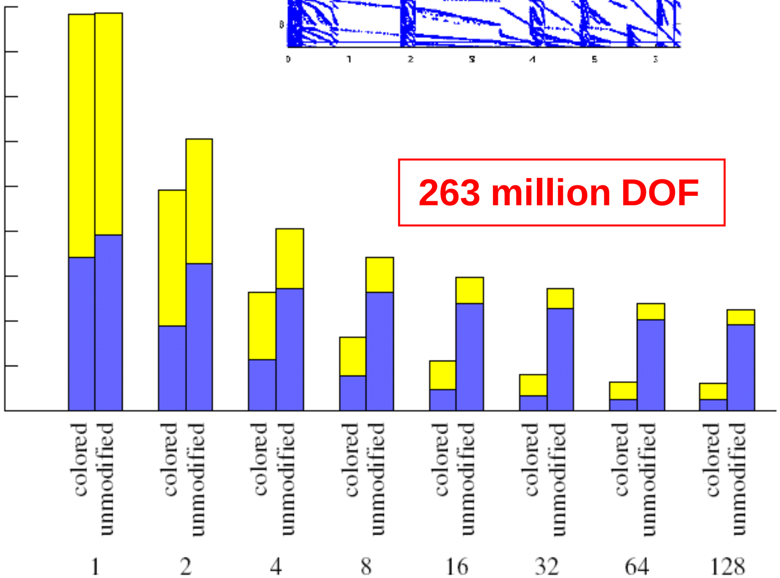
UCSB

Coloring for parallel nonsymmetric preconditioning [Aggarwal et al.]





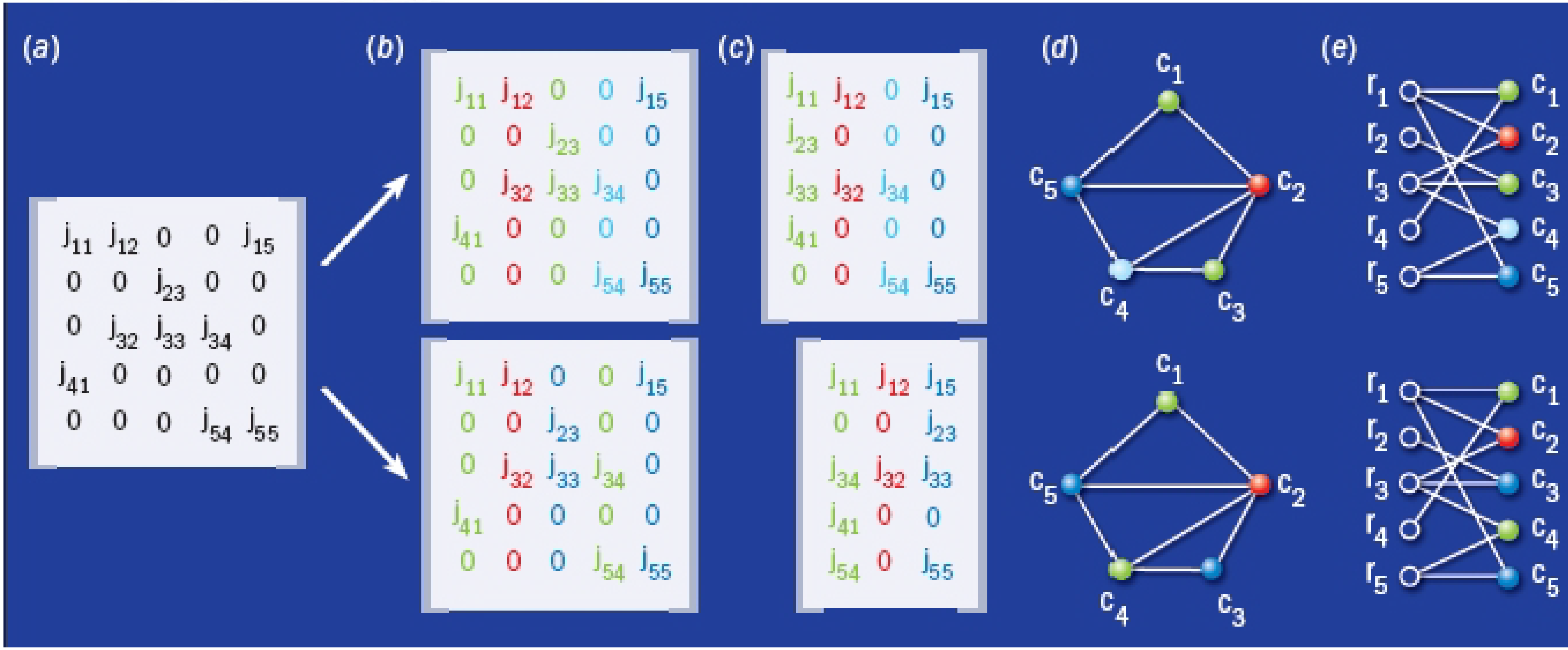




- Level set method for multiphase interface problems in 3D.
- Nonsymmetric-structure, second-order-accurate octree discretization.
- BiCGSTAB preconditioned by parallel triangular solves.

Coloring for evaluation of sparse Jacobians

[Pothen et al., courtesy CSCapes]



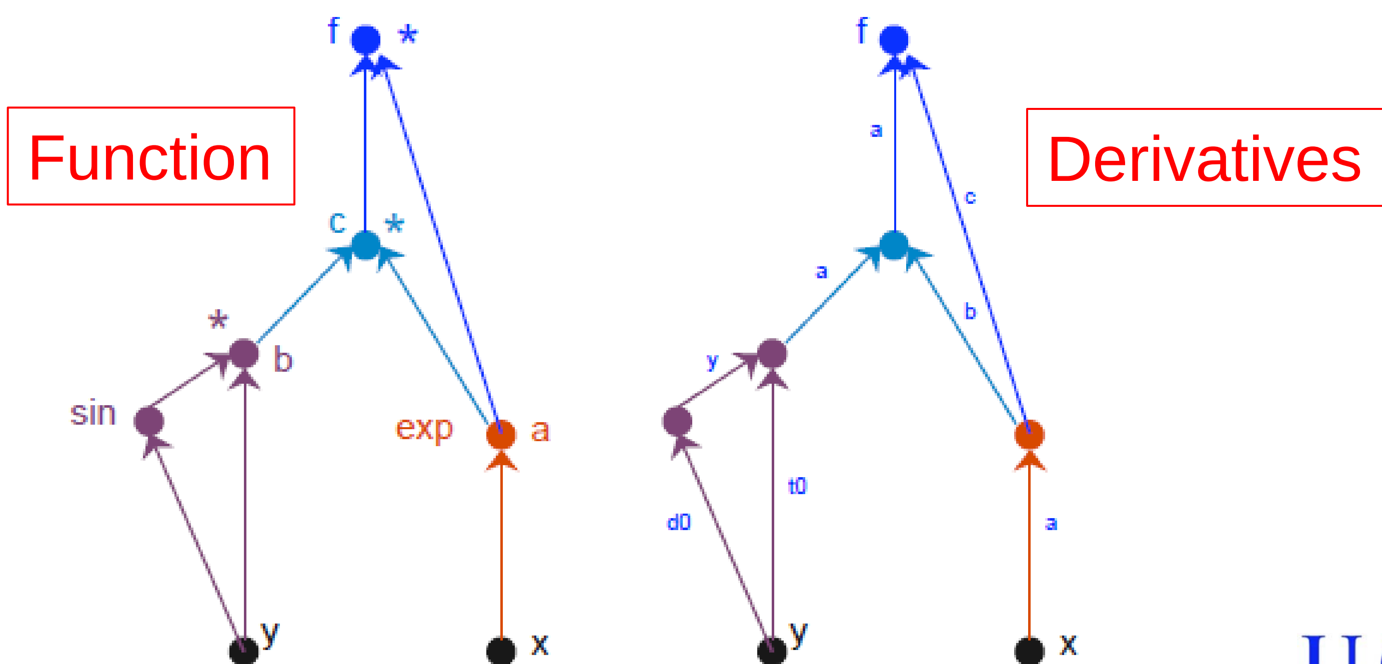
- Goal: compute (sparse) matrix J of partial derivatives, $J(i,j) = \partial y_i / \partial x_j$
- Finite differences give one column at a time …
- … but nonoverlapping columns can be computed together.
- Many variations, extensions, generalizations.

Automatic Differentiation of Programs

[Hovland et al., courtesy CSCapes]

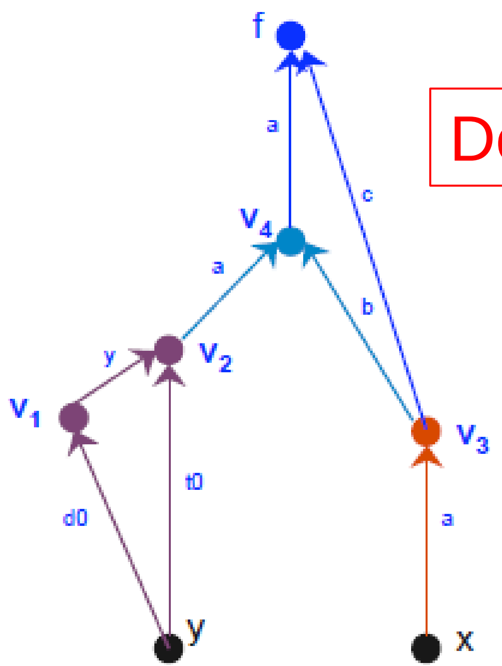
- Technique to convert a (complicated) program that computes $f(x)$ into a program that computes $\partial f_i / \partial x_j$ for all i and j
- Represent a computation as a DAG; vertices are elementary operations
- Label edges with partial derivatives of elementary ops

```
t0 = sin(y)
d0 = cos(y)
b = t0*y
a = exp(x)
c = a*b
f = a*c
```



Automatic Differentiation

- Label edges with partial derivatives of elementary ops
- Using the chain rule, eliminate internal vertices
- End up with partial derivatives of outputs with respect to inputs

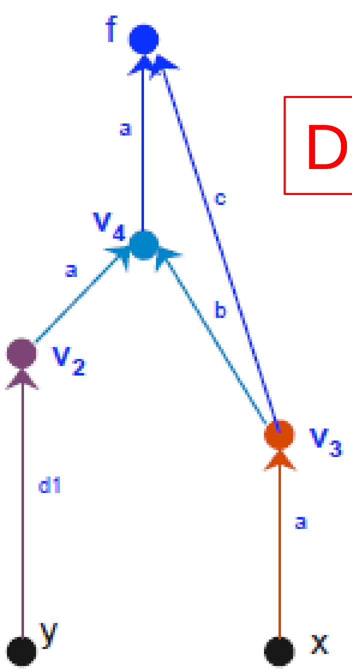


Derivatives

t0 = sin(y)
d0 = cos(y)
b = t0*y
a = exp(x)
c = a*b
f = a*c

Automatic Differentiation

- Label edges with partial derivatives of elementary ops
- Using the chain rule, eliminate internal vertices
- End up with partial derivatives of outputs with respect to inputs

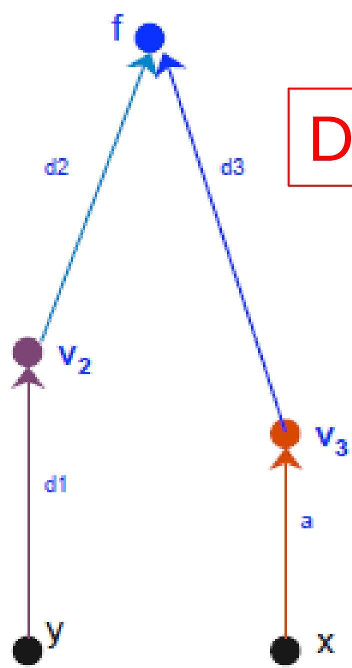


Derivatives

t0 = sin(y)
d0 = cos(y)
b = t0*y
a = exp(x)
c = a*b
f = a*c
d1 = t0 + d0*y

Automatic Differentiation

- Label edges with partial derivatives of elementary ops
- Using the chain rule, eliminate internal vertices
- End up with partial derivatives of outputs with respect to inputs

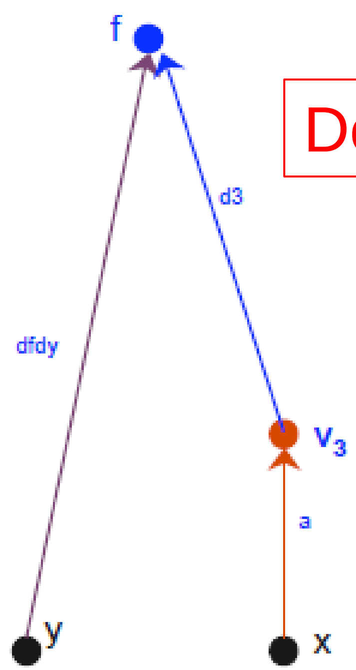


Derivatives

t0 = sin(y)
d0 = cos(y)
b = t0*y
a = exp(x)
c = a*b
f = a*c
d1 = t0 + d0*y
d2 = a*a
d3 = c + b*a

Automatic Differentiation

- Label edges with partial derivatives of elementary ops
- Using the chain rule, eliminate internal vertices
- End up with partial derivatives of outputs with respect to inputs



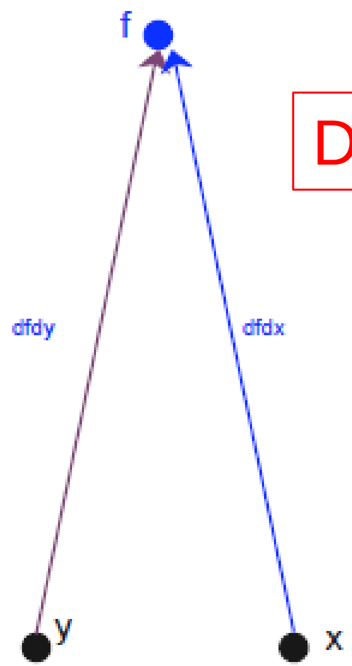
Derivatives

t0 = sin(y)
d0 = cos(y)
b = t0*y
a = exp(x)
c = a*b
f = a*c
d1 = t0 + d0*y
d2 = a*a
d3 = c + b*a
dfdy = d1*d2

Automatic Differentiation

www.autodiff.org

- Used in practice on very large programs => large computational graphs
- Work depends on elimination order; best order is NP-hard
- Checkpointing to trade time for memory; many combinatorial problems



Derivatives

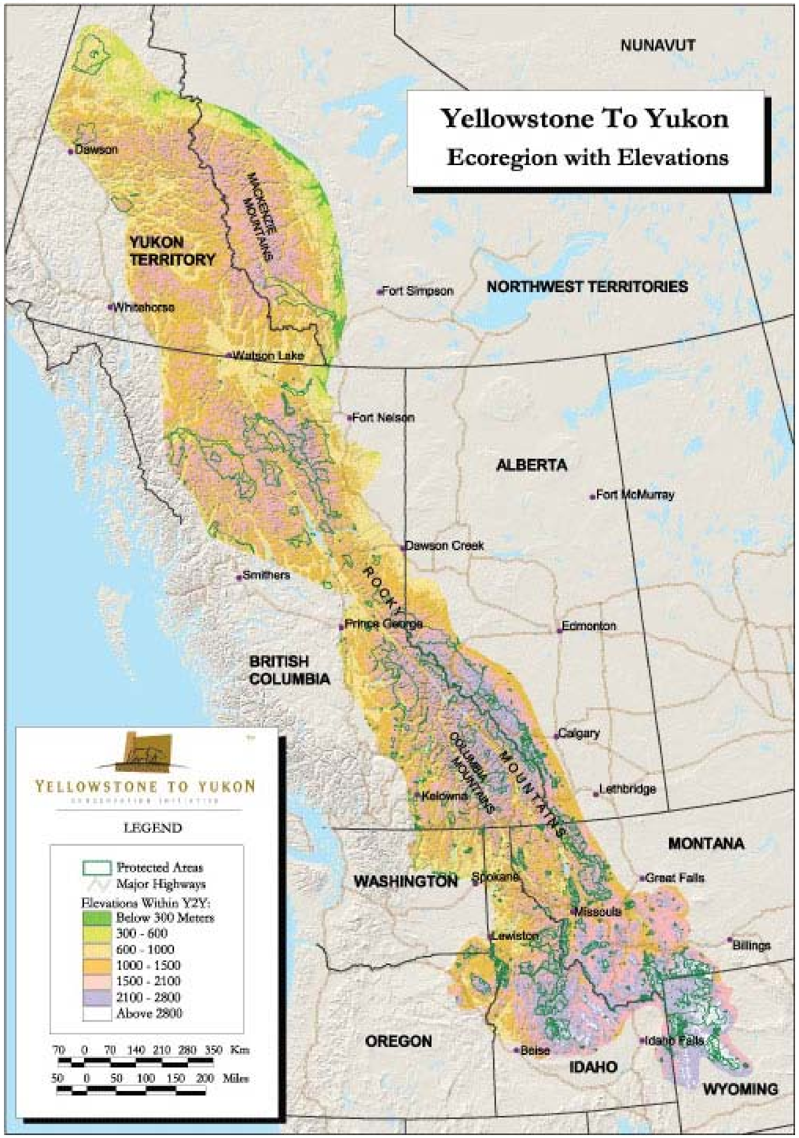
```
t0 = sin(y)
d0 = cos(y)
b = t0*y
a = exp(x)
c = a*b
f = a*c
d1 = t0 + d0*y
d2 = a*a
d3 = c + b*a
dfdy = d1*d2
dfdx = a*d3
```

5 mults 2 adds

Landscape connectivity modeling

[Shah et al.]

- Habitat quality, gene flow, corridor identification, conservation planning
- Pumas in southern California: 12 million nodes, < 1 hour
- Targeting larger problems: Yellowstone-to-Yukon corridor



Figures courtesy of Brad McRae

UCSB

Circuitscape [McRae, Shah]

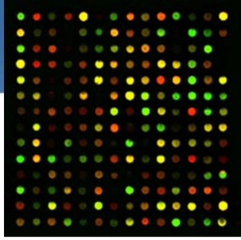
- Predicting gene flow with resistive networks
- Matlab, Python, and Star-P (parallel) implementations
- Combinatorics:
  - Initial discrete grid: ideally 100m resolution (for pumas)
  - Partition landscape into connected components
  - Graph contraction: habitats become nodes in resistive network
- Numerics:
  - Resistance computations for pairs of habitats in the landscape
  - Iterative linear solvers invoked via Star-P: Hypre (PCG+AMG)

UCSB

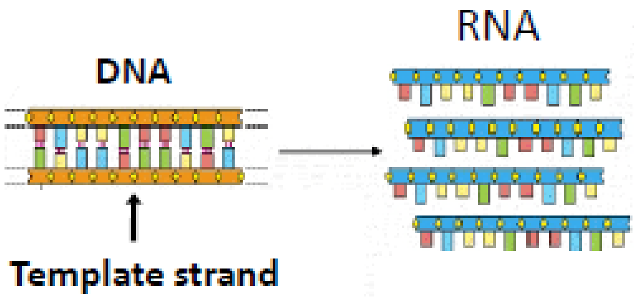
Reverse-engineering genetic transcription

[Abdur-Rahim, dePuit, Yuraszeck after Liao etc.]

Microarray ->



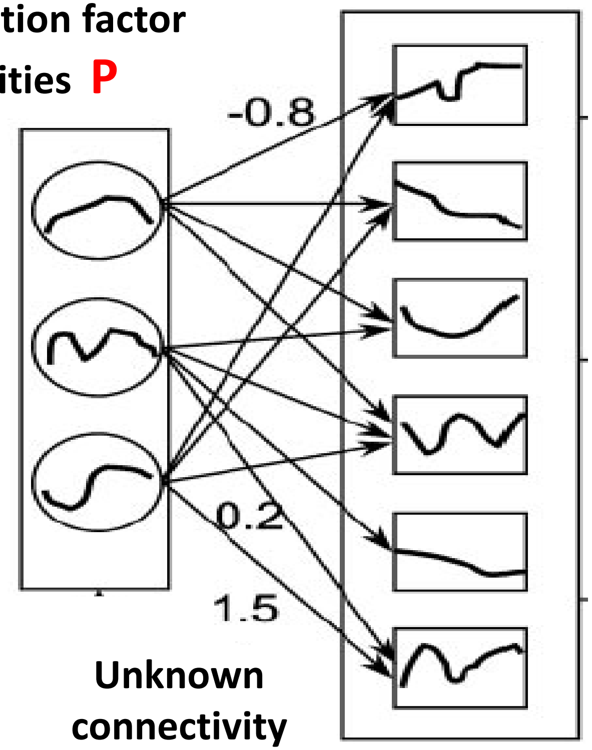
DNA makes RNA, regulated by proteins called transcription factors.



- How strongly does each transcription factor activate or repress expression of each gene?
- Factorize observation matrix **E** as **P** × **A**
- Topological constraints on **A**
- Possible nonnegativity constraint on **P**
- Regularized alternating least squares, etc.

Unknown transcription factor activities P

Observed gene expression data E



Unknown connectivity strengths A

Liao et al. 2003, PNAS

UCSB

A Few Challenges in Combinatorial Scientific Computing

The Challenge of Architecture and Algorithms

UCSB

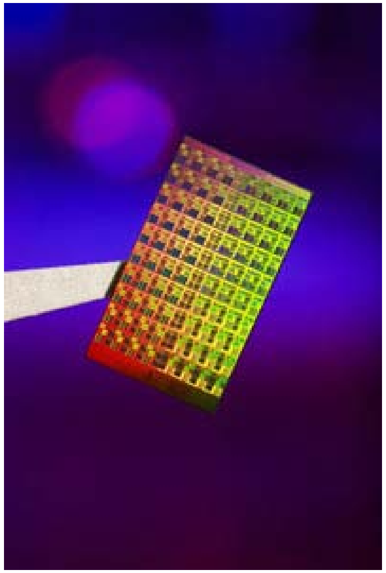
The Architecture & Algorithms Challenge

Oak Ridge / Cray Jaguar

> 1.75 PFLOPS

Two Nvidia 8800 GPUs

> 1 TFLOPS



Intel 80-core chip

> 1 TFLOPS

- Parallelism is no longer optional…
- … in every part of a computation.

UCSB

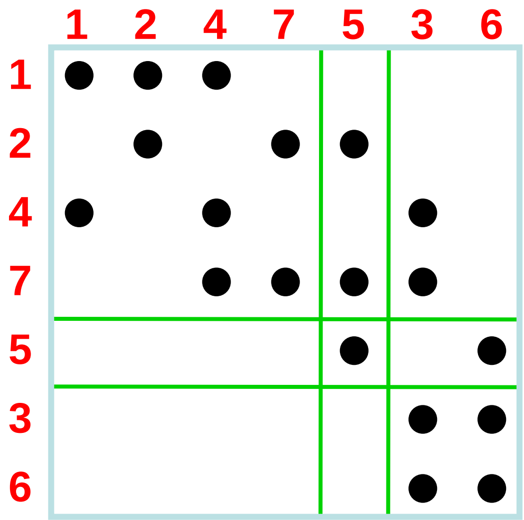
High-performance architecture

- Most high-performance computer designs allocate resources to optimize Gaussian elimination on large, dense matrices.
- Originally, because linear algebra is the middleware of scientific computing.
- Nowadays, largely for bragging rights.

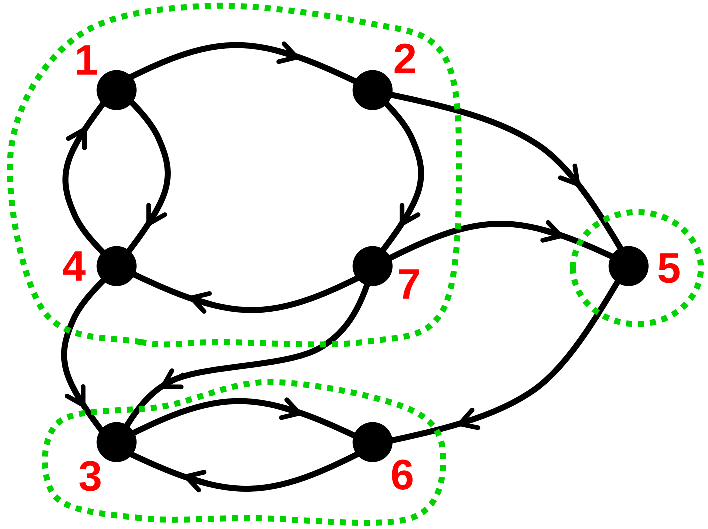
$$P\,A = L \times U$$

Rank	Site	Manufacturer	Computer
1	DOE/NNSA/LANL	IBM	Roadrunner - BladeCen QS22/LS21
2	Oak Ridge National Laboratory	Cray Inc.	Jaguar - Cray XT5 QC 2 GHz
3	NASA/Ames Research Center/NAS	SGI	Pleiades - SGI Altix ICE 8200EX
4	DOE/NNSA/LLNL	IBM	eServer Blue Gene Solu
5	Argonne National Laboratory	IBM	Blue Gene/P Solution
6	Texas Advanced Computing Center/ Univ. of Texas	Sun	Ranger - SunBlade x64
7	NERSC/LBNL	Cray Inc.	Franklin - Cray XT4

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]

Architectural impact on algorithms

Matrix multiplication: C = A * B

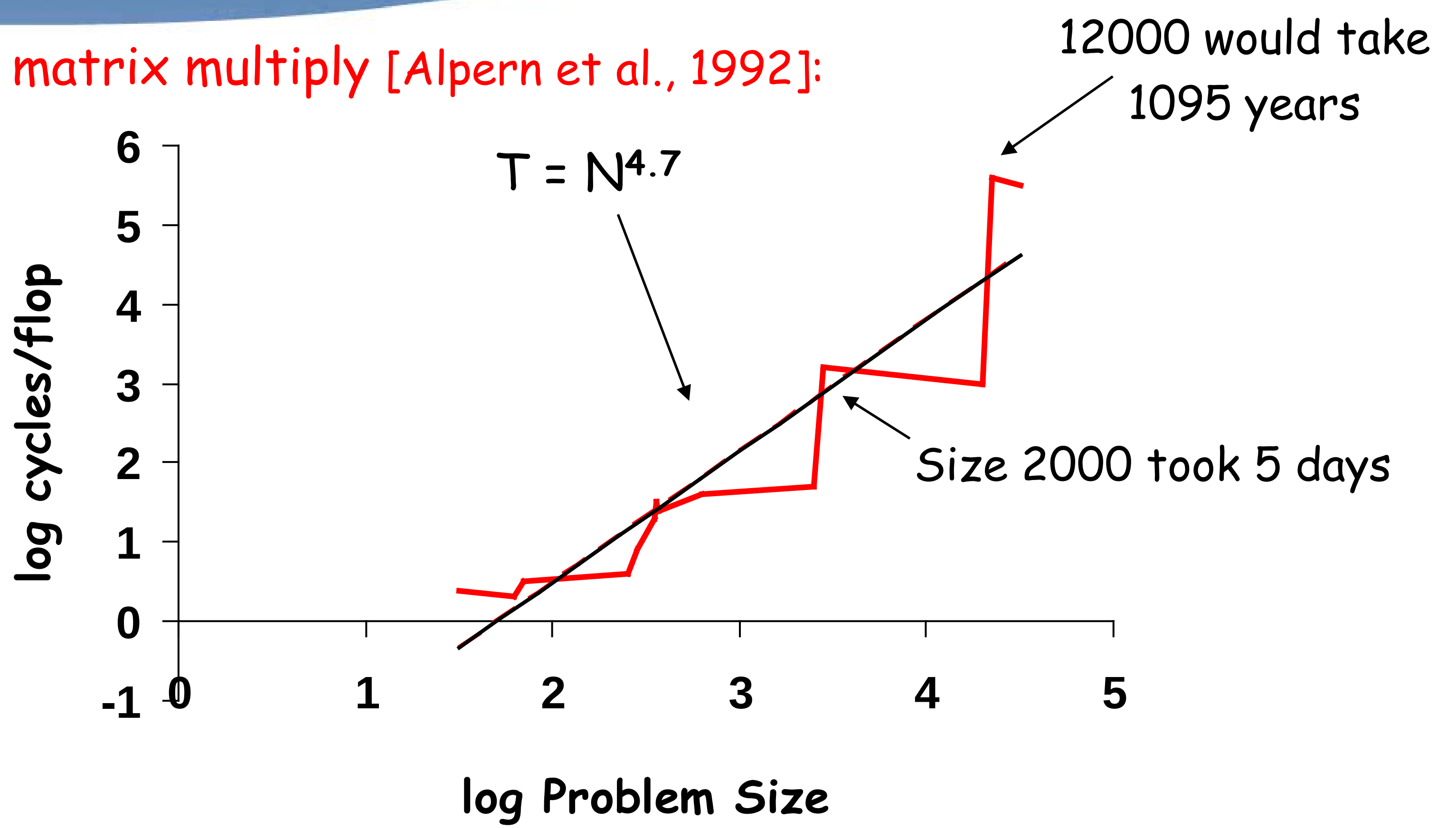
```
C = 0;
for  i = 1 : n
   for  j = 1 : n
      for  k = 1 : n
         C(i,j) = C(i,j) + A(i,k) * B(k,j);
```

$O(n^3)$ operations

UCSB

Architectural impact on algorithms

Naïve 3-loop matrix multiply [Alpern et al., 1992]:



Naïve algorithm is $O(N^5)$ time under UMH model.

BLAS-3 DGEMM and recursive blocked algorithms are $O(N^3)$.

Diagram from Larry Carter

The architecture & algorithms challenge

- A big opportunity exists for computer architecture to influence combinatorial algorithms.

- (Maybe even vice versa.)

The Challenge of Primitives

An analogy?

Continuous physical modeling

↓

Linear algebra

↓

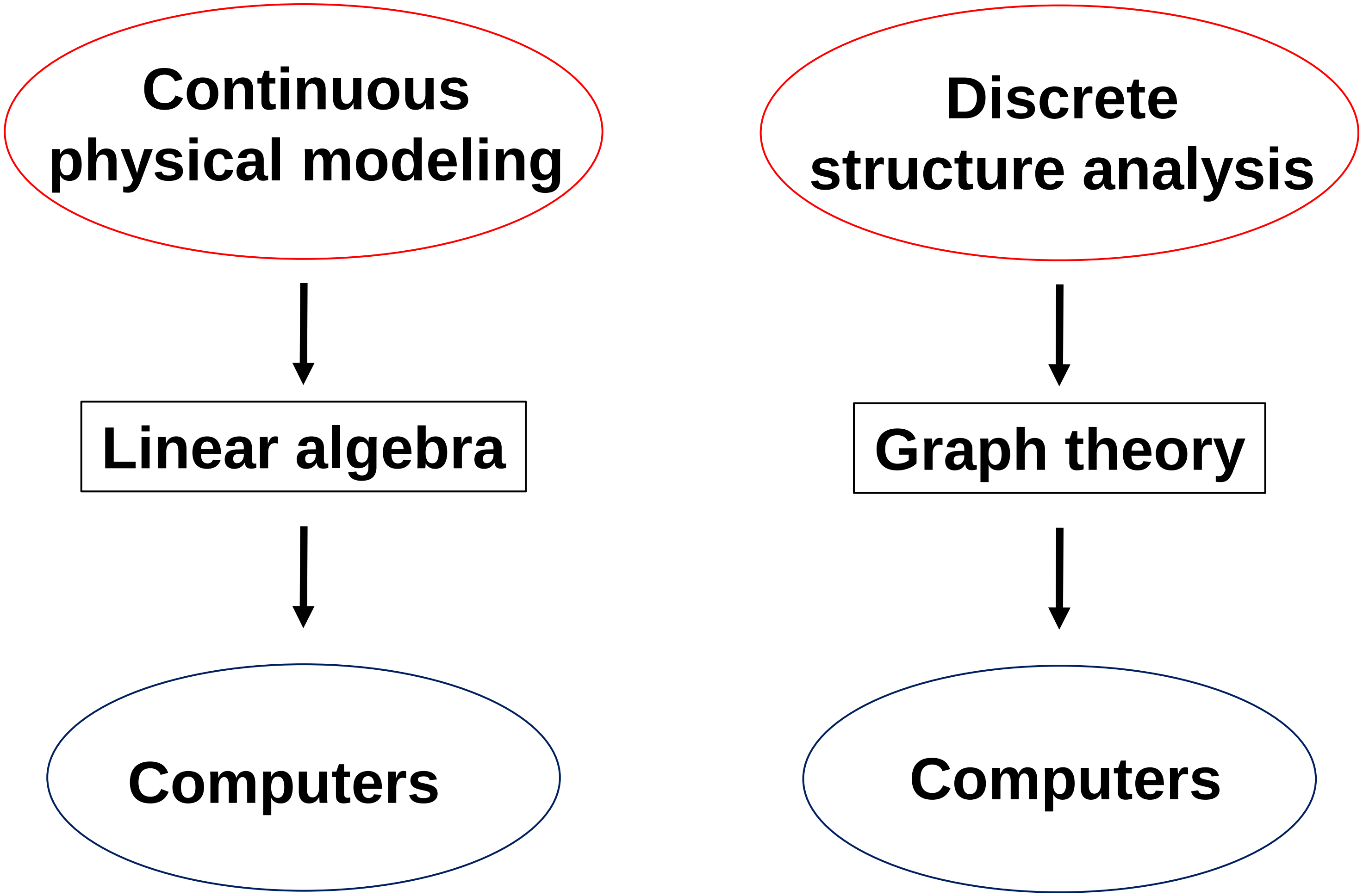
Computers

As the "middleware" of scientific computing, linear algebra has supplied or enabled:

- Mathematical tools
- "Impedance match" to computer operations
- High-level primitives
- High-quality software libraries
- Ways to extract performance from computer architecture
- Interactive environments

UCSB

An analogy?


Continuous physical modeling
Linear algebra
Computers
Discrete structure analysis
Graph theory
Computers

An analogy? Well, we're not there yet ….

- √ Mathematical tools
- ? "Impedance match" to computer operations
- ? High-level primitives
- ? High-quality software libs
- ? Ways to extract performance from computer architecture
- ? Interactive environments

Discrete structure analysis

Graph theory

Computers

Primitives should…

- Supply a common notation to express computations
- Have broad scope but fit into a concise framework
- Allow programming at the appropriate level of abstraction and granularity
- Scale seamlessly from desktop to supercomputer
- Hide architecture-specific details from users

Frameworks for graph primitives

Many possibilities; none completely satisfactory;

little work on common frameworks or interoperability.

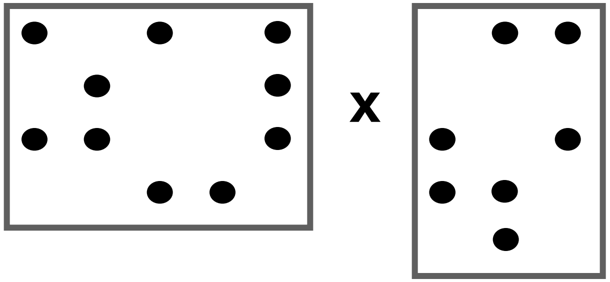
- Visitor-based, distributed-memory: PBGL
- Visitor-based, multithreaded: MTGL
- Heterogeneous, tuned kernels: SNAP
- Scan-based vectorized: NESL
- Map-reduce: lots of visibility
- Sparse array-based: Matlab *P-KDT, CBLAS

UCSB

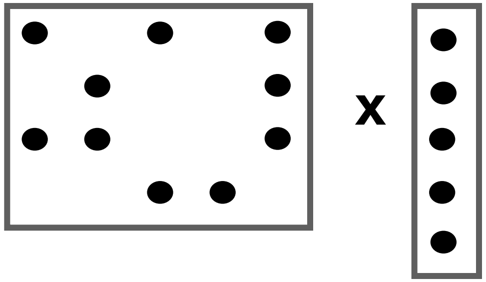
The Case for Sparse Matrices

Sparse array-based primitives

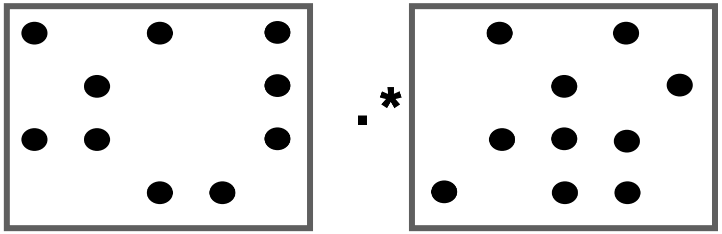
Sparse matrix-matrix multiplication (SpGEMM)



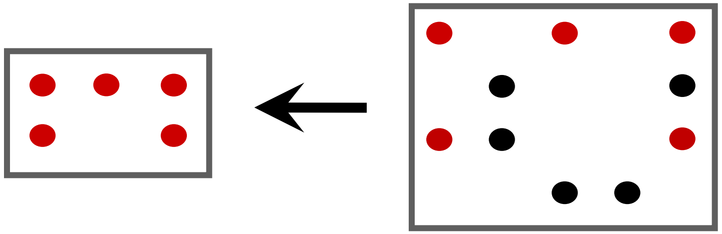
Sparse matrix-dense vector multiplication



Element-wise operations

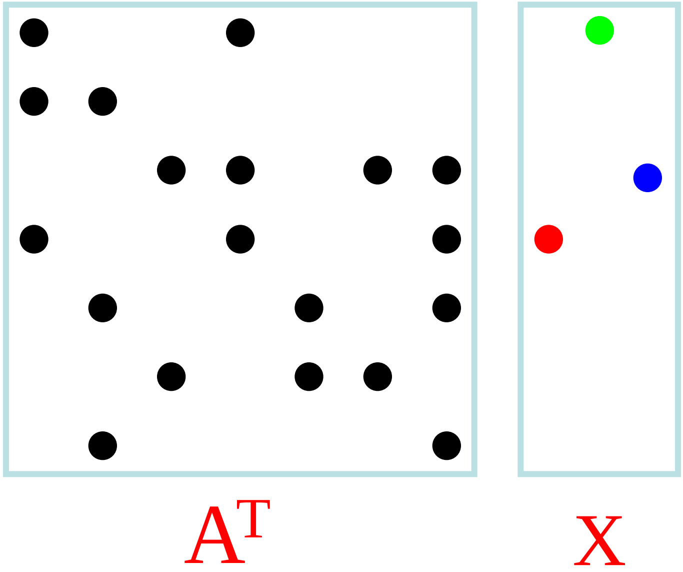


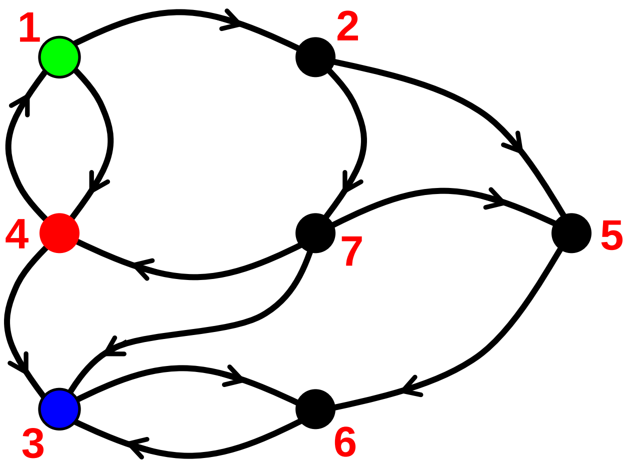
Sparse matrix indexing



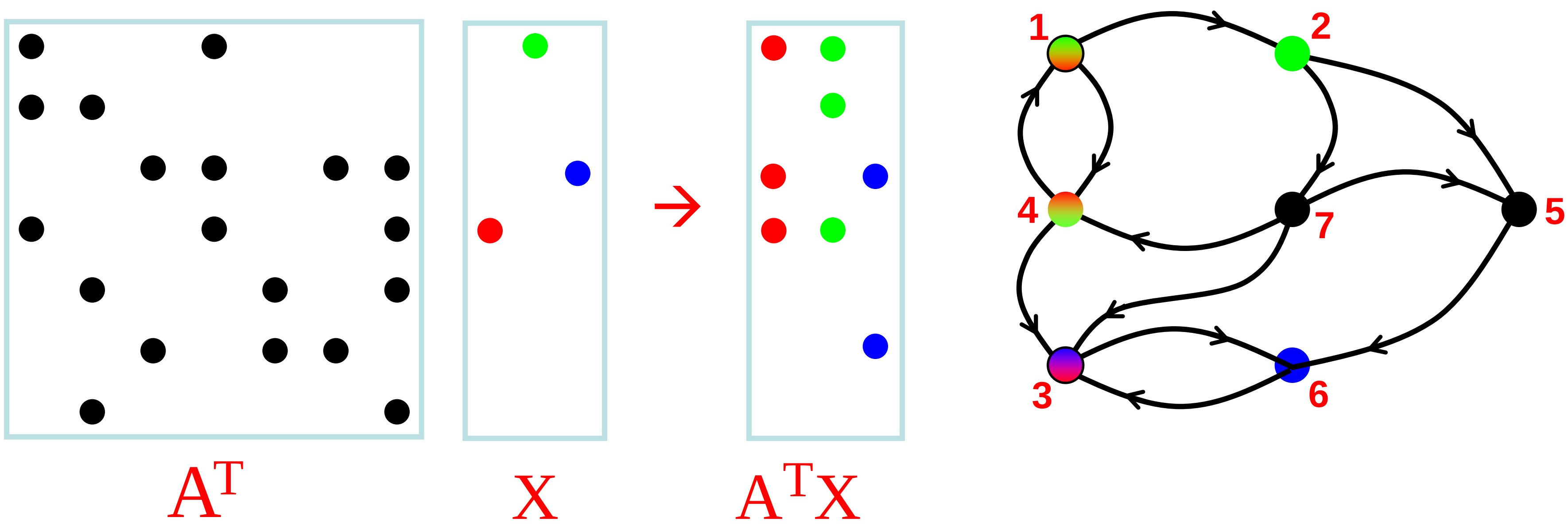
Matrices on various semirings: (x, +) , (and, or) , (+, min) , …

Multiple-source breadth-first search



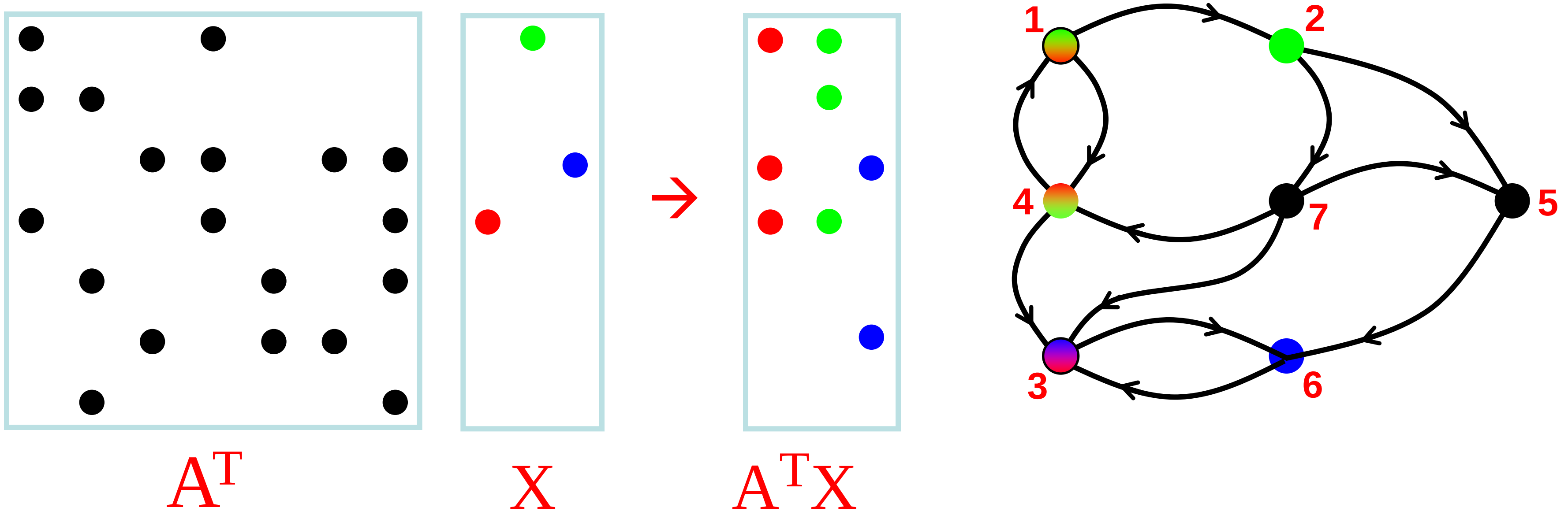


Multiple-source breadth-first search



UCSB

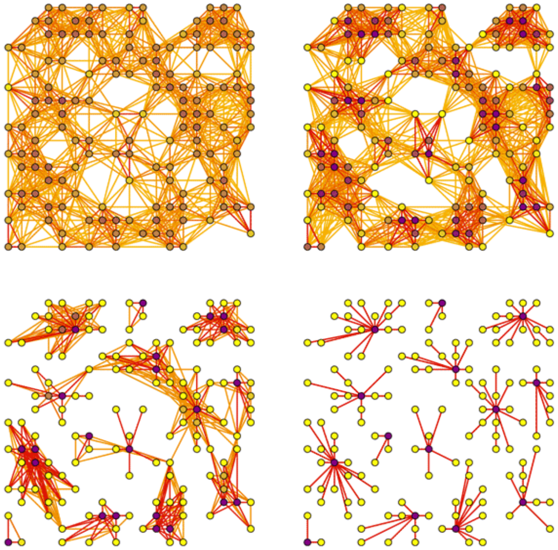
Multiple-source breadth-first search

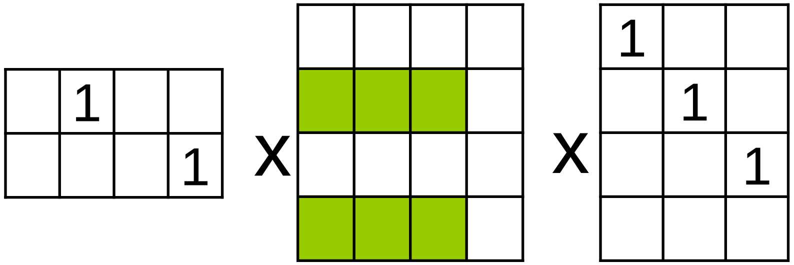


- Sparse array representation => space efficient
- Sparse matrix-matrix multiplication => work efficient
- Load balance depends on SpGEMM implementation

SpGEMM: Sparse Matrix x Sparse Matrix

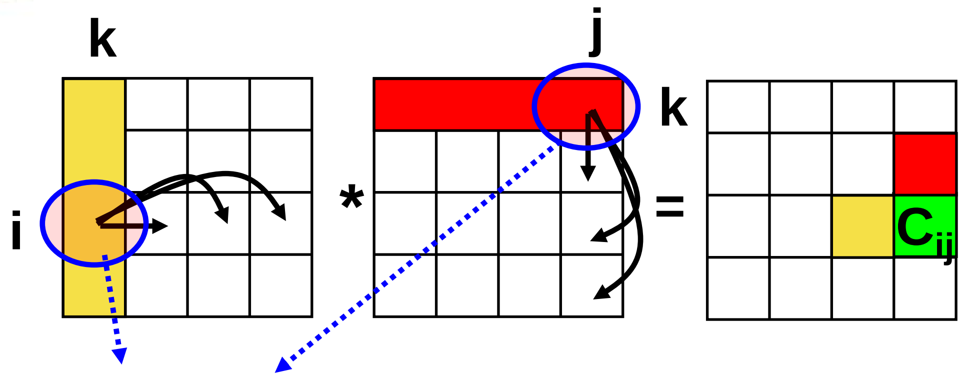
- Graph clustering (Markov, peer pressure)
- Subgraph / submatrix indexing
- Shortest path calculations
- Betweenness centrality
- Graph contraction
- Cycle detection
- Multigrid interpolation & restriction
- Colored intersection searching
- Applying constraints in finite element computations
- Context-free parsing ...



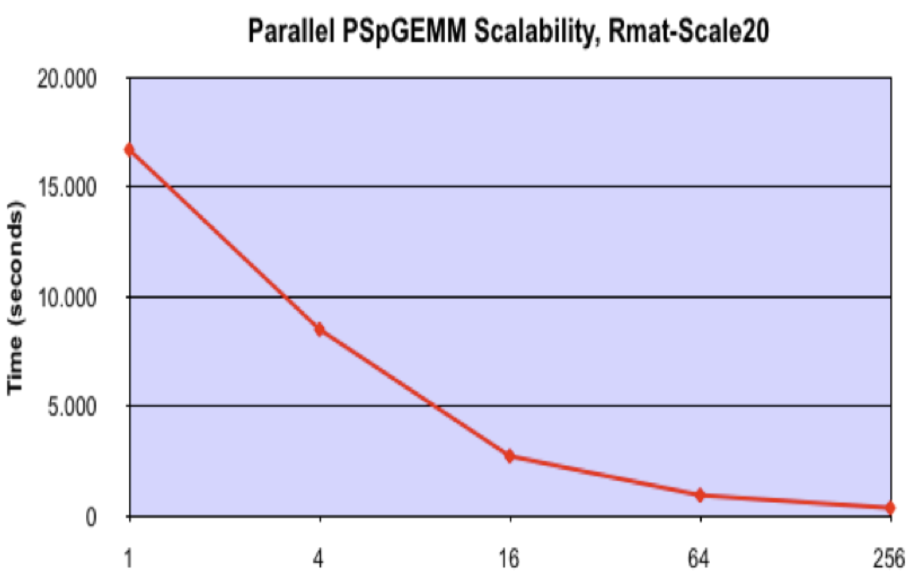


Distributed-memory sparse matrix-matrix multiplication

- 2D block layout
- Outer product formulation
- Sequential “hypersparse” kernel



C_{ij} += A_{ik} * B_{kj}



- Scales well to hundreds of processors
- Betweenness centrality benchmark: over 200 MTEPS
- Experiments: TACC Lonestar cluster

Time vs Number of cores -- 1M-vertex RMAT

UCSB

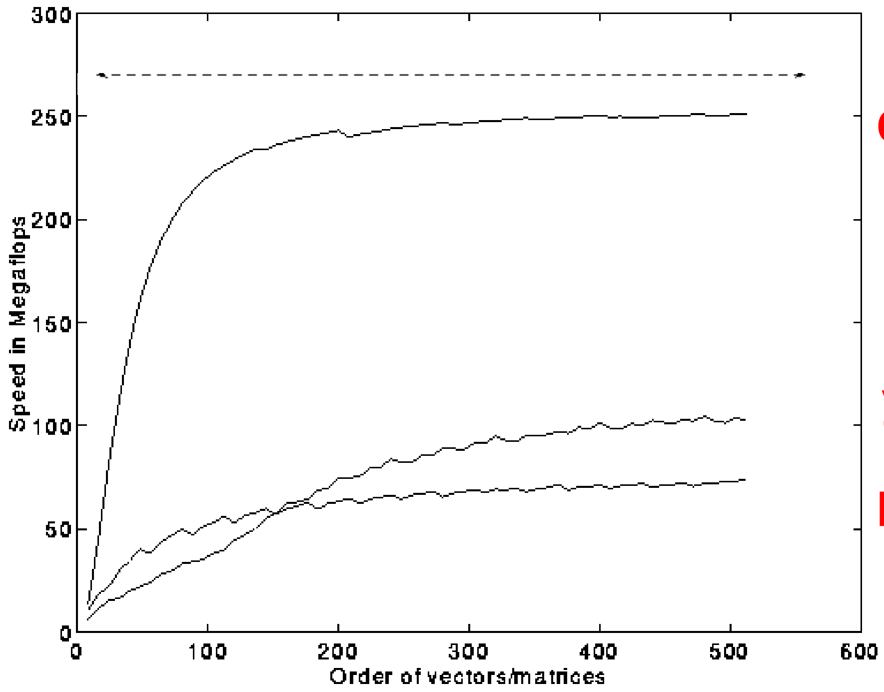
A Parallel Library: Combinatorial BLAS

The Primitives Challenge

- By analogy to numerical scientific computing. . .

- What should the combinatorial BLAS look like?

Basic Linear Algebra Subroutines (BLAS): Speed (MFlops) vs. Matrix Size (n)

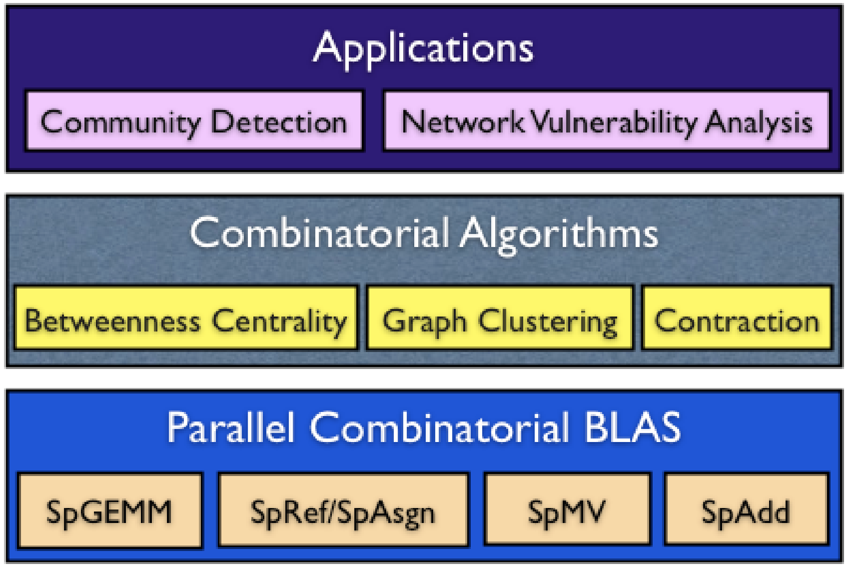


$C = A*B$

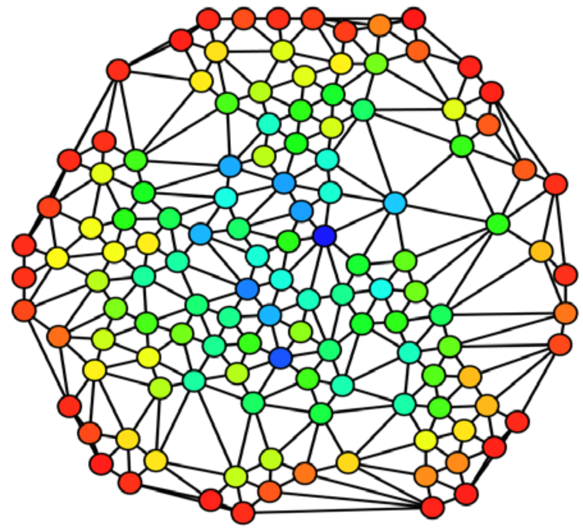
$y = A*x$

$\mu = x^T y$

The Combinatorial BLAS: Example of use



Software stack for an application of the Combinatorial BLAS



Betweenness Centrality (BC)

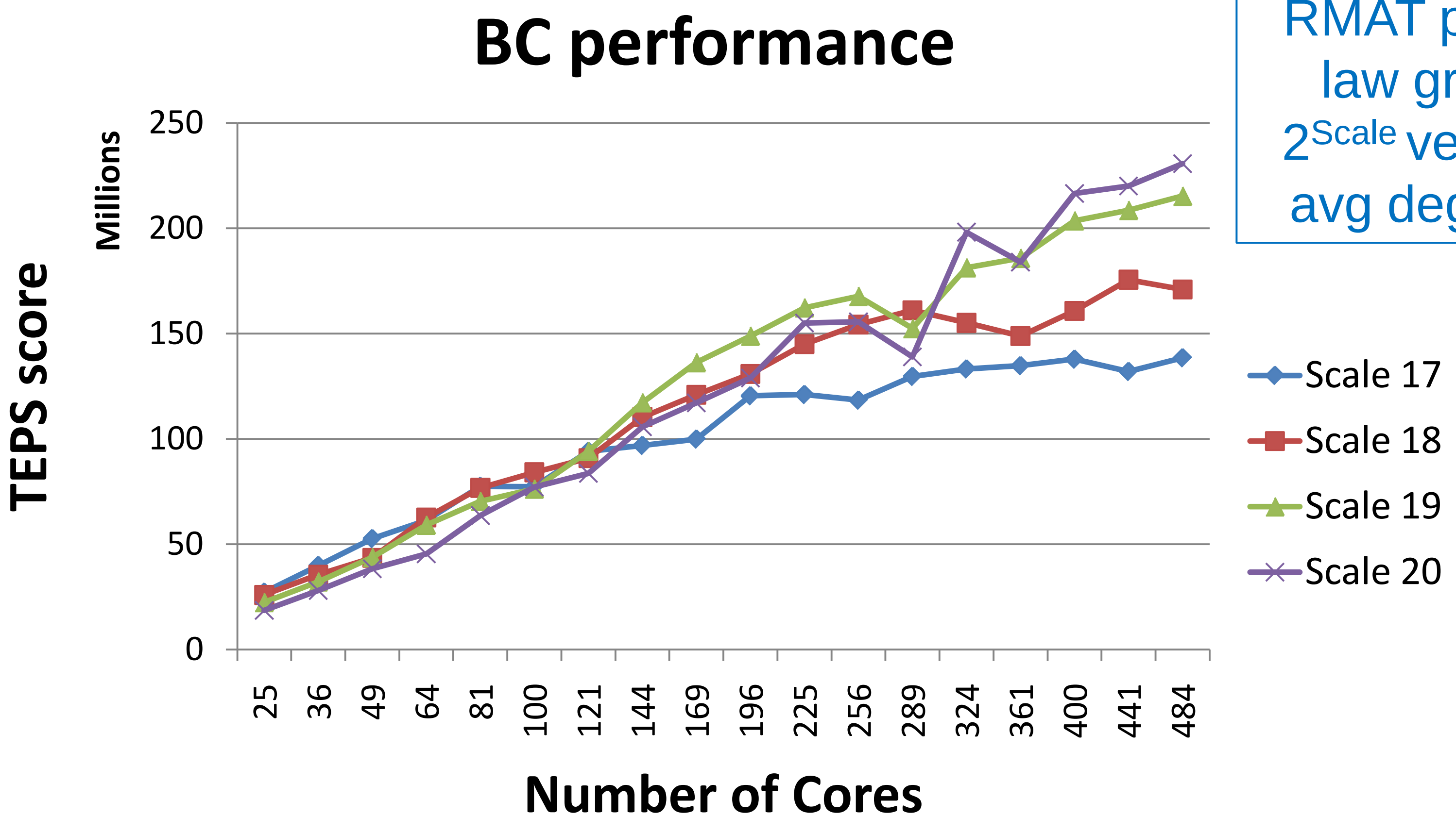
What fraction of shortest paths pass through this node?

$$C_B(v) = \sum_{\substack{s \neq v \neq t \in V \\ s \neq t}} \frac{\sigma_{st}(v)}{\sigma_{st}}$$

Brandes' algorithm

UCSB

BC performance in distributed memory



RMAT power-law graph, 2^{Scale} vertices, avg degree 8

- TEPS = Traversed Edges Per Second
- One page of code using CBLAS

The Education Challenge

- How do you teach this stuff?
- Where do you go to take courses in
  - Graph algorithms …
  - … on massive data sets …
  - … in the presence of uncertainty …
  - … analyzed on parallel computers …
  - … applied to a domain science?

Final thoughts

- Combinatorial algorithms are pervasive in scientific computing and will become more so.
- Linear algebra and combinatorics can support each other in computation as well as in theory.
- A big opportunity exists for computer architecture to influence combinatorial algorithms.
- This is a great time to be doing research in combinatorial scientific computing!

UCSB