Combinatorial Algorithms in Scientific Computing
MS110

Robert Kirby (Texas Tech)

Michael Wolf (UIUC, Sandia)

2008 SIAM Annual Meeting

7/11/2008

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.

Combinatorial Scientific Computing (CSC)

- Long played important role in enabling scientific computing
- Traditionally researchers spread across several different communities
- Recent effort to form more cohesive CSC community
- CSC Workshops
  - San Francisco, 2/2004 (SIAM PP04)
  - Toulouse, 6/2005
  - Costa Mesa, 2/2007 (SIAM CSE07)
  - Monterey, 10/2009 (SIAM LA09)

Combinatorial Scientific Computing (CSC)

- Numerous conference minisymposia
- DOE SciDAC institute
  - Combinatorial Scientific Computing and Petascale Simulations (CSCAPES)

- For more info on CSC activities
  - http://www.cscapes.org/

Combinatorial Algorithms in Scientific Computing

- Inherently combinatorial tasks in scientific computing
  - e.g., data partitioning

- More subtle underlying discrete structures in scientific computing
  - Complements analytic structure of problem
  - e.g., combinatorial structure in discretized PDEs

Combinatorial Algorithms in Scientific Computing

- Michael Wolf
  - “Hypergraph-based combinatorial optimization of matrix-vector multiplication”
- Dmitry Karpeev
  - “Using Sieve for particle tracking, embedding meshing and field-particle interaction computations”
- Kevin Long
  - “Combinatorial dataflow analysis for differentiation of high-level PDE representations”
- Andrew Lyons
  - “Exploitation of Jacobian scarcity”

Hypergraph-Based Combinatorial Optimization of Matrix-Vector Multiplication

Michael M. Wolf
University of Illinois,
Sandia National Laboratories

2008 SIAM Annual Meeting
7/11/2008

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.

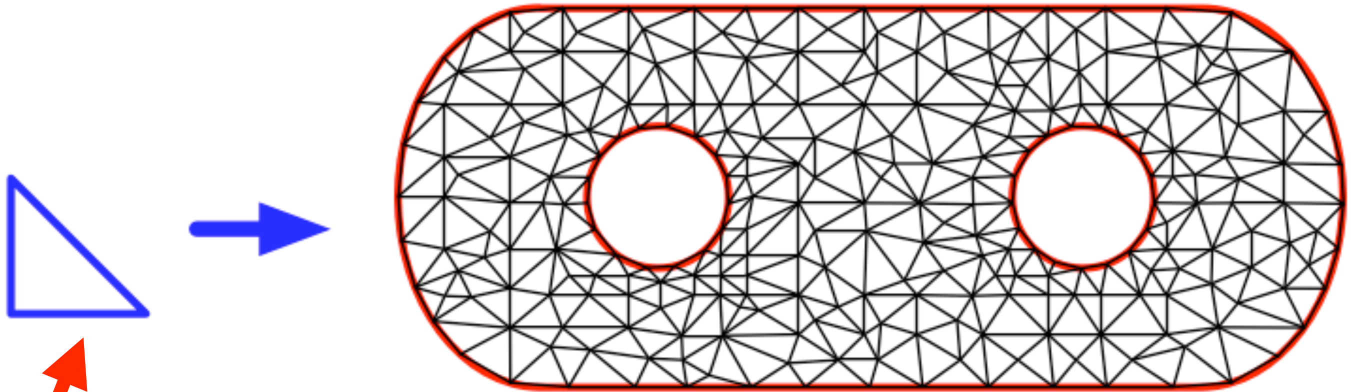
Optimization Problem

Objective: Generate set of operations for computing matrix-vector product with minimal number of multiply-add pairs (MAPs)

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

$$\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \mathbf{r_1}^T \\ \hline \mathbf{r_2}^T \\ \hline \vdots \\ \hline \mathbf{r_m}^T \end{bmatrix} \begin{bmatrix} \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{r_1}^T\mathbf{x} \\ \mathbf{r_2}^T\mathbf{x} \\ \vdots \\ \mathbf{r_m}^T\mathbf{x} \end{bmatrix}$$

Motivation



Based on reference element, generate code to optimize construction of local stiffness matrices

Can use optimized code for every element in domain

- Reducing redundant operations in building finite element (FE) stiffness matrices
  - Reuse optimized code when problem is rerun

Related Work

- Finite element “Compilers” (FEniCS project)
  - www.fenics.org
  - FIAT (automates generations of FEs)
  - FFC (variational forms -> code for evaluation)
- Following work by Kirby, et al., Texas Tech, University of Chicago on FErari
  - Optimization of FFC generated code
  - Equivalent to optimizing matrix-vector product code
  - Graph model

Matrix-Vector Multiplication

For 2D Laplace equation, we obtain following matrix-vector product to determine entries in local stiffness matrix

$$\mathbf{S}^e_{i,j} = y_{ni+j} = \mathbf{A}_{(ni+j,*)}\mathbf{x}$$

where

$$\underbrace{\mathbf{A}^T_{(ni+j,*)}}_{\text{Element independent}} = \begin{bmatrix} \left(\frac{\partial \phi_i}{\partial r}, \frac{\partial \phi_j}{\partial r}\right)_{\hat{e}} \\ \left(\frac{\partial \phi_i}{\partial r}, \frac{\partial \phi_j}{\partial s}\right)_{\hat{e}} \\ \left(\frac{\partial \phi_i}{\partial s}, \frac{\partial \phi_j}{\partial r}\right)_{\hat{e}} \\ \left(\frac{\partial \phi_i}{\partial s}, \frac{\partial \phi_j}{\partial s}\right)_{\hat{e}} \end{bmatrix} \qquad \underbrace{\mathbf{x}}_{\text{Element dependent}} = \det(\mathbf{J}) \begin{bmatrix} \frac{\partial r}{\partial x}\frac{\partial r}{\partial x} + \frac{\partial r}{\partial y}\frac{\partial r}{\partial y} \\ \frac{\partial r}{\partial x}\frac{\partial s}{\partial x} + \frac{\partial r}{\partial y}\frac{\partial s}{\partial y} \\ \frac{\partial s}{\partial x}\frac{\partial r}{\partial x} + \frac{\partial s}{\partial y}\frac{\partial r}{\partial y} \\ \frac{\partial s}{\partial x}\frac{\partial s}{\partial x} + \frac{\partial s}{\partial y}\frac{\partial s}{\partial y} \end{bmatrix}$$

Possible Optimizations - Collinear Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_2} = 1.5\mathbf{r_1}$$

Possible Optimizations - Collinear Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_2} = 1.5\mathbf{r_1} \Rightarrow y_2 = 1.5 y_1$$ 1 MAP

Possible Optimizations - Collinear Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_3} = \mathbf{r_1} \Rightarrow y_3 = y_1 \quad \boxed{\text{0 MAPs}}$$

Special case when
rows identical

Possible Optimizations - Partial Collinear Rows

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r}_4 = 2.5\mathbf{r}_1 + 8\mathbf{e}_4$$

Possible Optimizations - Partial Collinear Rows

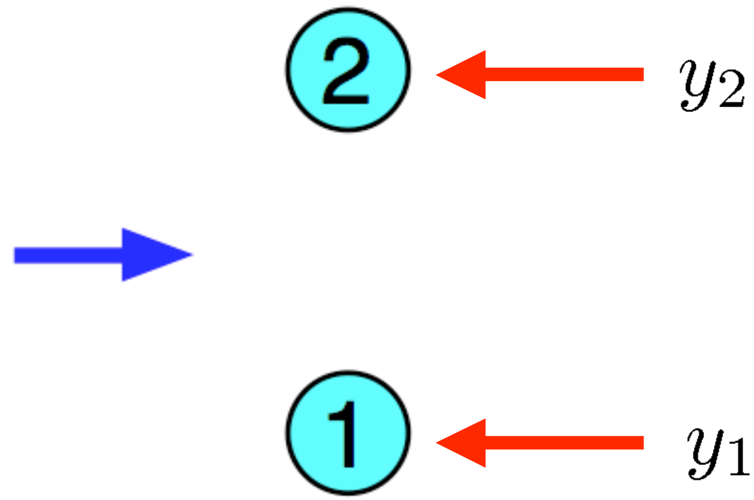
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 0 \\ 3 & 3 & 3 & 0 \\ 2 & 2 & 2 & 0 \\ 5 & 5 & 5 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_4} = 2.5\mathbf{r_1} + 8\mathbf{e_4} \Rightarrow y_4 = 2.5y_1 + 8x_4$$

2 MAPs

Graph Model - Resulting Vector Entry Vertices

$\mathbf{r_1}^T$	8	4	4	8
$\mathbf{r_2}^T$	0	-2	-2	-4

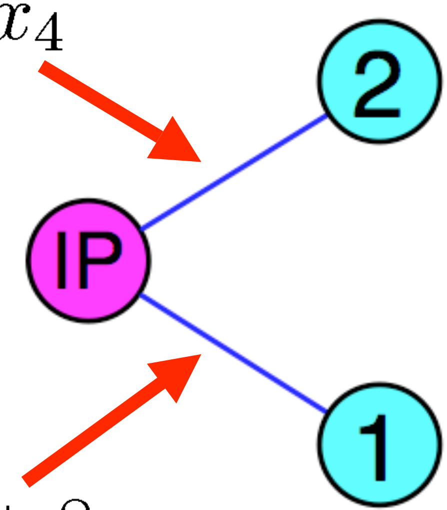


- Entries in resulting vector represented by vertices in graph model

Graph Model - Inner-Product Vertex and Edges

$$y_2 = -2x_2 - 2x_3 - 4x_4$$

$$\begin{array}{c|cccc|} \mathbf{r_1}^T & 8 & 4 & 4 & 8 \\ \mathbf{r_2}^T & 0 & -2 & -2 & -4 \end{array}$$

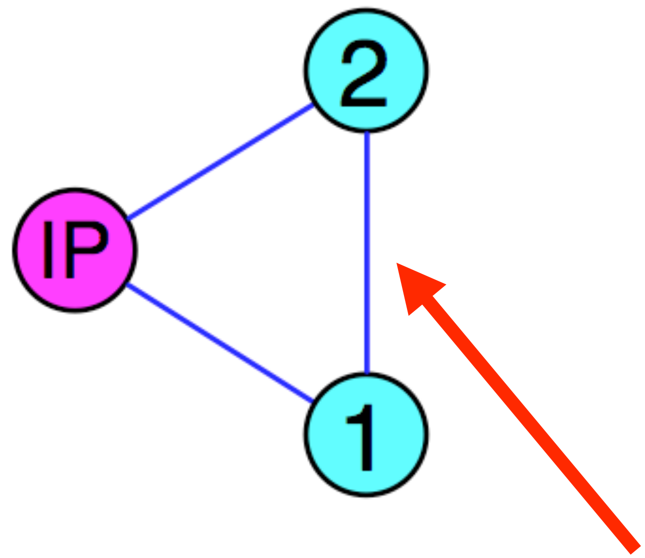


$$y_1 = 8x_1 + 4x_2 + 4x_3 + 8x_4$$

- Additional inner-product (IP) vertex
- Edges connect IP vertex to every other vertex, representing inner-product operation

Graph Model - Row Relationship Edges

$$\begin{array}{c} \mathbf{r_1}^T \\ \mathbf{r_2}^T \end{array} \begin{bmatrix} 8 & 4 & 4 & 8 \\ 0 & -2 & -2 & -4 \end{bmatrix}$$



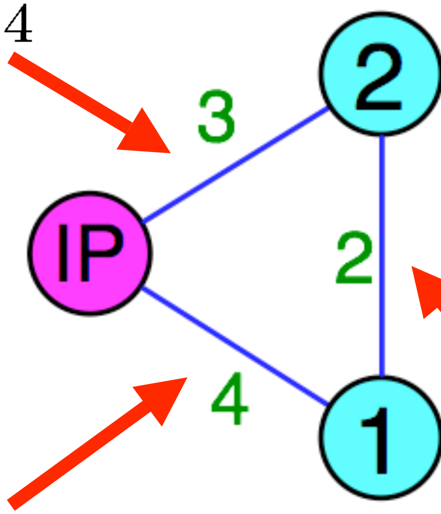
$$y_1 = -2y_2 + 8x_1$$
$$y_2 = -0.5y_1 + 4x_1$$

- Operations resulting from relationships between rows represented by edges between corresponding vertices

Graph Model - Edge Weights

$$y_2 = -2x_2 - 2x_3 - 4x_4$$

$\mathbf{r_1}^T$	8	4	4	8
$\mathbf{r_2}^T$	0	-2	-2	-4



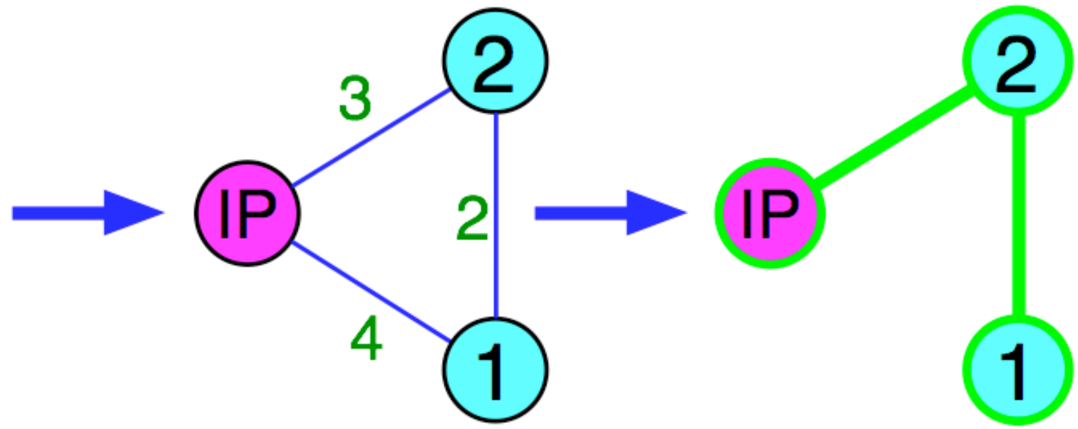
$$y_1 = 8x_1 + 4x_2 + 4x_3 + 8x_4$$

$$y_1 = -2y_2 + 8x_1$$
$$y_2 = -0.5y_1 + 4x_1$$

- Edge weights are MAP costs for operations

Graph Model Solution

$\mathbf{r_1}^T$	8	4	4	8
$\mathbf{r_2}^T$	0	-2	-2	-4



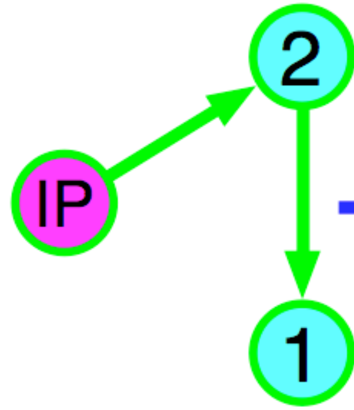
Matrix Graph MST(5)

- Solution is minimum spanning tree (MST)
  - Minimum cost subgraph
  - Connected and spans vertices
  - Acyclic

Graph Model Solution

$\mathbf{r_1}^T$	8	4	4	8
$\mathbf{r_2}^T$	0	-2	-2	-4

Matrix



MST Traversal

$$y_2 = -2x_2 - 2x_3 - 4x_4$$
$$y_1 = -2y_2 + 8x_1$$

Instructions

- Prim's algorithm to find MST (polynomial time)
- MST traversal yields operations to optimally compute (for these relationships) matrix-vector product

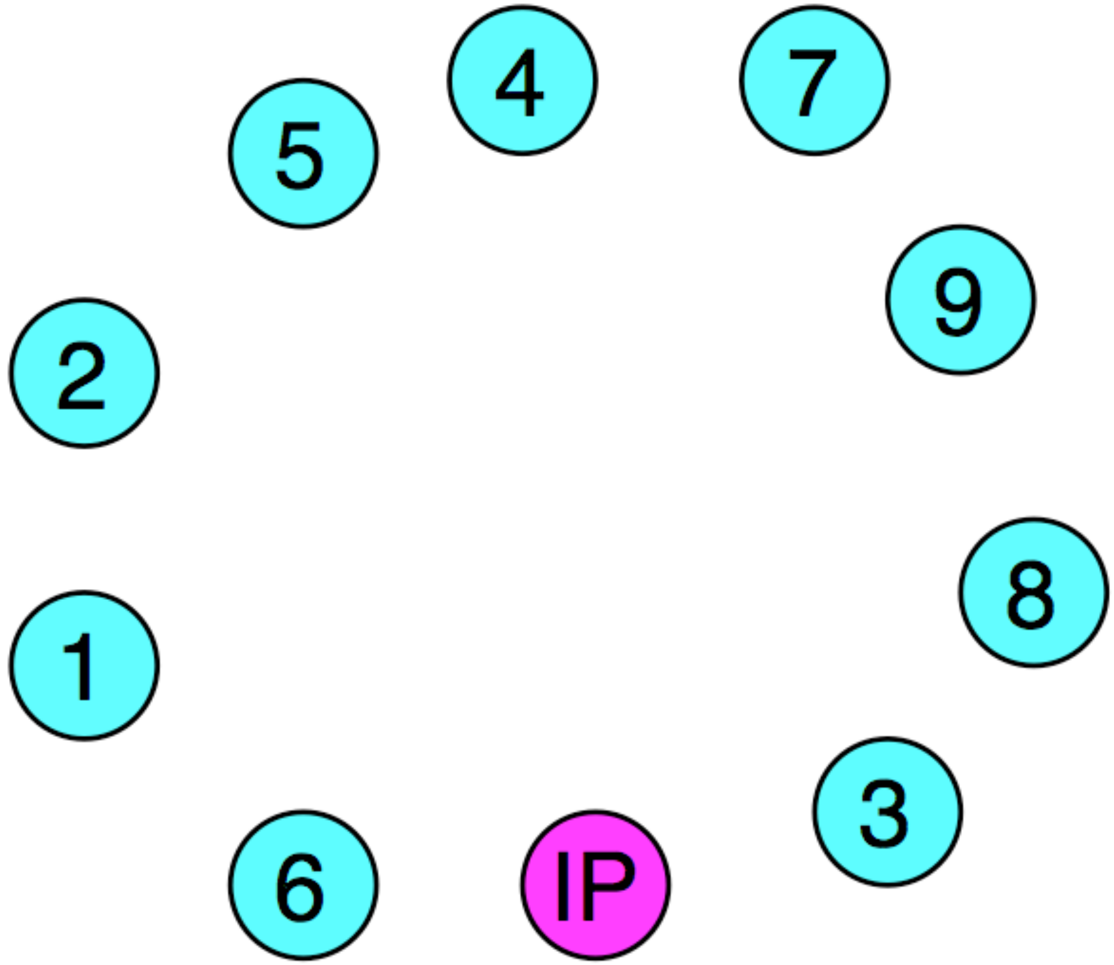
Graph Model Example

$\mathbf{r}_1{}^T$	0	4/3	0
$\mathbf{r}_2{}^T$	0	0	1/2
$\mathbf{r}_3{}^T$	1/2	0	0
$\mathbf{r}_4{}^T$	1/6	1/6	0
$\mathbf{r}_5{}^T$	0	1/6	1/6
$\mathbf{r}_6{}^T$	0	-2/3	-2/3
$\mathbf{r}_7{}^T$	-4/3	-4/3	0
$\mathbf{r}_8{}^T$	0	-4/3	-4/3
$\mathbf{r}_9{}^T$	4/3	4/3	4/3

- Matrix used for building FE local stiffness matrices
  - 2D Laplace Equation
  - 2nd order Lagrange polynomial basis
- Simplified version of matrix
  - Identical rows removed
  - Several additional rows removed

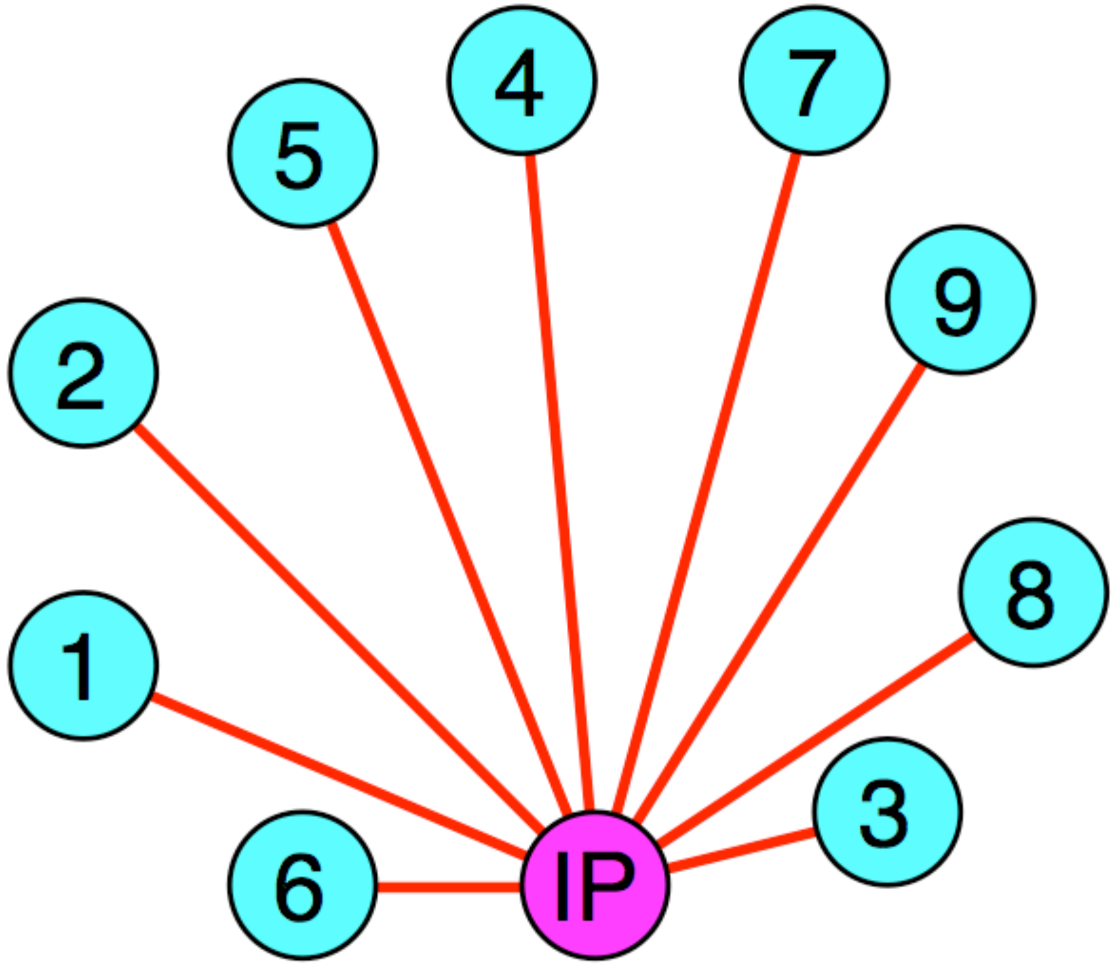
Graph Model Example - Vertices

$\mathbf{r}_1{}^T$	0	4/3	0
$\mathbf{r}_2{}^T$	0	0	1/2
$\mathbf{r}_3{}^T$	1/2	0	0
$\mathbf{r}_4{}^T$	1/6	1/6	0
$\mathbf{r}_5{}^T$	0	1/6	1/6
$\mathbf{r}_6{}^T$	0	-2/3	-2/3
$\mathbf{r}_7{}^T$	-4/3	-4/3	0
$\mathbf{r}_8{}^T$	0	-4/3	-4/3
$\mathbf{r}_9{}^T$	4/3	4/3	4/3



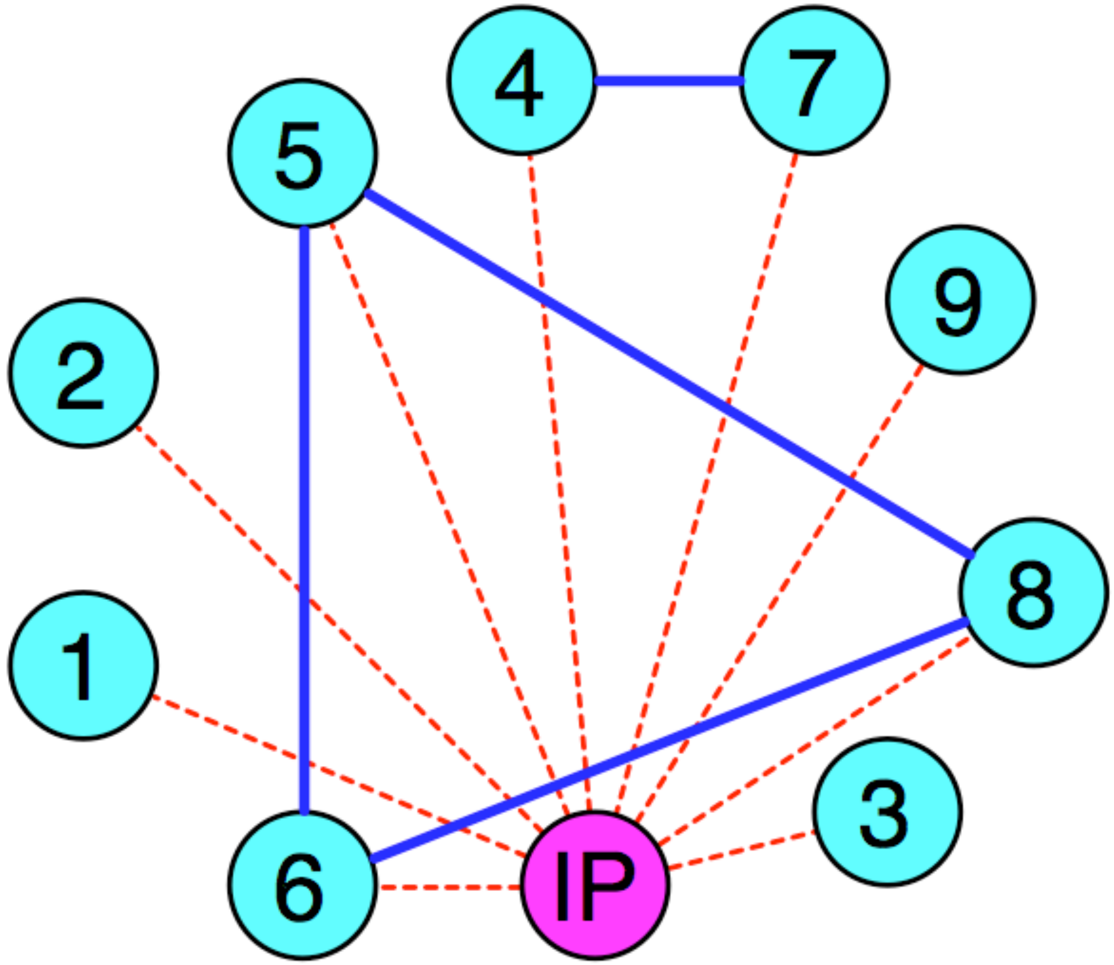
Graph Model Example - Inner Product Edges

$\mathbf{r_1}^T$	0	4/3	0
$\mathbf{r_2}^T$	0	0	1/2
$\mathbf{r_3}^T$	1/2	0	0
$\mathbf{r_4}^T$	1/6	1/6	0
$\mathbf{r_5}^T$	0	1/6	1/6
$\mathbf{r_6}^T$	0	-2/3	-2/3
$\mathbf{r_7}^T$	-4/3	-4/3	0
$\mathbf{r_8}^T$	0	-4/3	-4/3
$\mathbf{r_9}^T$	4/3	4/3	4/3



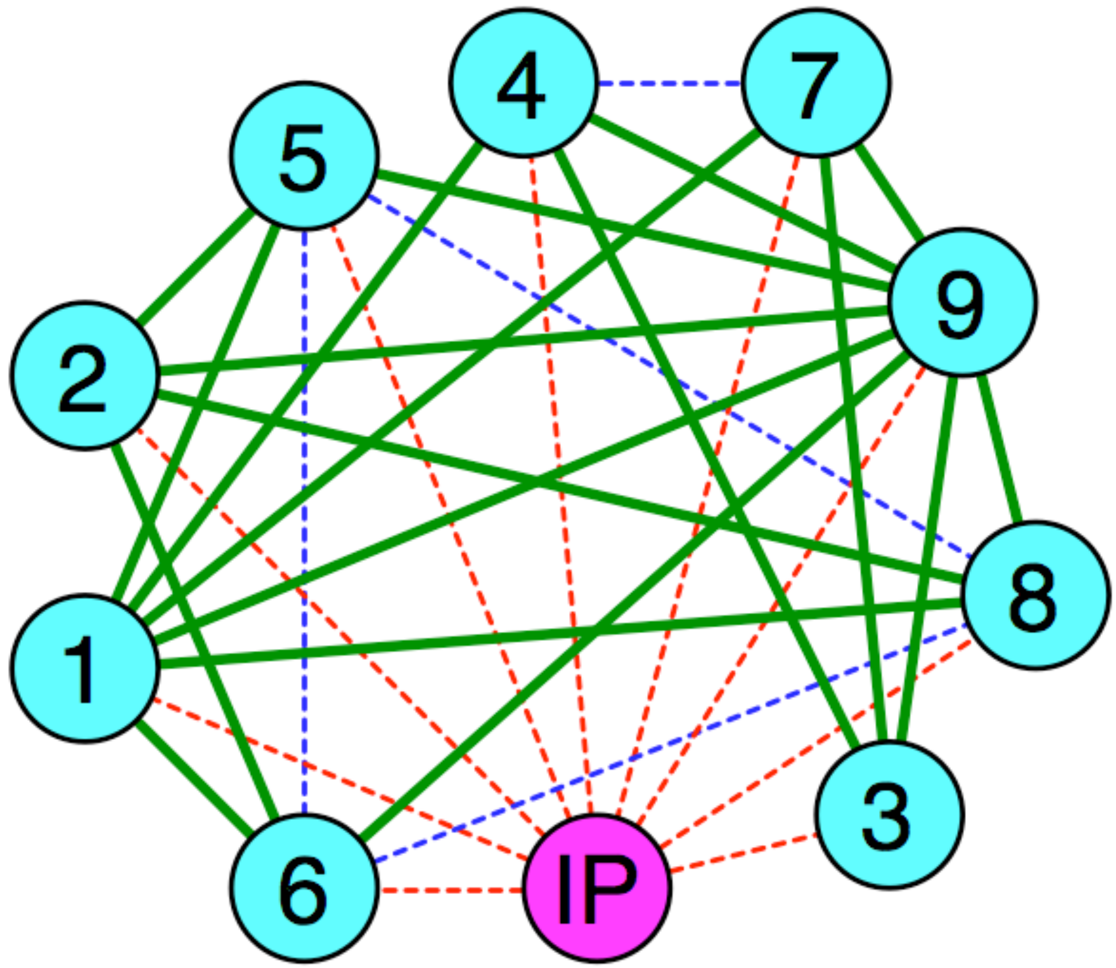
Graph Model Example - Collinear Edges

$\mathbf{r_1}^T$	0	4/3	0
$\mathbf{r_2}^T$	0	0	1/2
$\mathbf{r_3}^T$	1/2	0	0
$\mathbf{r_4}^T$	1/6	1/6	0
$\mathbf{r_5}^T$	0	1/6	1/6
$\mathbf{r_6}^T$	0	-2/3	-2/3
$\mathbf{r_7}^T$	-4/3	-4/3	0
$\mathbf{r_8}^T$	0	-4/3	-4/3
$\mathbf{r_9}^T$	4/3	4/3	4/3



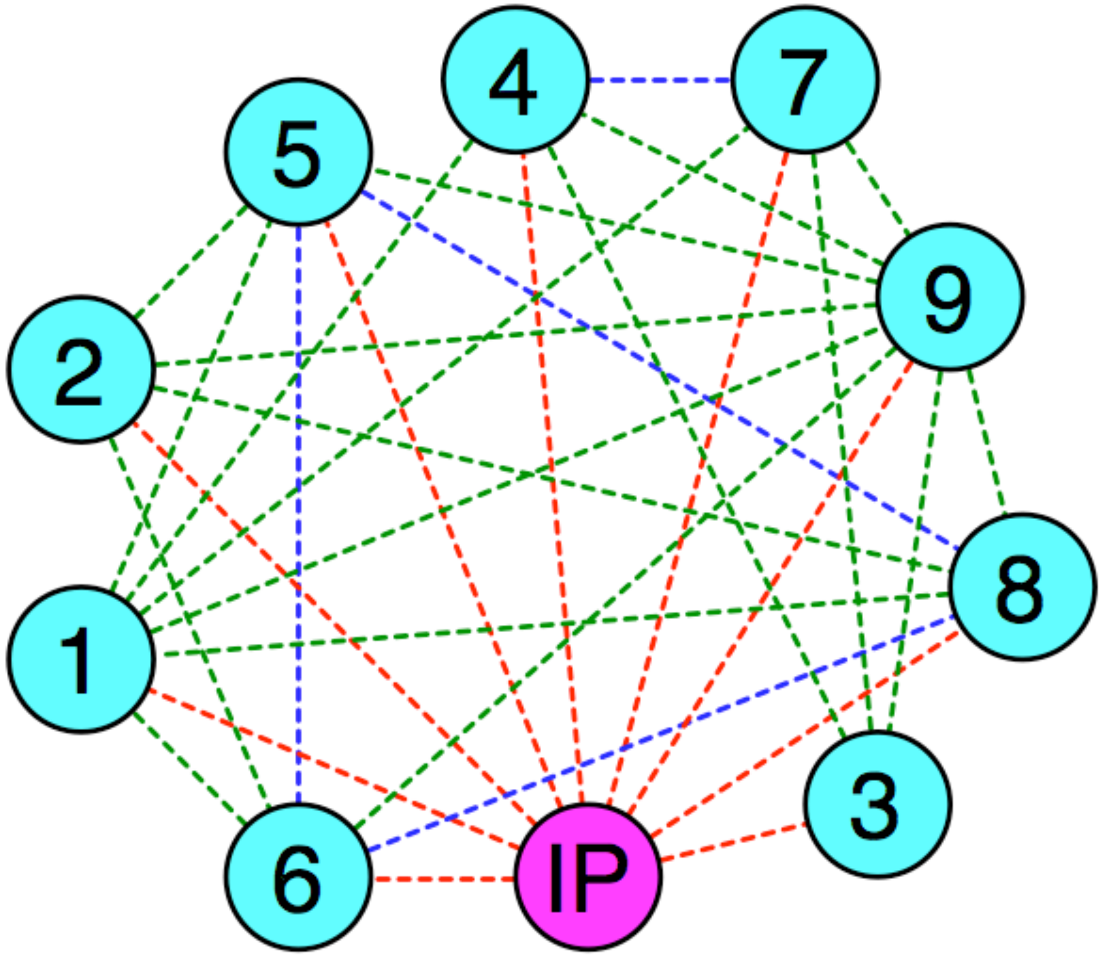
Graph Model Example - Partial Collinear Edges

$\mathbf{r}_1{}^T$	0	4/3	0
$\mathbf{r}_2{}^T$	0	0	1/2
$\mathbf{r}_3{}^T$	1/2	0	0
$\mathbf{r}_4{}^T$	1/6	1/6	0
$\mathbf{r}_5{}^T$	0	1/6	1/6
$\mathbf{r}_6{}^T$	0	-2/3	-2/3
$\mathbf{r}_7{}^T$	-4/3	-4/3	0
$\mathbf{r}_8{}^T$	0	-4/3	-4/3
$\mathbf{r}_9{}^T$	4/3	4/3	4/3



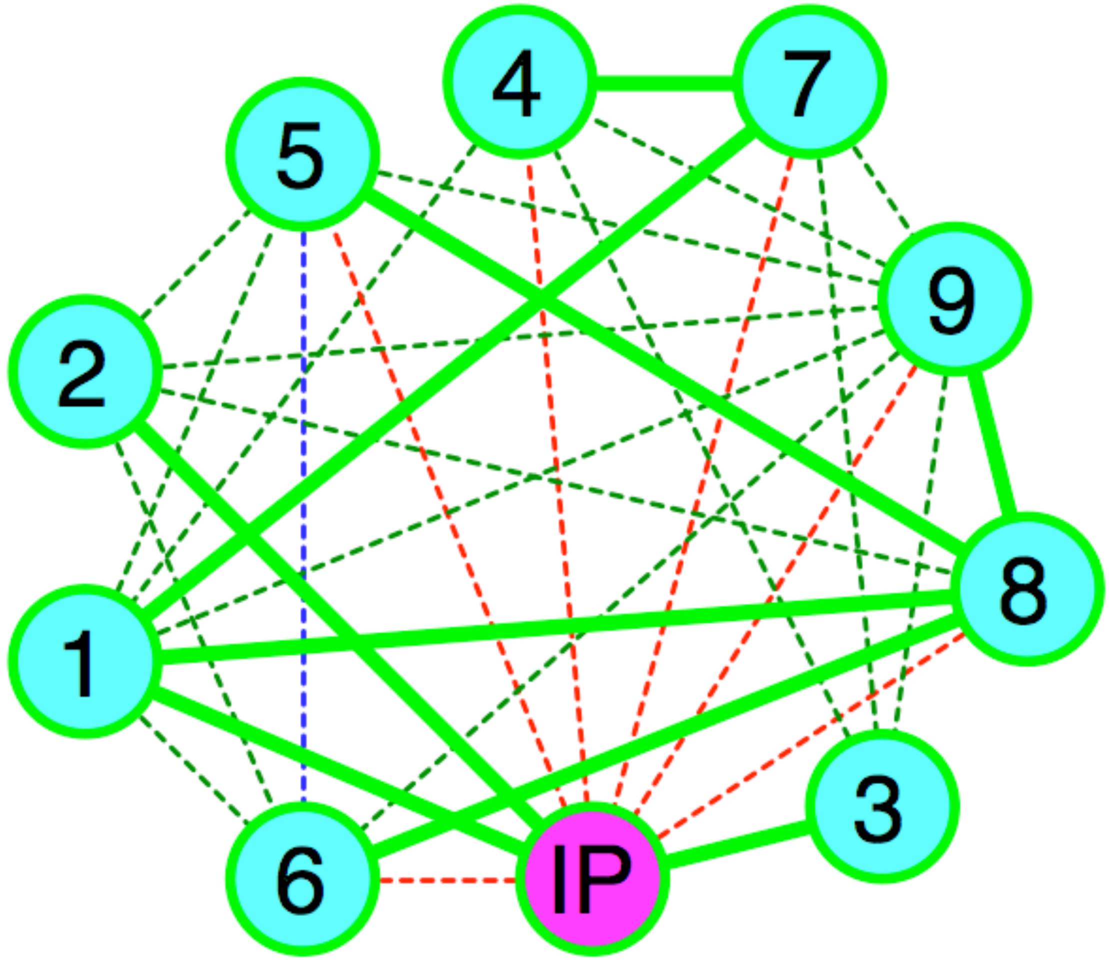
Graph Model Example - Final Graph

$\mathbf{r_1}^T$	0	4/3	0
$\mathbf{r_2}^T$	0	0	1/2
$\mathbf{r_3}^T$	1/2	0	0
$\mathbf{r_4}^T$	1/6	1/6	0
$\mathbf{r_5}^T$	0	1/6	1/6
$\mathbf{r_6}^T$	0	-2/3	-2/3
$\mathbf{r_7}^T$	-4/3	-4/3	0
$\mathbf{r_8}^T$	0	-4/3	-4/3
$\mathbf{r_9}^T$	4/3	4/3	4/3



Graph Model Example - Solution (MST)

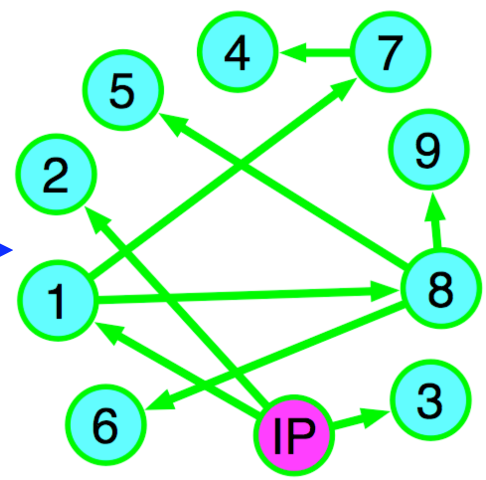
${\mathbf{r}_1}^T$	0	4/3	0
${\mathbf{r}_2}^T$	0	0	1/2
${\mathbf{r}_3}^T$	1/2	0	0
${\mathbf{r}_4}^T$	1/6	1/6	0
${\mathbf{r}_5}^T$	0	1/6	1/6
${\mathbf{r}_6}^T$	0	-2/3	-2/3
${\mathbf{r}_7}^T$	-4/3	-4/3	0
${\mathbf{r}_8}^T$	0	-4/3	-4/3
${\mathbf{r}_9}^T$	4/3	4/3	4/3



Graph Model Example - Instructions Generated

$\mathbf{r}_1{}^T$	0	4/3	0
$\mathbf{r}_2{}^T$	0	0	1/2
$\mathbf{r}_3{}^T$	1/2	0	0
$\mathbf{r}_4{}^T$	1/6	1/6	0
$\mathbf{r}_5{}^T$	0	1/6	1/6
$\mathbf{r}_6{}^T$	0	-2/3	-2/3
$\mathbf{r}_7{}^T$	-4/3	-4/3	0
$\mathbf{r}_8{}^T$	0	-4/3	-4/3
$\mathbf{r}_9{}^T$	4/3	4/3	4/3

Matrix (16 nz)



MST traversal

$$
\begin{aligned}
y_3 &= 0.5x_1 \\
y_2 &= 0.5x_3 \\
y_1 &= (4/3)x_2 \\
y_8 &= -y_1 - (4/3)x_3 \\
y_7 &= -y_1 - (4/3)x_1 \\
y_9 &= -y_8 + (4/3)x_1 \\
y_6 &= 0.5y_8 \\
y_5 &= (-1/8)y_8 \\
y_4 &= (-1/8)y_7
\end{aligned}
$$

Instructions (9 MAPs)

Graph Model Results - 2D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs
1	10	**7**
2	34	**14**
3	108	**43**
4	292	**152**
5	589	**366**
6	1070	**686**

← 60% decrease

- Graph model shows significant improvement over unoptimized algorithm

Graph Model Results - 3D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs
1	21	**17**
2	177	**79**
3	789	**342**
4	2586	**1049**
5	7125	**3592**
6	16749	**8835**

59% decrease

- Again graph model requires significantly fewer MAPs than unoptimized algorithm

Limitation of Graph Model

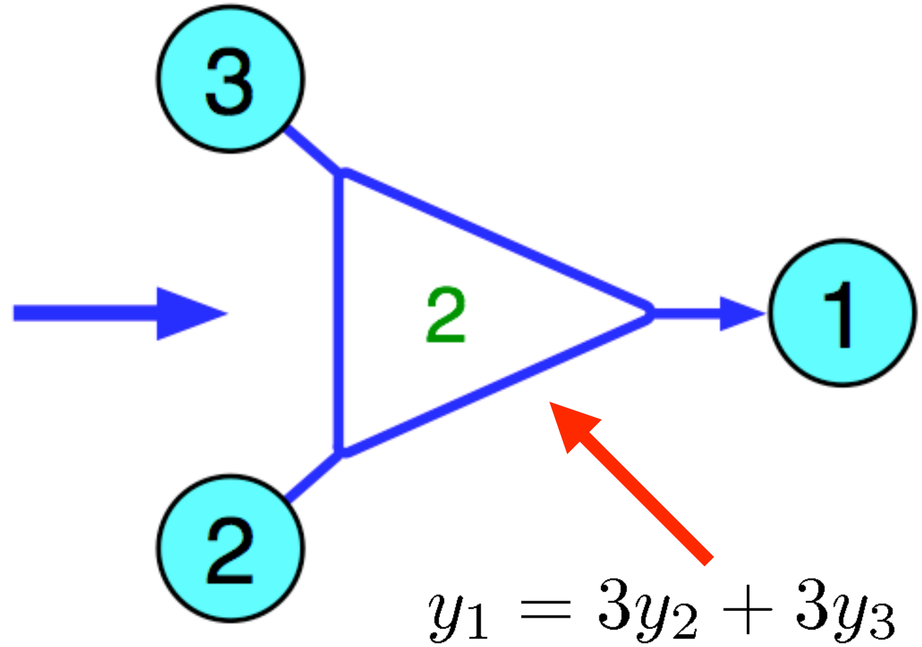
$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 4 & 4 & 4 & 4 \\ 0 & 0 & 2 & 2 \\ 2 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}$$

$$\mathbf{r_2} = 2\mathbf{r_3} + 2\mathbf{r_4} \Rightarrow y_2 = 2y_3 + 2y_4$$

- Edges connect 2 vertices
- Can represent only binary row relationships
- Cannot exploit linear dependency of more than two rows
- Thus, hypergraphs needed

Hypergraph Model

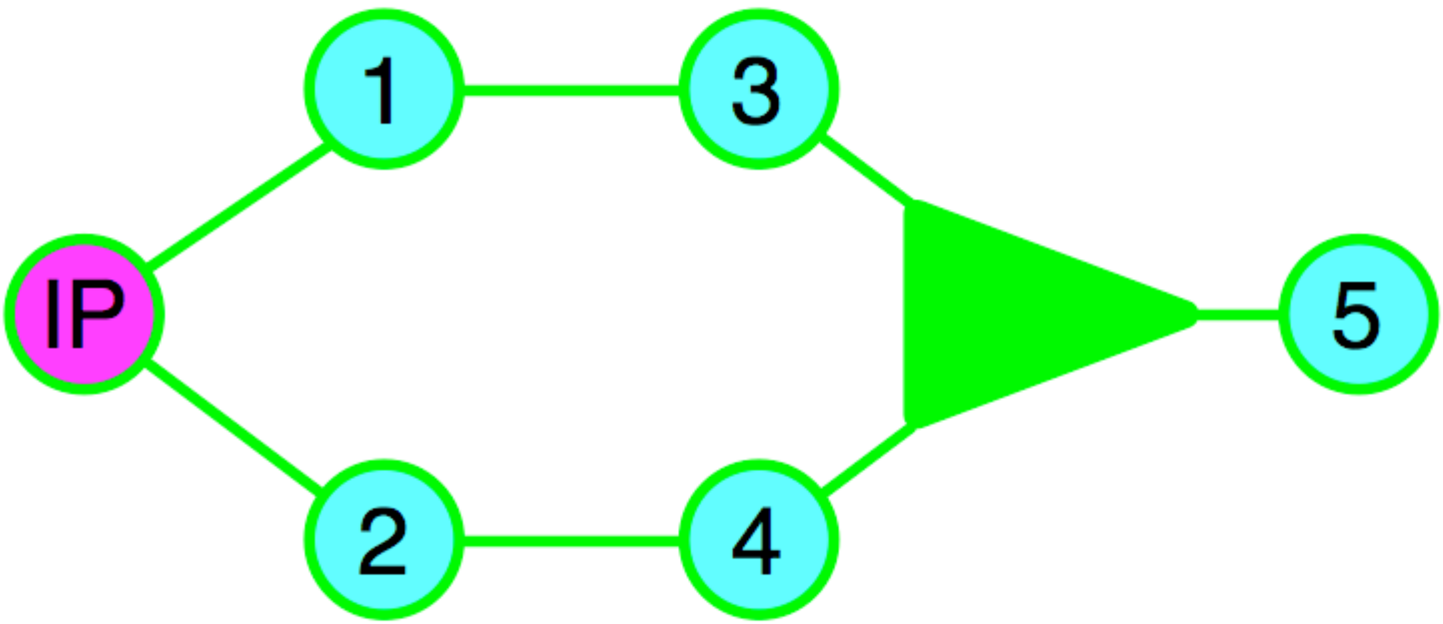
$$
\begin{array}{c|c:c:c:c|}
\hline
\mathbf{r_1}^T & 3 & 3 & 3 & 3 \\
\hline
\mathbf{r_2}^T & 1 & 1 & 0 & 0 \\
\hline
\mathbf{r_3}^T & 0 & 0 & 1 & 1 \\
\hline
\end{array}
$$



- Same edges (2-vertex hyperedges) as graph model
- Additional higher cardinality hyperedges for more complicated relationships
  - Limiting to 3-vertex linear dependency hyperedges for this talk

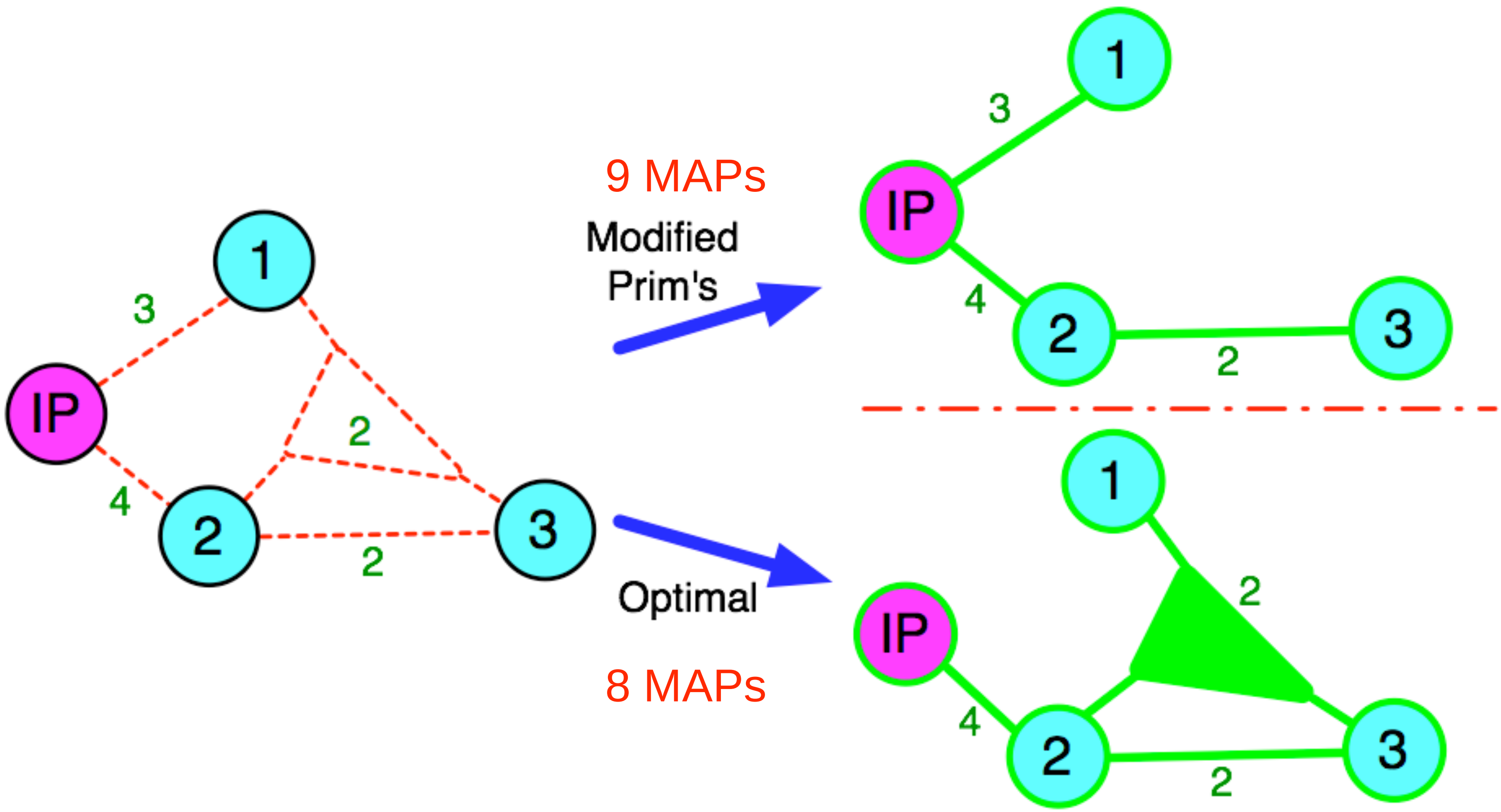
Hypergraph Model Solution: Modified Prim's

- Extended Prim's algorithm to include hyperedges
- Polynomial time algorithm
- Solution not necessarily a tree
  - {IP,1,3,5}
  - {IP,2,4,5}
- No guarantee of optimum solution



Hypergraph Model Solution: Modified Prim's

- No guarantee of optimum solution



Hypergraph Model Results - 2D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs	HGraph MAPs
1	10	7	**6**
2	34	**14**	**14**
3	108	43	**38**
4	292	152	**131**
5	589	366	**352**
6	1070	686	**677**

14% additional decrease

- Hypergraph solution shows modest improvement over graph solution
- Graph algorithm solutions close to optimal for some orders?
  - 3 Columns

Hypergraph Model Results - 3D Laplace Equation

Order	Unoptimized MAPs	Graph MAPs	HGraph MAPs
1	21	**17**	**17**
2	177	79	**65**
3	789	342	**262**
4	2586	1049	**760**
5	7125	3592	**3165**
6	16749	8835	**8228**

28% additional decrease

- Hypergraph solution significantly better than graph solution for many orders

Ongoing Work: New Hypergraph Method(s)

- Greedy modified Prim's algorithm yields suboptimal solutions for hypergraphs
- Want improved method that yields better (or optimal) solutions
  - Improved solution
  - Optimality of greedy solution
- New approach: formulate as vertex ordering

Ongoing Work: Vertex Ordering Method

- Order vertices
  - Roughly represents order of calculation for entries
- For given vertex ordering, can determine optimal solution subhypergraph
  - Greedy algorithm of selecting cheapest available hyperedge
  - Fast!
- Ordering is challenging part
- Traversal of greedy solution good starting pt.
- Implemented very simple local refinement
  - Local refinement on starting point

Vertex Ordering: Preliminary Results — 2D Laplace

Order	Graph MAPs	HGraph MAPs	Local Refine MAPs
1	7	**6**	**6**
2	**14**	**14**	**14**
3	43	**38**	**38**
4	152	**131**	**129**
5	366	**352**	**339**
6	686	**677**	**657**

4% additional decrease

- Simple local refinement method
  - Pairwise swapping to improve initial ordering
- Slight additional improvement
- Graph/hypergraph solutions close to optimal?
  - 3 Columns

Vertex Ordering: Preliminary Results — 3D Laplace

Order	Graph MAPs	HGraph MAPs	Local Refine MAPs
1	**17**	**17**	**17**
2	79	**65**	**65**
3	342	262	**239**
4	1049	760	**757**
5	3592	3165	**3105**
6	8835	8228	**8141**

9% additional decrease

- Simple local refinement method
  - Pairwise swapping to improve initial ordering
- Slight additional improvement
- Perhaps more improvement with global vertex ordering method

Future Work

- Higher cardinality hyperedges
  - Perhaps useful for 3D FE problems
  - Implemented 4, 5, 6 vertex hyperedges
  - Hyperedge explosion
  - Need efficient hyperedge pruning algorithms
- Improve hypergraph solution methods
  - Develop global vertex ordering method
- Focus on reducing runtime of resulting operations
  - Best feasible vertex ordering for given solution
- Other matrices

Acknowledgements/Thanks

- Michael Heath, advisor
- Robert Kirby, Texas Tech University
- Erik Boman, Sandia National Laboratories
- Bruce Hendrickson, Sandia National Laboratories
- Funding
  - DOE CSGF, DE-FG02-97ER25308
  - SIAM -- travel support
  - CSCAPES (Sandia) -- travel support

Extra

2D Laplace Equation Matrices

Order	Rows	Entries	Nonzeros
1	6	18	10
2	21	63	34
3	55	165	108
4	120	360	292
5	231	693	589
6	406	1218	1070

- 3 Columns

3D Laplace Equation Matrices

Order	Rows	Entries	Nonzeros
1	10	60	21
2	55	330	177
3	210	1260	789
4	630	3780	2586
5	1596	9576	7125
6	3570	21420	16749

- 6 Columns

Accuracy

Relative Error 2D Laplace

Order	GPCR Error	HGraph Error
1	0	0
2	2.53565e-09	2.55594e-09
3	6.40668e-09	2.44340e-09
4	2.47834e-10	9.30090e-09
5	4.95544e-09	5.87721e-09
6	4.28141e-09	4.28166e-09

Relative Error 3D Laplace

Order	GPCR Error	HGraph Error
1	0	0
2	9.33830e-09	7.35996e-09
3	2.60053e-08	3.51190e-08
4	8.31206e-09	1.47134e-08
5	4.22496e-08	6.30277e-08
6	1.07992e-06	1.41391e-06

- Single precision input matrices
- Single precision code generation