Hypergraph-Based
Combinatorial Optimization of
Matrix-Vector Multiplication

Preliminary Exam — 4/16/2008
Michael Wolf
Color Scheme of Text

- Original contribution

- Future work
  - Ongoing research
  - Proposed research
Combinatorial Optimization of Mat-Vec Multiplication

• Two main subtopics

• Serial matrix-vector multiplication
  - Reducing redundant operations
  - Dense relatively small matrices

• Parallel matrix-vector multiplication
  - Minimization of communication volume
  - Large, sparse matrices
Optimization of Serial Mat-Vec Multiplication

Motivation:

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

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

Can use optimized code for every element in domain
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 to evaluate finite element matrices
  - Equivalent to optimizing matrix-vector product code
Matrix-Vector Multiplication

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

\[ S^e_{i,j} = y_{ni+j} = A_{(ni+j,*)} \mathbf{x} \]

where

\[
A^T_{(ni+j,*)} = \left[ \begin{array}{c}
\left( \frac{\partial \phi_i}{\partial r}, \frac{\partial \phi_i}{\partial s} \right) \\
\left( \frac{\partial \phi_j}{\partial r}, \frac{\partial \phi_j}{\partial s} \right) \\
\left( \frac{\partial \phi_i}{\partial s}, \frac{\partial \phi_j}{\partial s} \right) \\
\left( \frac{\partial \phi_j}{\partial s}, \frac{\partial \phi_j}{\partial s} \right)
\end{array} \right] \begin{array}{c} \hat{e} \\ \hat{e} \\ \hat{e} \end{array}
\]

\[ \mathbf{x} = \text{det}(J) \]

\[
\begin{bmatrix}
\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\
\frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} \\
\frac{\partial r}{\partial x} & \frac{\partial r}{\partial y} \\
\frac{\partial s}{\partial x} & \frac{\partial s}{\partial y}
\end{bmatrix}
\]

Element independent

Element dependent
Optimization Problem

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

\[ y = Ax \]

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_n
\end{bmatrix}
= \begin{bmatrix}
  r_1^T \\
  \vdots \\
  r_m^T
\end{bmatrix}
\begin{bmatrix}
  x
\end{bmatrix}
= \begin{bmatrix}
  r_1^T x \\
  \vdots \\
  r_m^T x
\end{bmatrix}
Possible Optimizations - Collinear Rows

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

\[r_2 = 1.5r_1\]
Possible Optimizations - Colinear Rows

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

\[r_2 = 1.5r_1 \Rightarrow y_2 = 1.5y_1\]

1 MAP
Possible Optimizations - Identical Rows

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

\[r_3 = r_1 \Rightarrow y_3 = y_1\]

0 MAPs

Special case when rows identical
Possible Optimizations - Partial Colinear Rows

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

\[r_4 = 2.5r_1 + 8e_4\]
Possible Optimizations - Partial Collinear Rows

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

\[r_4 = 2.5r_1 + 8e_4 \Rightarrow y_4 = 2.5y_1 + 8x_4\]

2 MAPs
Graph Model - Resulting Vector Entry Vertices

\[ r_1^T = \begin{bmatrix} 8 & 4 & 4 & 8 \end{bmatrix}, \quad r_2^T = \begin{bmatrix} 0 & -2 & -2 & -4 \end{bmatrix} \]

- Entries in resulting vector represented by vertices in graph model
Graph Model - Inner-Product Vertex and Edges

\[ y_2 = -2x_2 - 2x_3 - 4x_4 \]

\[ \begin{bmatrix} r_1^T \\ r_2^T \end{bmatrix} = \begin{bmatrix} 8 & 4 & 4 & 8 \\ 0 & -2 & -2 & -4 \end{bmatrix} \]

\[ y_1 = 8x_1 + 4x_2 + 4x_3 + 8x_4 \]

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

\[
\begin{array}{c}
\begin{array}{cccc}
\mathbf{r}_1^T & 8 & 4 & 4 & 8 \\
\mathbf{r}_2^T & 0 & -2 & -2 & -4 \\
\end{array}
\end{array}
\]

\[
y_1 = -2y_2 + 8x_1 \\
y_2 = -0.5y_1 + 4x_1
\]

- Operations resulting from relationships between rows represented by edges between corresponding vertices
Graph Model - Edge Weights

\[ y_2 = -2x_2 - 2x_3 - 4x_4 \]

\[
\begin{pmatrix}
  r_1^T \\
  r_2^T
\end{pmatrix} = \begin{pmatrix}
  8 & 4 & 4 & 8 \\
  0 & -2 & -2 & -4
\end{pmatrix}
\]

\[ y_1 = 8x_1 + 4x_2 + 4x_3 + 8x_4 \]

\[
y_1 = -2y_2 + 8x_1 \\
y_2 = -0.5y_1 + 4x_1
\]

- Edge weights are MAP costs for operations
Graph Model Solution

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

Matrix

<table>
<thead>
<tr>
<th></th>
<th>8</th>
<th>4</th>
<th>4</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-2</td>
<td>-2</td>
<td>-4</td>
<td></td>
</tr>
</tbody>
</table>

Graph

MST(5)
Graph Model Solution

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

\[
\begin{pmatrix}
8 & 4 & 4 & 8 \\
0 & -2 & -2 & -4
\end{pmatrix}
\]

\[
y_2 = -2x_2 - 2x_3 - 4x_4 \\
y_1 = -2y_2 + 8x_1
\]
Graph Model Results - 2D Laplace Equation

- Graph model shows significant improvement over unoptimized algorithm

<table>
<thead>
<tr>
<th>Order</th>
<th>Unoptimized MAPs</th>
<th>Graph MAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>292</td>
<td>152</td>
</tr>
<tr>
<td>5</td>
<td>589</td>
<td>366</td>
</tr>
<tr>
<td>6</td>
<td>1070</td>
<td>686</td>
</tr>
</tbody>
</table>

60% decrease
Graph Model Results - 2D Laplace Equation

<table>
<thead>
<tr>
<th>Order</th>
<th>Unoptimized MAPs</th>
<th>FErari MAPs</th>
<th>Graph MAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>15</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>45</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>292</td>
<td>176</td>
<td>152</td>
</tr>
<tr>
<td>5</td>
<td>589</td>
<td>443</td>
<td>366</td>
</tr>
<tr>
<td>6</td>
<td>1070</td>
<td>867</td>
<td>686</td>
</tr>
</tbody>
</table>

21% decrease

- Improved graph model shows significant improvement over FErari
Graph Model Results - 3D Laplace Equation

<table>
<thead>
<tr>
<th>Order</th>
<th>Unoptimized MAPs</th>
<th>Graph MAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>177</td>
<td>79</td>
</tr>
<tr>
<td>3</td>
<td>789</td>
<td>342</td>
</tr>
<tr>
<td>4</td>
<td>2586</td>
<td>1049</td>
</tr>
<tr>
<td>5</td>
<td>7125</td>
<td>3592</td>
</tr>
<tr>
<td>6</td>
<td>16749</td>
<td>8835</td>
</tr>
</tbody>
</table>

- Again graph model requires significantly fewer MAPs than unoptimized algorithm

59% decrease
Graph Model Results - 3D Laplace Equation

<table>
<thead>
<tr>
<th>Order</th>
<th>Unoptimized MAPs</th>
<th>FErari MAPs</th>
<th>Graph MAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>–</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>177</td>
<td>101</td>
<td>79</td>
</tr>
<tr>
<td>3</td>
<td>789</td>
<td>370</td>
<td>342</td>
</tr>
<tr>
<td>4</td>
<td>2586</td>
<td>1118</td>
<td>1049</td>
</tr>
<tr>
<td>5</td>
<td>7125</td>
<td>–</td>
<td>3592</td>
</tr>
<tr>
<td>6</td>
<td>16749</td>
<td>–</td>
<td>8835</td>
</tr>
</tbody>
</table>

• Again graph model requires significantly fewer MAPs than FErari

22% decrease
Limitation of Graph Model

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

\[
r_2 = 2r_3 + 2r_4 \Rightarrow y_2 = 2y_3 + 2y_4
\]

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

- 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

\[
y_1 = 3y_2 + 3y_3
\]
Hypergraph Model

- 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 Results - 2D Laplace Equation

<table>
<thead>
<tr>
<th>Order</th>
<th>Unoptimized MAPs</th>
<th>Graph MAPs</th>
<th>HGraph MAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>7</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>14</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>108</td>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>292</td>
<td>152</td>
<td>150</td>
</tr>
<tr>
<td>5</td>
<td>589</td>
<td>366</td>
<td>363</td>
</tr>
<tr>
<td>6</td>
<td>1070</td>
<td>686</td>
<td>686</td>
</tr>
</tbody>
</table>

- Hypergraph solution slightly better for some orders but not significantly better
- Graph algorithm solution close to optimal?
  - 3 Columns
  - Binary relationships may be good enough
## Hypergraph Model Results - 3D Laplace Equation

<table>
<thead>
<tr>
<th>Order</th>
<th>Unoptimized MAPs</th>
<th>Graph MAPs</th>
<th>HGraph MAPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>2</td>
<td>177</td>
<td>79</td>
<td>68</td>
</tr>
<tr>
<td>3</td>
<td>789</td>
<td>342</td>
<td>297</td>
</tr>
<tr>
<td>4</td>
<td>2586</td>
<td>1049</td>
<td>852</td>
</tr>
<tr>
<td>5</td>
<td>7125</td>
<td>3592</td>
<td>3261</td>
</tr>
<tr>
<td>6</td>
<td>16749</td>
<td>8835</td>
<td>8340</td>
</tr>
</tbody>
</table>

- Hypergraph solution significantly better than graph solution for many orders

19% additional decrease
Future 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

- First approach: integer programming method
  - Express valid hypergraph solution more formally
  - Exponential number of variables/constraints discouraging

- New approach: formulate as vertex ordering
Future Work: Vertex Ordering Method

• Order vertices
  - Roughly represents order of calculation for entries
• For given ordering, can determine optimal solution subhypergraph!
  - Greedy algorithm of selecting cheapest available hyperedge
  - Fast
• Ordering is challenging part
  - Traversal of greedy solution good starting point
  - Local refinement on starting point
• Develop global ordering method
Future Work: Hyperedge Detection/Construction

- Hyperedge detection/construction is bottleneck
- Currently brute force operation (nested loops)
  - e.g. $O(n^3)$ calls to coplanar detection kernel for $n$ rows
- Detection kernel: originally SVD, now hybrid

<table>
<thead>
<tr>
<th>Matrix</th>
<th>n</th>
<th>Orig Time (s)</th>
<th>Hybrid Time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2DP5</td>
<td>231</td>
<td>9.1</td>
<td>1.4</td>
</tr>
<tr>
<td>2DP6</td>
<td>406</td>
<td>50.8</td>
<td>4.8</td>
</tr>
<tr>
<td>3DP3</td>
<td>210</td>
<td>4.8</td>
<td>0.4</td>
</tr>
<tr>
<td>3DP4</td>
<td>630</td>
<td>115.4</td>
<td>7.3</td>
</tr>
<tr>
<td>3DP5</td>
<td>1596</td>
<td>1921.9</td>
<td>117.5</td>
</tr>
<tr>
<td>3DP6</td>
<td>3570</td>
<td>26510.9</td>
<td>1248.2</td>
</tr>
</tbody>
</table>

- Improvement over brute force method
  - Better complexity than $O(n^3)$
Future Work: Hyperedge Pruning

- Hyperedge explosion
  - Over 10 million hyperedges for FE matrices
  - Hypergraphs too large to fit on one processor
- Most hyperedges won’t be in optimal solution
- Want to prune as many as possible
- For example, currently prune
  - Hyperedge if weight greater or equal than number of nonzeros for all involved vertices
  - Coplanar (3 V) hyperedge if two of rows are collinear
- Need additional pruning heuristics
  - One possibility: use graph solution
Future Work: Miscellaneous

• Runtime of resulting operations
  - Preliminary studies show slight improvement
  - Not as good as MAP improvement
  - More complete study necessary

• Better instruction ordering
  - Currently do naive traversal of solution subgraph
  - Can do something more clever/cache-friendly
  - Solution is dependency graph
Sparse Matrix Partitioning

- Work with Dr. Erik Boman (SNL)
  - CSCAPES Institute
- Researched and developed two new two-dimensional methods
- If successful, will be implemented as part of new matrix partitioning suite in Zoltan
Parallel Sparse Matrix-Vector Multiplication

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  y_3 \\
  y_4 \\
  y_5 \\
  y_6 \\
  y_7 \\
  y_8 \\
\end{bmatrix}
= 
\begin{bmatrix}
  1 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  5 & 1 & 9 & 0 & 5 & 0 & 0 & 0 & 0 \\
  0 & 8 & 1 & 7 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 & 1 & 8 & 0 & 0 & 1 \\
  4 & 0 & 0 & 0 & 3 & 1 & 3 & 0 & 4 \\
  0 & 0 & 0 & 6 & 0 & 9 & 1 & 4 & 2 \\
  0 & 0 & 0 & 0 & 0 & 0 & 2 & 1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
  1 \\
  2 \\
  4 \\
  3 \\
  1 \\
  4 \\
  2 \\
  1 \\
\end{bmatrix}
\]

\[y = Ax\]

- Partition matrix nonzeros
- Partition vectors
Objective

• Ideally we minimize total run-time
• Settle for easier objective
  - Work balanced
  - Minimize total communication volume
• Can partition matrices in different ways
  - 1-D
  - 2-D
• Can model problem in different ways
  - Graph
  - Bipartite graph
  - Hypergraph
1-D Partitioning

1-D Column

- Each process assigned nonzeros for set of columns

1-D Row

- Each process assigned nonzeros for set of rows
When 1-D Partitioning is Inadequate

- For any 1-D bisection of nxn arrowhead matrix:
  - $\text{nnz} = 3n-2$
  - Volume $\approx (3/4)n$
- $O(k)$ volume partitioning possible

“Arrowhead” matrix
n=12
nnz=34 (18,16)
volume = 9

37
2-D Partitioning

- More flexibility in partitioning
- No particular part for given row or column
- More general sets of nonzeros assigned parts

- Fine-grain hypergraph model
  - Ultimate flexibility
  - Assign each nz separately

- Corner symmetric partitioning method
- Graph model for symmetric 2-D partitioning
- Nested dissection symmetric partitioning method
Fine-Grain Hypergraph Model

- Catalyurek and Aykanat (2001)
- Nonzeros represented by vertices in hypergraph
Fine-Grain Hypergraph Model

- Rows represented by hyperedges
Fine-Grain Hypergraph Model

- Columns represented by hyperedges
Fine-Grain Hypergraph Model

- 2n hyperedges
Fine-Grain Hypergraph Model

<table>
<thead>
<tr>
<th></th>
<th>h9</th>
<th>h10</th>
<th>h11</th>
<th>h12</th>
<th>h13</th>
<th>h14</th>
<th>h15</th>
<th>h16</th>
</tr>
</thead>
<tbody>
<tr>
<td>h1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h2</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h3</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h4</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>h6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h8</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

- Partition vertices into $k$ equal sets
- For $k=2$
  - Volume = number of hyperedges cut
- Minimum volume partitioning when optimally solved
- Larger NP-hard problem than 1-D

$k=2$, volume = cut = 2
New 2-D Method: “Corner” Symmetric Partitioning

- Optimal partitioning of arrowhead matrix suggests new partitioning method
"Corner" Symmetric Partitioning

- 1-D parts reflected across diagonal
"Corner" Symmetric Partitioning

- Take lower triangular portion of matrix
“Corner” Symmetric Partitioning

- 1-D (column) hypergraph partitioning of lower triangular matrix
“Corner” Symmetric Partitioning

- Reflect partitioning symmetrically across diagonal
“Corner” Symmetric Partitioning

- Optimal partitioning

Volume = 2
Comparison of Methods -- Arrowhead Matrix

<table>
<thead>
<tr>
<th>k</th>
<th>1D column</th>
<th>Corner</th>
<th>Fine grain</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>29101</td>
<td>2*</td>
<td>2*</td>
</tr>
<tr>
<td>4</td>
<td>40001</td>
<td>6*</td>
<td>6*</td>
</tr>
<tr>
<td>16</td>
<td>40012</td>
<td>30*</td>
<td>30*</td>
</tr>
<tr>
<td>64</td>
<td>40048</td>
<td>126*</td>
<td>126*</td>
</tr>
</tbody>
</table>

• n = 40,000  
• nnz = 119,998  
• Communication volume for 3 methods

*optimal

Order n  
2(k-1)
Preliminary Results

<table>
<thead>
<tr>
<th>Name</th>
<th>N</th>
<th>nnz</th>
<th>nz/N</th>
<th>nz/(N)²</th>
</tr>
</thead>
<tbody>
<tr>
<td>cage10</td>
<td>11,397</td>
<td>150,645</td>
<td>13.2</td>
<td>1.16 × 10⁻³</td>
</tr>
<tr>
<td>finan512</td>
<td>74,752</td>
<td>596,992</td>
<td>8.0</td>
<td>1.07 × 10⁻⁴</td>
</tr>
<tr>
<td>bcsstk30</td>
<td>28,924</td>
<td>2,043,492</td>
<td>70.7</td>
<td>2.44 × 10⁻³</td>
</tr>
<tr>
<td>asic680ks</td>
<td>682,712</td>
<td>2,329,176</td>
<td>3.4</td>
<td>5.00 × 10⁻⁶</td>
</tr>
</tbody>
</table>

- Symmetric matrices
- First 3 from Professor Rob Bisseling’s (Utrecht University) Mondriaan paper
- Last from Sandia Xyce circuit simulation

- Hypergraph partitioning for all methods
  - Zoltan with PaToH
## Preliminary Results: Communication Volume

<table>
<thead>
<tr>
<th>Name</th>
<th>k</th>
<th>1d hyp.Col</th>
<th>fine-grain hyp.</th>
<th>corner</th>
</tr>
</thead>
<tbody>
<tr>
<td>cage10</td>
<td>2</td>
<td>2308.2</td>
<td>1879.6</td>
<td>1866.6</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>5379.0</td>
<td>4063.7</td>
<td>4089.3</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>12874.5</td>
<td>8865.5</td>
<td>8920.9</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>23463.3</td>
<td>16334.7</td>
<td>17164.0</td>
</tr>
<tr>
<td>finan512</td>
<td>2</td>
<td>147.8</td>
<td>126.1</td>
<td>100.0</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>295.7</td>
<td>261.2</td>
<td>215.0</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>1216.7</td>
<td>1027.4</td>
<td>845.0</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>9986.0</td>
<td>8624.6</td>
<td>8135.2</td>
</tr>
<tr>
<td>bcsstk30</td>
<td>2</td>
<td>605.6</td>
<td>662.6</td>
<td>618.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1794.4</td>
<td>1935.7</td>
<td>1531.0</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>8624.7</td>
<td>9774.8</td>
<td>7232.2</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>23308.0</td>
<td>25677.2</td>
<td>20351.4</td>
</tr>
<tr>
<td>asic680ks</td>
<td>2</td>
<td>1543.5</td>
<td>686.6</td>
<td>936.9</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>3560.4</td>
<td>1813.3</td>
<td>2214.2</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>9998.5</td>
<td>4634.0</td>
<td>5562.8</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>21785.8</td>
<td>9554.9</td>
<td>11147.3</td>
</tr>
</tbody>
</table>
Future Work: Reordering

• Ordering not advantageous in 1D methods
  - Same graphs/hypergraph models
• Corner method partitioning quality depends greatly on ordering
  - Ordering impacts off-diagonal nz partitioning

• Symmetric reordering to further reduce communication
• Focus on bisection
  - Recursive bisection for k>2
Reordering (Bisection)

• Graph model $G(V,E)$
  - Vector entries represented by vertices
  - Off-diagonal nonzeros represented by edges
  - Each vertex $v_i$ assigned part $s_i$ and position $\pi_i$

• $v_i$ “costs” 2 words of communication iff
  $$\exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j$$

• $v_i$ “free” otherwise
• $v_i$ “costs” 2 words if
\[ \exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j \]
Reordering (Bisection)

• Ideally find optimal partitioning/ordering
  - Very difficult combinatorial problem
• Instead we propose
  - Fix ordering, partition
    • Corner method
  - Fix vertex partitioning, find optimal ordering
• Can iterate two steps

• Need to find optimal vertex ordering given fixed vertex partitioning
  - Divide graph into 3 categories of vertices
• Interior vertices: not adjacent to any vertex owned by different part
• Boundary vertices: adjacent to at least one vertex owned by different part
Reordering (Bisection): Vertex Categories

- Bipartite graph obtained by
  - Removing interior vertices
  - Removing non-cut edges
Reordering (Bisection): Vertex Categories

- Minimum vertex cover of bipartite graph
  - Cover boundary vertices
Reordering (Bisection): Vertex Categories

- Non-cover boundary vertices
Reordering (Bisection): Vertex Categories

• 3 Categories
  - Interior vertices
  - Non-cover boundary vertices
  - Cover boundary vertices
Reordering (Bisection): Ordering Interior V

• \( v_i \) “costs” if
  \[ \exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j \]

• Interior vertices can be given any position with no affect on volume
  - Since adjacent vertices have same part
  - Position these first
Reordering (Bisection): Ordering Other V

• \( v_i \) "costs" if 
  \[ \exists v_j : (v_i, v_j) \in E, s_i \neq s_j, \pi_i > \pi_j \]

• Find ordering of remaining V such that
  - Minimum set of vertices result in communication
  - Equivalently, minimum set of vertices such that for each edge in bipartite graph, vertex with larger numbered position is contained in this set
  - Minimum vertex cover gives us this set
    • With cover vertices ordered last

• Order cover boundary vertices last
• Only cover boundary vertices “cost”
Graph Model for Symmetric 2-D Partitioning

- Given symmetric matrix $A$
- Symmetric partition
  - $a(i,j)$ and $a(j,i)$ assigned same partition
  - Input and output vectors have same distribution
- Corresponding graph $G(V,E)$
  - Vertices correspond to vector elements
  - Edges correspond to off-diagonal nonzeros
Graph Model for Symmetric 2-D Partitioning

- Corresponding graph $G(V,E)$
  - Vertices correspond to vector elements
  - Edges correspond to off-diagonal nonzeros
Graph Model for Symmetric 2-D Partitioning

- Symmetric 2-D partitioning
  - Partition both V and E
  - Gives partitioning of both matrix and vectors
• Communication is assigned to vertices
• Vertex incurs communication iff incident edge is in different part
• Want small vertex separator -- $S=\{V_8\}$
Nested Dissection Partitioning Method - Bisection

- Suppose $A$ is symmetric
- Let $G(V,E)$ be graph of $A$
- Find small, balanced separator $S$
  - Yields vertex partitioning $V = (V_0, V_1, S)$
- Partition the edges
  - $E_0 = \{\text{edges that touch a vertex in } V_0\}$
  - $E_1 = \{\text{edges that touch a vertex in } V_1\}$
Nested Dissection Partitioning Method - Bisection

- Vertices in $S$ and corresponding edges
  - Can be assigned to either partition
  - Can use flexibility to maintain balance
- Communication Volume = $2|S|$
  - Regardless of $S$ partitioning
  - $|S|$ in each phase
Nested Dissection Partitioning Method

- Recursive bisection to partition into >2 partitions
- Use nested dissection!
Preliminary Numerical Experiments

- Compared 3 methods
  - 1-D hypergraph partitioning
  - Fine-grain hypergraph partitioning
  - Nested dissection partitioning
- Hypergraph partitioning for all methods
  - Zoltan with PaToH
- Symmetric and nonsymmetric matrices
  - Mostly from Prof. Rob Bisseling (Utrecht Univ.)
- $k = 4, 16, 64$ partitions
Communication Volume - Symmetric Matrices

- **cage10**
  - k=4
  - k=16
  - k=64

- **finan512**
  - k=4
  - k=16
  - k=64

- **bcsstk32**
  - k=4
  - k=16
  - k=64

- **bcsstk30**
  - k=4
  - k=16
  - k=64
Runtimes

**cage10**

- 1d Column
- Fine-grain
- nested diss.

**finan512**

- 1d Column
- Fine-grain
- nested diss.

**bcsstk32**

- 1d Column
- Fine-grain
- nested diss.

**bcsstk30**

- 1d Column
- Fine-grain
- nested diss.
Nonsymmetric Matrices

- **Given** nonsymmetric matrix $A$
- **Construct** bipartite graph $G'(R,C,E)$
  - $R$ vertices correspond to rows, $C$ vertices to columns
  - $E$ correspond to nonzeros
  - Can be represented by symmetric adjacency matrix

![Bipartite graph](image)

- **Apply nested dissection approach to** $G'$
  - Use same algorithm as for symmetric case

$$A' = \begin{bmatrix} 0 & A \\ A^T & 0 \end{bmatrix}$$
Communication Volume - Nonsymmetric Matrices

**Rectangular**
- cre\_b
- memplus

**Square**
- tbdlinux
- Ihr34

The graphs compare communication volume for different matrix types and sizes, with categories such as 1d*, Fine-grain, and nested diss.
Messages Sent (or Received) per Process

- **cage10**
  - `k=4`
  - `k=16`
  - `k=64`

- **finan512**
  - `k=4`
  - `k=16`
  - `k=64`

- **bcsstk32**
  - `k=4`
  - `k=16`
  - `k=64`

- **bcsstk30**
  - `k=4`
  - `k=16`
  - `k=64`
Summary of Nested Dissection Method Results

• New nested dissection 2-D algorithm
  - Implemented using existing algorithms and software
  - Quality better than 1-D, and similar to fine-grain hypergraph method for many matrices
  - Faster to compute than fine-grain hypergraph
  - Fewer messages than fine-grain hypergraph
Progress: Serial Matrix-Vector Multiplication (1)

- Improved hypergraph algorithm
  - Develop vertex ordering algorithm
Progress: Serial Matrix-Vector Multiplication (2)

- Hyperedge pruning
  - Develop one or two more heuristics
  - One based on MST graph solution
Progress: Serial Matrix-Vector Multiplication (2)

- Hyperedge detection
  - Need to improve $O(n^3)$ looping (for coplanar)
- Operation ordering
  - More cache friendly ordering
Progress: Sparse Matrix Partitioning (1)

- Corner reordering
  - Implement proposed method
- Nested dissection approach
  - Improve partitioning of separator vertices/edges
• **Nonsymmetric matrices**
  - *Corner method*
  - *Improve nested dissection approach to nonsymmetric*
Progress: Sparse Matrix Partitioning (2)

- Other communication metrics
  - Messages
- Numerical experiments
  - Larger matrices
Acknowledgements/Thanks

• **Professor Michael Heath,**  
  - Advisor

• **Dr. Erik Boman, Sandia National Laboratories**  
  - Summer technical advisor  
  - Collaborator on partitioning work  
  - Suggested serial matrix-vector optimization problem, hypergraphs, etc.

• **Dr. Bruce Hendrickson, Sandia**  
  - Corner method ordering  
  - Suggested vertex-ordering for serial opt. problem

• **Professor Robert Kirby, Texas Tech University**  
  - Serial matrix-vector optimization/FErari discussions
Acknowledgements/Thanks

- **Professor Luke Olson**
  - Help with FE code to generate matrices
- **Professor Jeff Erickson**
  - Discussion about serial optimization problem
    - Suggested vertex-ordering
    - Hyperedge detection
- **Professor William Gropp**
  - Discussion about serial optimization problem
- **Funding sources**
  - DOE CSGF (Krell Institute)
  - CSCAPES
  - Professor Heath’s Fulton Watson Copp Chair