Combinatorial Algorithms: The *Real* Power Behind Parallel Computing

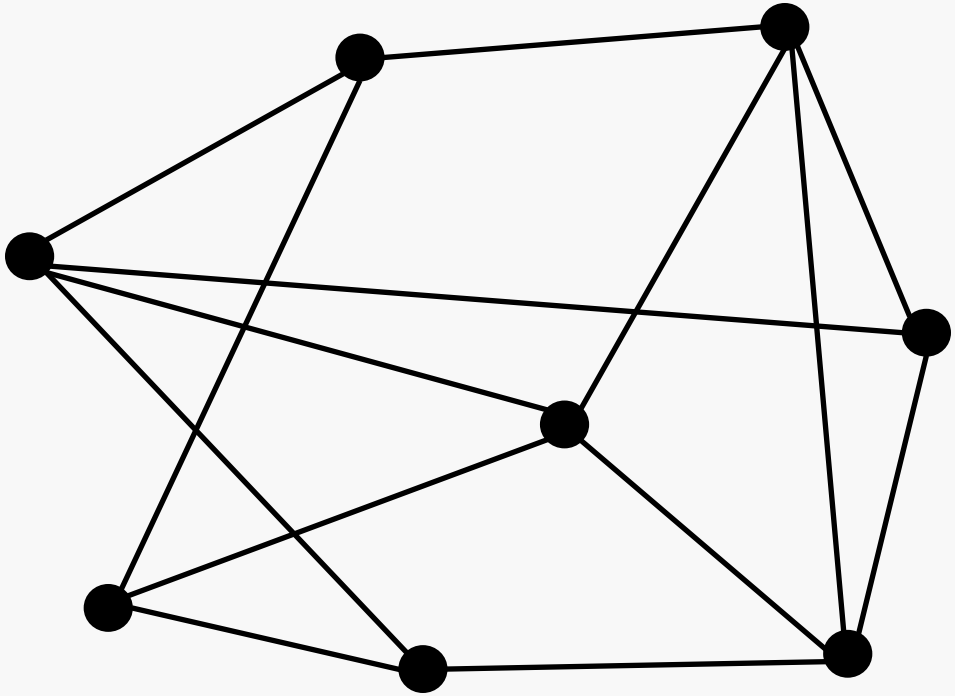
Bruce Hendrickson
Sandia National Labs &
The University of New Mexico

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the US Department of Energy under contract DE-AC04-94AL85000.

Outline

- Introduction to graphs
- The many roles of graphs in scientific computing
- Graphs as enablers of parallel computing
- Algorithm exploration: Graph coloring
- Architectures & exascale implications
- Conclusions

What's a Graph?



- Vertices and edges = entities and relationships
  - Social relations, web links, data dependencies, etc., etc.
- Many variations:
  - Directed graphs, hypergraphs, semantic graphs, etc.

Why Are Graphs Important to CSE?

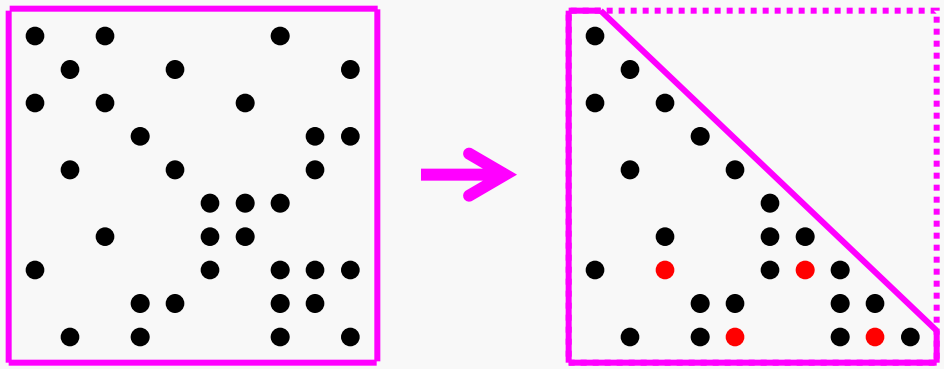
- Sparse matrix algorithms
- Preconditioning
  - Reordering for heavy diagonal, incomplete factorizations, AMG, support theory, etc.
- Automatic differentiation and optimization
  - Sparse Jacobian construction, spase bases
- Mesh generation
  - Topology exploitation, mesh refinement
- Computational biology, chemistry & physics

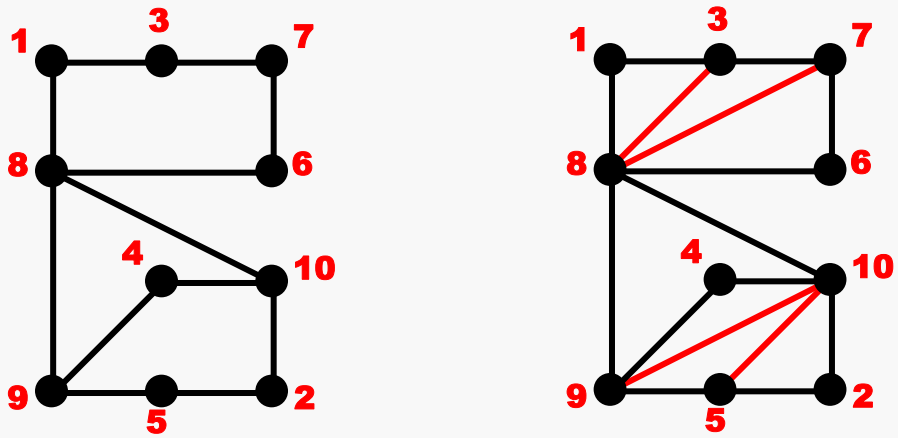
- Improved use of memory hierarchy
- Parallel algorithms

Sparse Matrix Algorithms

- Reorderings for sparse factorizations
  - Powerfully phrased as graph problems
  - Fill reducing orderings
    - —Minimum degree (greedy)
    - —Nested dissection (divide & conquer)
  - Bandwidth reducing orderings
    - —graph traversals, graph eigenvectors
  - Heavy diagonal to reduce pivoting (matching)
- Efficient exploitation of sparsity
  - Factorization, triangular solves, etc.

Graphs & Sparse Gaussian Elimination





G(A)

$G^+(A)$
[chordal]

Fill: new nonzeros in factor

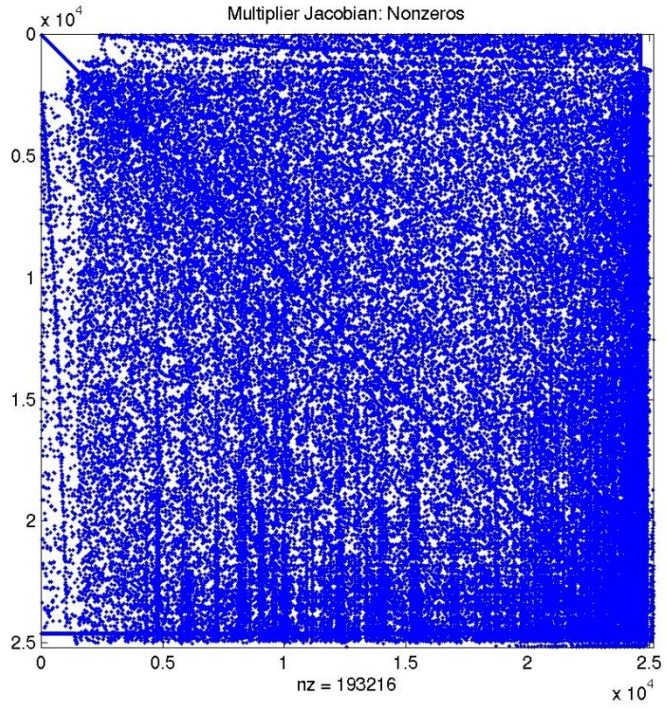
Cholesky factorization:

for j = 1 to n

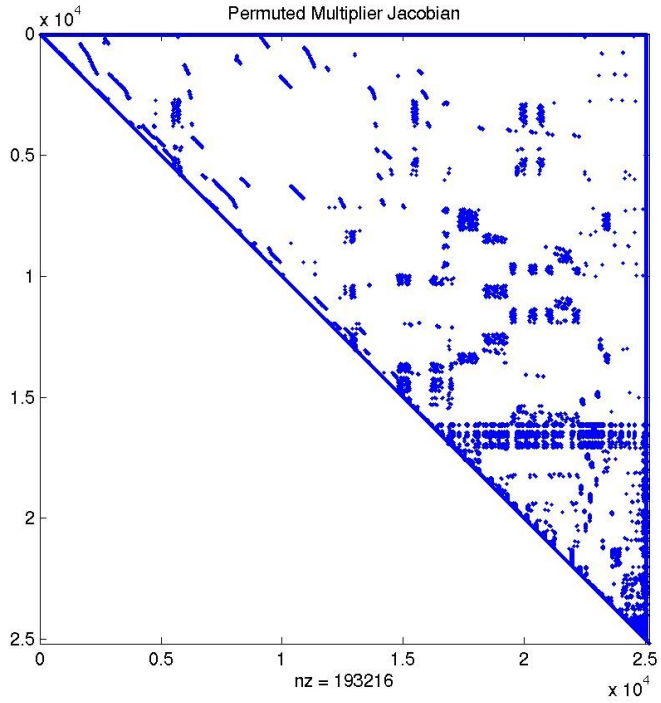
add edges between j's higher-numbered neighbors

Sandia National Laboratories

Matrix Reordering: Strongly Connected Components



Before

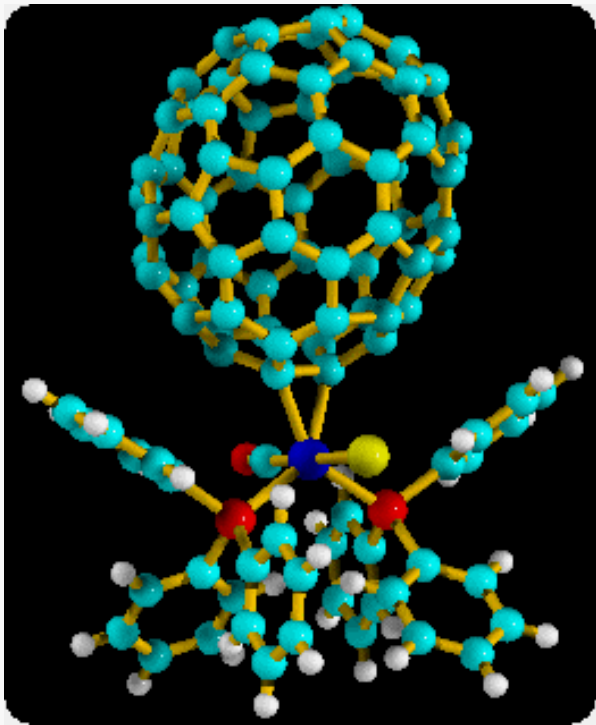


After

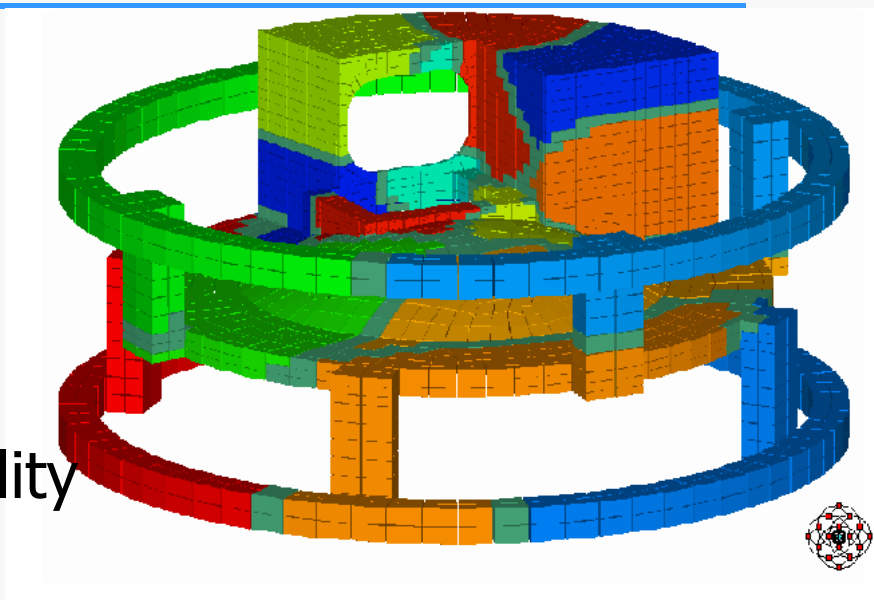
Sandia National Laboratories

Graphs & Chemistry

- Categorizing molecules by graph properties
  - Various topological invariants, graph properties
  - Used to screen molecules for desired properties
- Combinatorics of polymers
  - Geometric and graph properties
- Statistically correct ensembles
  - Graph enumeration and sampling



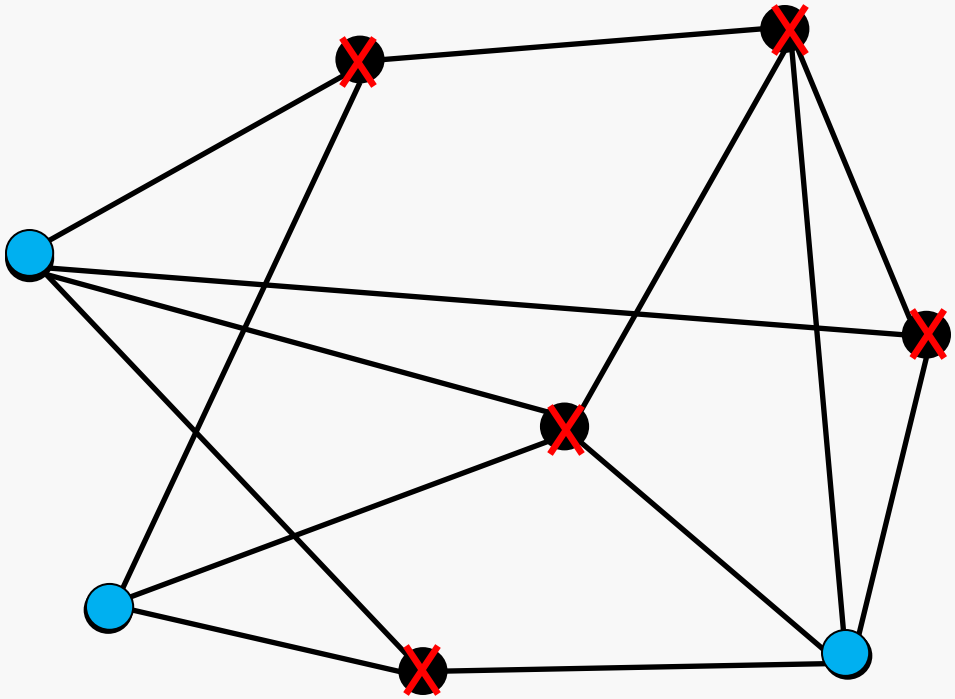
Graphs and Parallel Computing



- Graph Algorithms
  - Partitioning
  - Coloring
  - Independent sets, etc.
- Reordering for memory locality

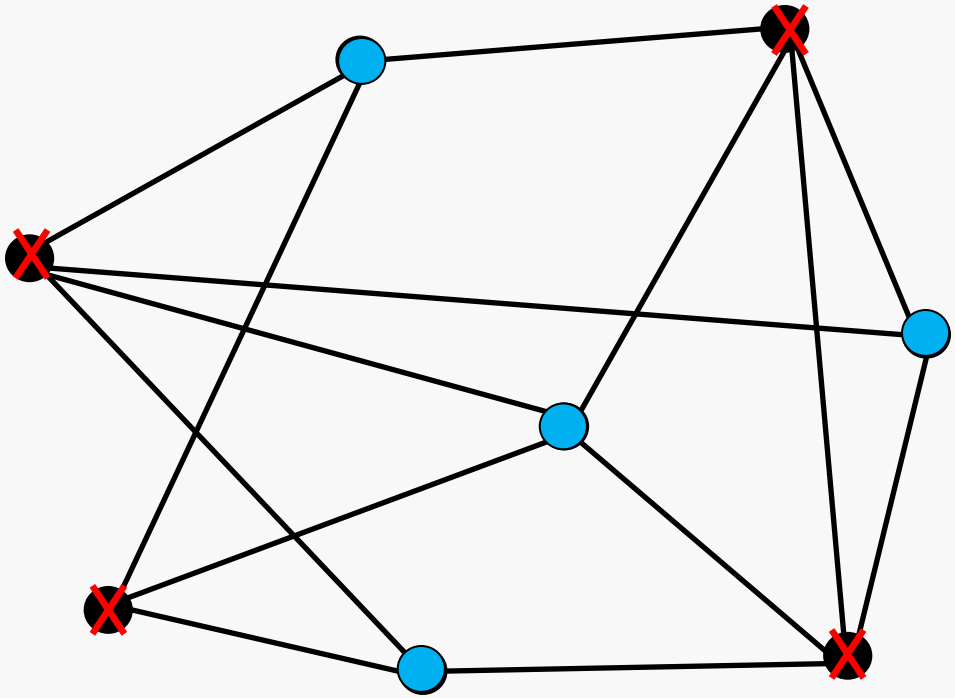
- Observation: Parallelization is usually orthogonal to numerics
- Driving parallelization challenges are non-numerical
  - Load balancing
  - Communication minimization
  - Scheduling, etc.

Example: Independent Set



- Find large subset of vertices without edges between any of them
  - Maximizing is NP-complete in general
- Useful for identifying a bunch of independent tasks
  - Mesh refinement, algebraic multigrid, etc.

Example: Independent Set

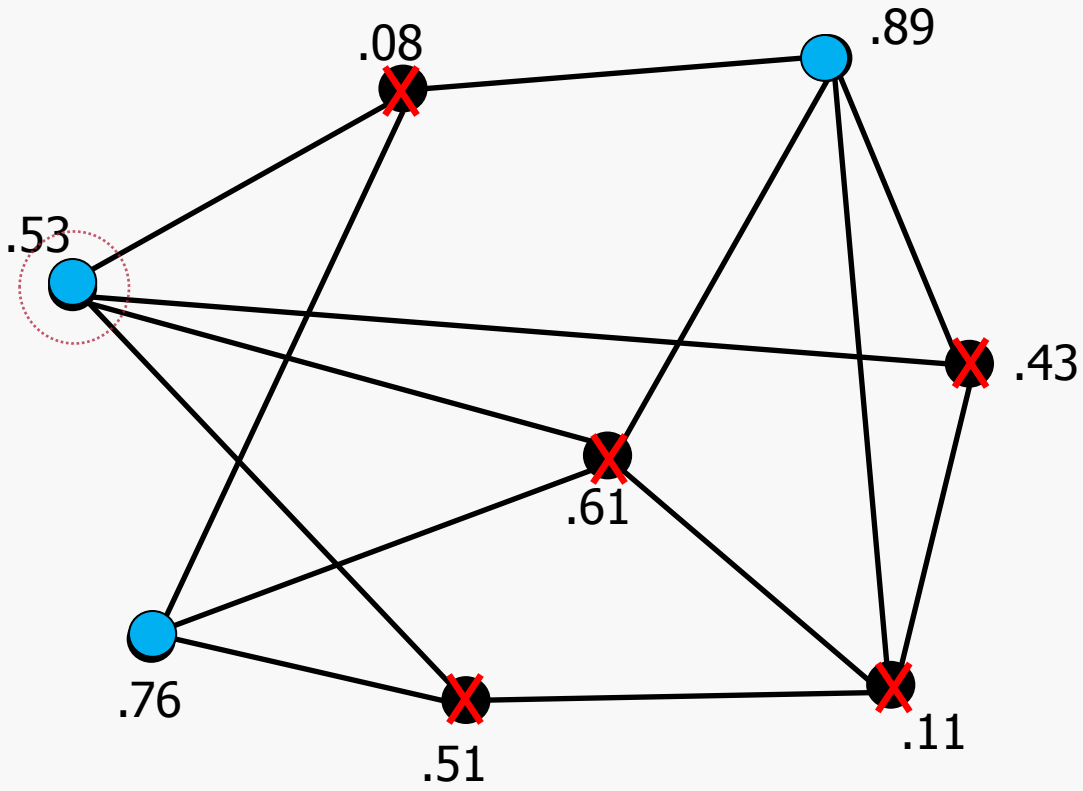


- Find large subset of vertices without joining edges
  - Maximizing is NP-complete in general
- Useful for identifying a bunch of independent tasks
  - Mesh refinement, algebraic multigrid, etc.

Independent Set Algorithms

- Natural algorithm is highly serial
  - Pick a vertex, exclude its neighbors, pick another
- Luby's algorithm
  - Assign a random number to each vertex
  - Exchange random numbers with neighbors
  - If my number is larger than those of all my neighbors, I'm selected
  - Repeat on what's left of the graph
    - — Vertices neither selected nor excluded
- Highly parallel, but more work than serial algorithm

Luby's Algorithm



- End of phase, not all vertices selected or excluded
- Need multiple phases to finish the task

Sandia National Laboratories

Coding Luby's Algorithm in MPI

- Each processor owns a subset of vertices

Comm.
- Send my vertex values to procs owning neighbors
- Receive values from neighboring processors

- Check which of my vertices dominate neighbors

Comm.
- Send message to procs owning vertices I dominate
- Receive dominance messages

- Exclude selected & dominated vertices
- Repeat on remaining graph until done

Communication forces bulk synchronization

Not much work but lots of comm/sync

Bulk-Synchronous Parallel Programming (BSP)

- Abstract model developed by Valiant
- Compute locally; communicate; repeat
- Basic approach to almost all successful MPI programs
  - Good fit for many scientific applications
- Works best when lots of load-balanced local computation, and only small amount of communication
- Often not true for graph algorithms

Graph Algorithms and MPI

- Minimal computation to hide communication time
- Runtime is dominated by latency
  - Complex memory access patterns
  - Parallelism can be fine grained and dynamic
- Access pattern is data dependent
  - Prefetching unlikely to help
  - Usually only want small part of cache line
- Potentially abysmal locality at all levels of memory hierarchy
- Many algorithms are not bulk synchronous

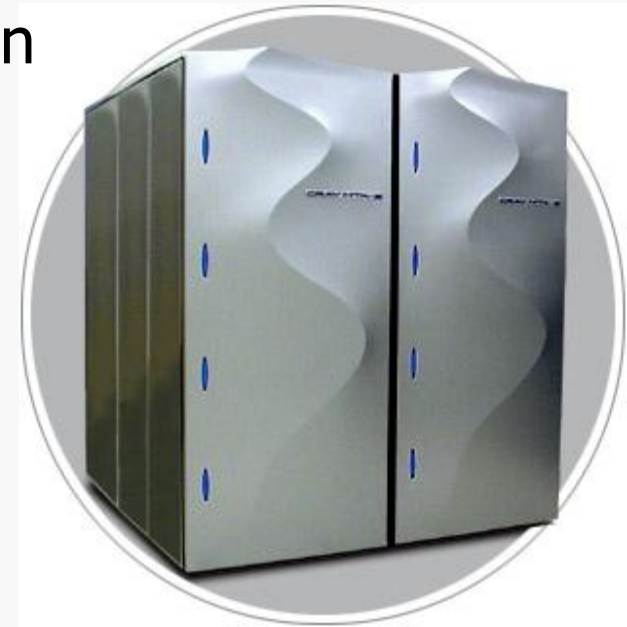
Properties of a Good Graph Machine

- Low latency / high bandwidth
  - For small messages!
- Latency tolerant
- Light-weight synchronization mechanisms for fine-grained parallelism
- Global address space
  - No graph partitioning required
  - Avoid memory-consuming profusion of ghost-nodes
  - No local/global numbering conversions

- One machine with these properties is the Cray XMT

How does the XMT Work?

- Latency tolerance via massive multi-threading
  - Context switch in a single tick
  - Global address space, hashed to reduce hot-spots
  - No cache or local memory.
  - Multiple outstanding loads
- Remote memory request doesn't stall processor
  - Other streams work while your request gets fulfilled
- Light-weight, word-level synchronization
  - Minimizes conflicts, enables parallelism
- Flexible dynamic load balancing
- Note:
  - 500 MHz processors
  - Predecessor the MTA had 220 MHz clock



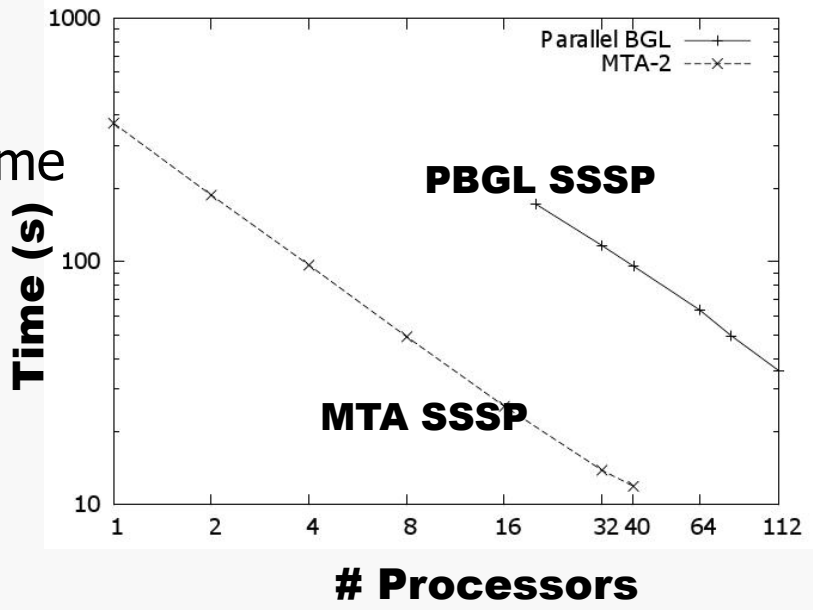
MPI versus Multithreading

- Single Source Shortest Path (Delta Stepping)

- Parallel Boost Graph Library (PBGL)
  - Lumsdaine, et al., on Opteron cluster
  - Some graph algorithms can scale on some inputs

- PBGL - MTA Comparison on SSSP
  - Erdös-Renyi random graph ($|V|=2^{28}$)
  - PBGL SSSP can scale on non-power law graphs
  - Order of magnitude speed difference
  - 2 orders of magnitude efficiency difference
    - Big difference in power consumption

[Lumsdaine, Gregor, H., Berry, 2007]

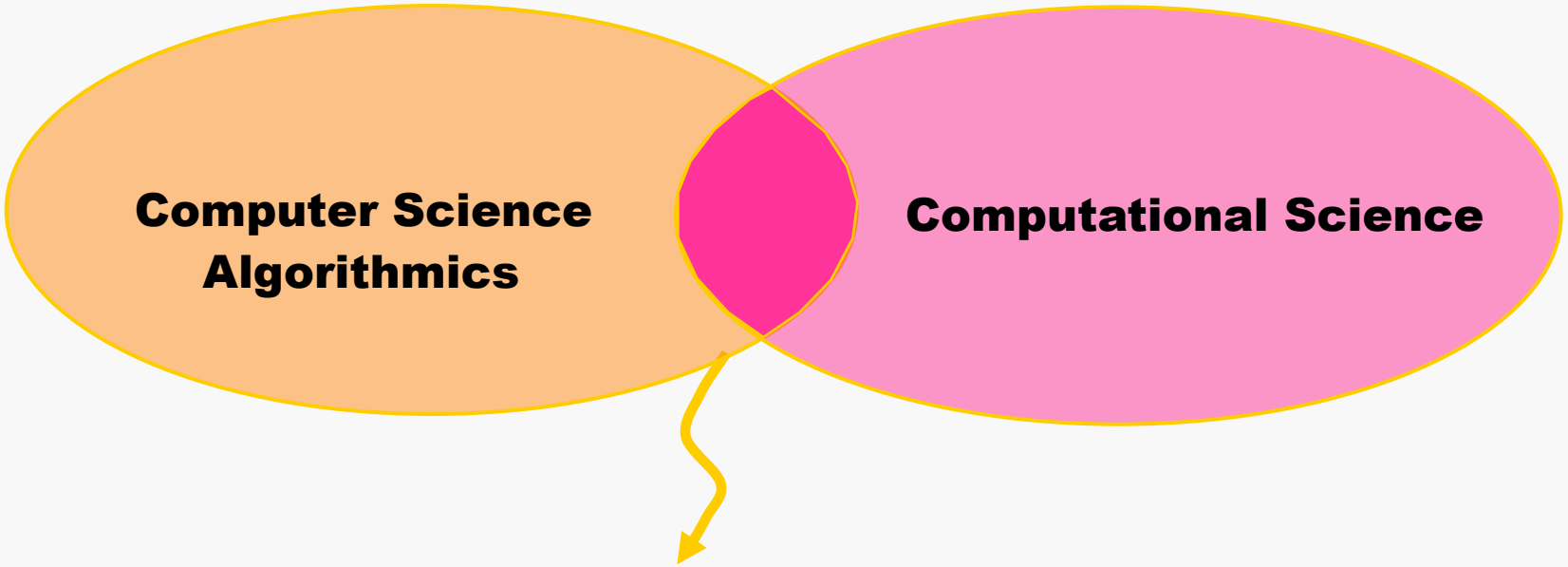


Sandia National Laboratories

Implications for Exascale Computing

- Exascale machines will require algorithmic changes
  - Avoid global communication & synchronization
  - Work more asynchronously
  - Design algorithms with more fault tolerance
- State of the art scientific applications are changing
  - Dynamic & unstructured data structures
  - Multiphysics, multilevel, multiscale
  - Becoming less bulk synchronous
- Looking more and more like graph algorithms!
  - Graph algorithms can serve as a canary in the coal mine for future architectures and programming models

Computational Science and Computer Science



- What's in this intersection?
  - **Combinatorial Scientific Computing**
  - *The development, application and analysis of combinatorial algorithms to enable scientific and engineering computations*

Sandia National Laboratories

Worlds Apart

- Computer Science =
  - Graph algorithms, set theory, complexity theory, etc.
- Computational Science & Engineering =
  - Numerical analysis, PDEs, linear algebra, etc.
- Differ in many ways
  - Vocabulary, concepts and abstractions
  - Culture – mathematics versus engineering
  - Definition of success
  - Aesthetics
- Not an easy divide to span!

Conclusions

- Graph algorithms are key enablers for scientific & parallel computing
- Graph algorithms have different characteristics than many traditional numerical algorithms
  - Latency dominated, not bulk synchronous
- Many future scientific applications will be similarly latency bound
  - Exascale trends exacerbate this challenge
- Research on graph algorithms can provide needed insights
  - And it's fun too!