What's New in Isorropia?

Erik Boman

Cedric Chevalier, Lee Ann Riesen

Sandia National Laboratories, NM, USA

Trilinos User's Group, Nov 4-5, 2009.

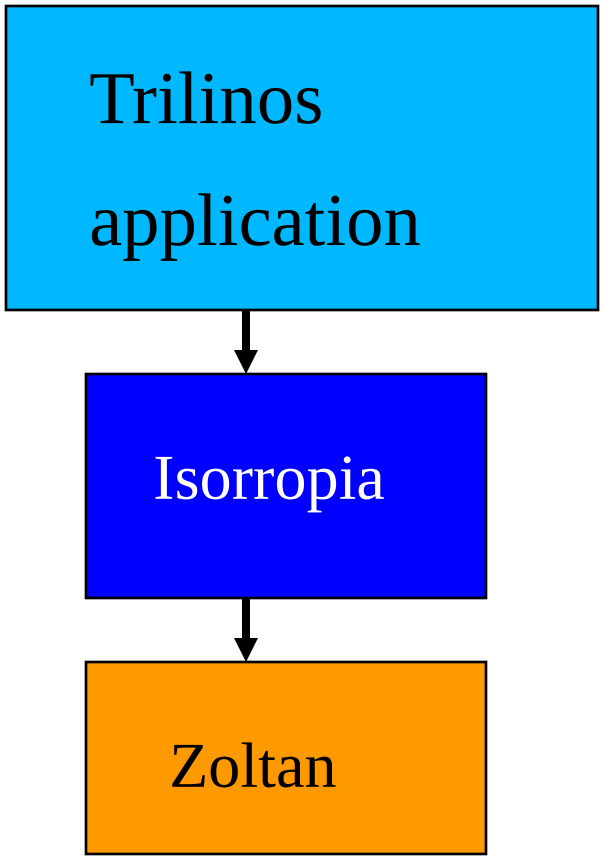
SAND 2009-7611P.

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Isorropia Overview

Isorropia

- is a package for combinatorial scientific computing:
  - Partitioning, load-balancing
  - Matrix/graph coloring
  - Matrix/graph ordering
- provides Epetra-based interface to Zoltan.



Comparison Chart

	Zoltan	Isorropia
Build system	CMake and Automake	CMake
Language	C (also C++ and F90 interfaces)	C++
Interface	Callback functions (user must provide)	Epetra data types
Package dependencies	None	Zoltan, Epetra, Teuchos
Features	Partitioning, Coloring, Ordering, Dist. data directory, Unstr. Comm. Lib.	Partitioning, Coloring, Ordering, Data redistribution

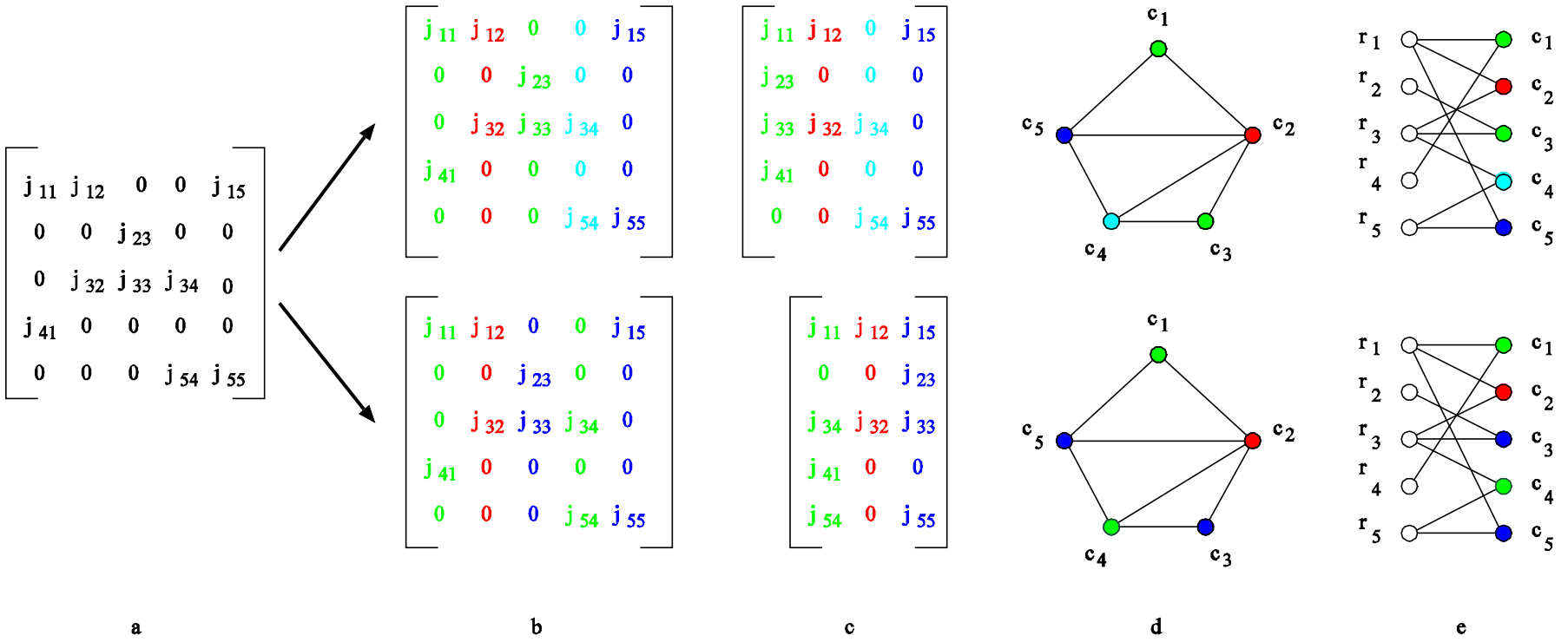
What's New?

- Parameters
  - Expanded set of Isorropia parameters
  - Zoltan parameters are now optional (expert users)
- Automatic symmetrization
  - A+A' is formed when algorithm requires sym. graph
- Partitioning
  - Geometric partitioning of points
    - Epetra_Multivector interface
    - Algorithms: RCB, RIB, HSFC (in Zoltan)
- Coloring
  - Support for Jacobian coloring

Coloring

- Isorropia supports graph/matrix coloring via the Colorer class
- Several variations of coloring (d1, d2)
- Scalable, parallel algorithm
  - Bozdag, Gebremedhin, Catalyurek, Manne, Boman, JPDC 2008.
- Default in Isorropia is to color matrix columns
  - Intended for sparse Jacobians

Coloring and Jacobians



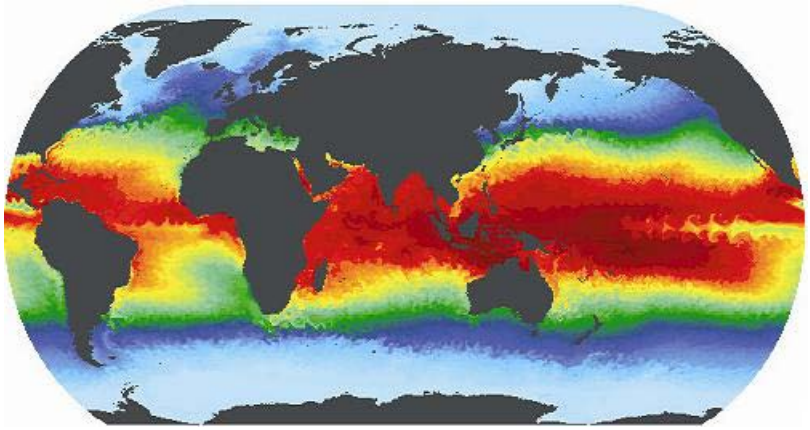
Original Jacobian

Compressed **representation**
Structurally orthogonal

D1 coloring
formulation on
column inter. graph

D2 coloring

POP



- POP is a parallel ocean simulator for climate
- SNL is working with LANL to use Trilinos in POP
- Solver uses JFNK
  - No explicit Jacobian
  - Forming Jacobian by finite differences is expensive

Coloring in POP (1)

Work led by Chris Siefert.

- Want to precondition Jacobian
- Need to explicitly form preconditioner
- Use coloring on graph of ~Jacobian
  - Approximation may be sufficient
- Form compressed ~Jacobian by finite diff.
- Uncompress ~Jacobian and precondition

Coloring in POP (2)

Example:

- 40x42x34 mesh -> 290K unknowns
- Isorropia parallel coloring gives 432 colors
  - # finite differences reduced from 290K (naïve) to 432!
- Jacobian build takes 30-50% of total time
  - Jac_build: 1377 s
  - Solve: 2618 s
  - Total: 4281 s

- Any reduction in #colors reduces total time
  - Work in progress: Use bipartite graph in Isorropia to reduce colors.

Ordering

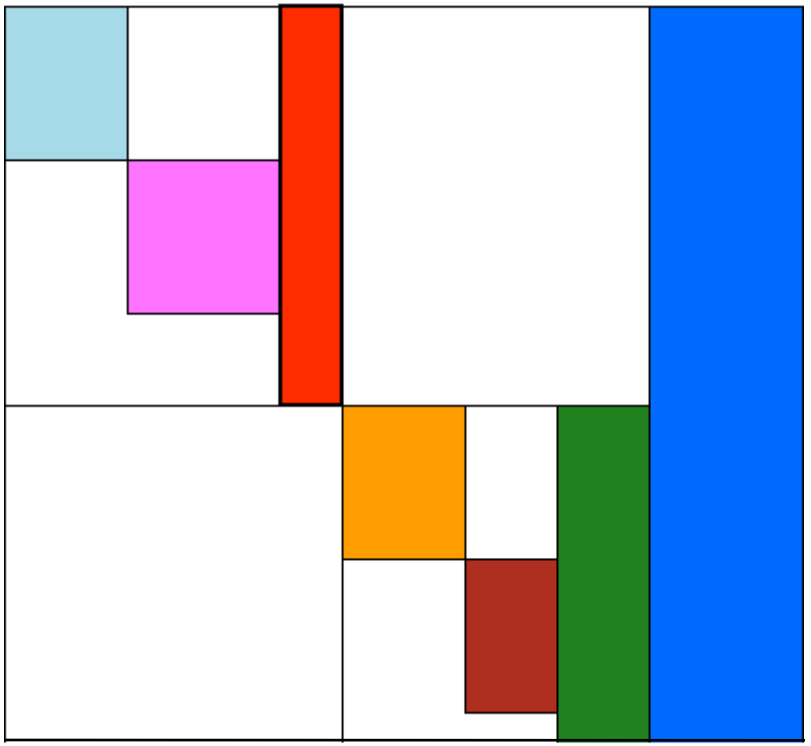
- Ordering for sparse matrices can help:
  - Reduce fill in direct factorization (Amesos)
  - Improve convergence in iterative methods (IFPACK)
  - Improve memory/cache performance in sparse kernels (Epetra, Tpetra)
- So far focus on global (parallel) ordering for fill
  - Rely on TPL: ParMetis or Scotch
  - In progress: Native Zoltan ordering
    - HUND for unsymmetric problems

Sparse LU

- A = LU → Solve Ly=b, Ux=y
- Permute to keep L, U sparse
  - Fill-reducing ordering
- Need (partial) pivoting for numerical stability:
  - PA = LU
    - P is a row permutation from pivoting
- Can reorder columns to reduce fill
  - PAQ = LU
  - We choose Q but P is not known a priori

Hypergraph Unsymmetric Nested Dissection (HUND)

- Permute columns
  - Also permute rows but allow row pivoting
- Use hypergraph SBBD ordering recursively
  - Grigori, Boman, Donfack, Davis ('08)
  - Analogous to nested dissection for symmetric problems
  - Fill is limited to nonzero blocks for any pivoting
  - Useful both in serial and in parallel



HUND in Zoltan

- Design for handling matrices for parallel solvers
  - Minimum Degree heuristics do not provide enough parallelism (and cannot really be parallelized)
  - Block form is computed with Zoltan's parallel hypergraph partitioner
- To improve quality inside the blocks, local heuristics may be applied (COLAMD, etc.)
  - Work in progress

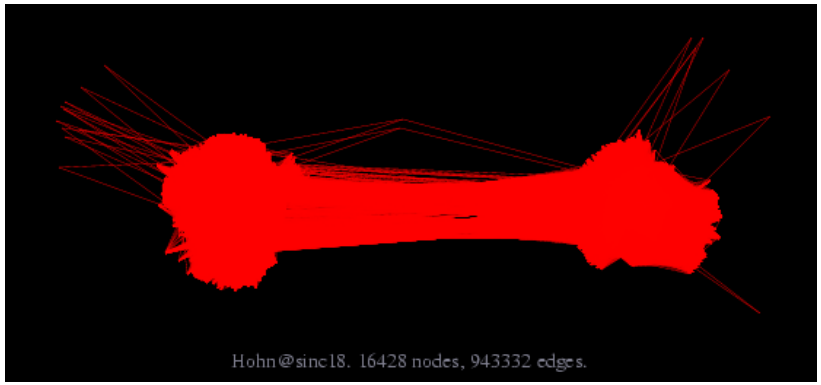
Preliminary Results

- Using HUND with only the computation of the structure:
  - worst case but give a upper bound of the factorization cost
- Evaluation of the quality using SuperLU dist on 64 processors on Franklin XT4 at Nersc.
- Comparisons against current aproachs ($A+A^t$)
  - Nested Dissection codes: ParMetis and Scotch
  - Minimum Degree
- Test Cases from Florida Collection:
  - Sinc18: crack simulation
  - ASIC_680ks: circuit simulation (from Xyce)

Sinc18

	HUND	ParMetis	MMD
L+U	53.1e+6	38.7e+6	30.3e+6
Factorization flops	84.3e+9	225e+9	51.6e+9
Factorization Time (s)	65	30.82	1.82

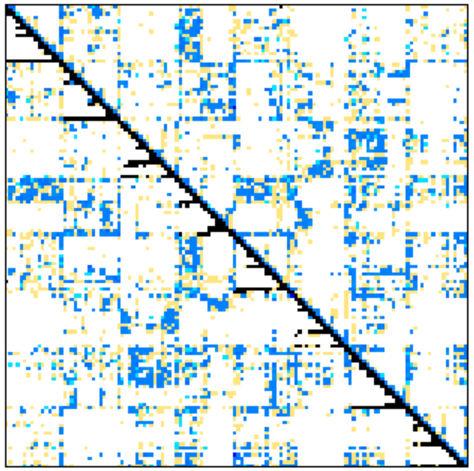
- No ordering inside the blocks can explain the timings for HUND
- Matrix structure seems appropriate for dissection approach on the highest levels



Xyce: ASIC_680ks

	HUND	Scotch	ParMetis	MMD
L+U	37.2e+6	83.8e+6	12.3e+6	3.6e+6
Factorization flops	34.6e+9	362e+9	3.04e+9	1.5e+9
Factorization Time (s)	34.49	88.19	36.90	25.82

- MMD does not provide enough parallelism
- Here, HUND is the fastest ordering to compute



Ordering Plans

- Davis' SuiteSparse as TPL in Zoltan
  - Access to AMD, COLAMD, etc.
  - Use in HUND
- How to use orderings in Amesos?
  - A) Isorropia computes permutation, Amesos passes vector to solver (if supported by TPL)
  - B) Isorropia computes permutation, Amesos permutes matrix (copy?) before calling solver
- Local (serial) orderings in Zoltan
  - RCM and space-filling curves
  - Michael Wolf (for climate project)

The End