Fixed-Parameter Tractability: Foundations, Implementations and Applications

Mike Langston

Faisal Abu-Khzam, Nicole Baldwin, Rebecca Collins, Mahesh Dorai, John Eblen, Lan Lin, Daniel Lucio, Jon Scharff, Xinxia Peng, Pushkar Shanbhag, Yongling Song, Henry Suters, Chris Symons, Ian Watkins
Salient Features

• Kernelization and branching methodology
• Parallelism based on search tree decomposition
• Fixed-parameter paradigms in optimization
The Vertex Cover Project

• Preprocessing plus kernelization

• Sequential branching

• Parallel branching
  – Search tree decomposition
  – Dynamic load balancing
Preprocessing

- Low degree rules (e.g., degree one)
- High degree rule
- Resultant graph has size at most $k(1+k/3)$
Kernelization

• Based on linear programming:
  
  – minimize: \( \sum_{u \in V(G)} X_u \)
  
  – subject to: \( X_u + X_v \geq 1 \forall uv \in E(G) \)
  
  – where: \( X_u \geq 0 \forall u \in V(G) \)

• Resultant graph has size at most \( 2k \)
Parallel Branching

• Focus on vertices of high degree
• Interleave preprocessing rules
• Based on a decomposition of the search tree
• Exploits the number of processors available
• Implemented using SSH or grid middleware
• Tasks run in a virtually unstructured manner
Intriguing Results on Synthetic Graphs of Size 600

<table>
<thead>
<tr>
<th>Graph name</th>
<th>Sequential Reduction</th>
<th>Sequential branching</th>
<th>Parallel branching</th>
</tr>
</thead>
<tbody>
<tr>
<td>RG30</td>
<td>1 second</td>
<td>Halted after two days</td>
<td>5 sec</td>
</tr>
<tr>
<td>RG31</td>
<td>1 second</td>
<td>Halted after two days</td>
<td>4 sec</td>
</tr>
<tr>
<td>RG32</td>
<td>1 second</td>
<td>Halted after two days</td>
<td>4 sec</td>
</tr>
</tbody>
</table>
Problems With the Parallel Method

- Extremely data dependent
- Unusually sensitive to rule order
- Super-super linear speedup on “yes” instances can equate to super-super linear slowdown on “no” instances
- Load balancing needed to achieve scalability

Solution: Dynamic Decomposition
Representative Results on Large Non-Synthetic Graphs

<table>
<thead>
<tr>
<th>Graph Name</th>
<th>Graph Size</th>
<th>Cover Size</th>
<th>Instance Type</th>
<th>Sequential Reduction</th>
<th>Sequential Branching</th>
<th>Parallel Branching</th>
<th>Dynamic Decomposition</th>
</tr>
</thead>
<tbody>
<tr>
<td>SH2-5</td>
<td>839</td>
<td>399</td>
<td>Yes</td>
<td>34 seconds</td>
<td>7 seconds</td>
<td>Not needed</td>
<td>Not needed</td>
</tr>
<tr>
<td>SH2-5</td>
<td>839</td>
<td>398</td>
<td>No</td>
<td>34 seconds</td>
<td>141 minutes</td>
<td>82 minutes</td>
<td>20 minutes</td>
</tr>
<tr>
<td>SH3-10</td>
<td>2466</td>
<td>2044</td>
<td>Yes</td>
<td>203 minutes</td>
<td>~ 5 days</td>
<td>~ 5 days</td>
<td>140 minutes</td>
</tr>
<tr>
<td>SH3-10</td>
<td>2466</td>
<td>2043</td>
<td>No</td>
<td>203 minutes</td>
<td>6+ days</td>
<td>6+ days</td>
<td>620 minutes</td>
</tr>
</tbody>
</table>
Sample Applications

- Dual to Clique

- Clique Useful in
  - Genomics (e.g., Phylogeny, Microarray Analysis)
  - Proteomics (e.g., SELDI)

- Codes Released in Clustal-XP
Applications to Phylogeny

A phylogenetic tree organizes life forms according to their putative evolutionary relationships.
Applications to SELDI

(a) Patient Samples → SELDI → Protein Peak Matrix → Pairwise Scoring and Mapping → Protein Correspondence Graph → Fast Graph Algorithms → Phenotype Identification Pattern

(b) Filtering
- Unweighted Graph → Graph Transform → Complement Graph → High Performance Computational Technologies → Vertex Cover
- Inverse Transform → Clique → Inverse Mapping
The Clique Compute Engine
Applications to Microarray Data Analysis

- DNA Microarrays
  - Raw Data
- Statistical Analysis
  - e.g., Normalization
  - Gene Expression Profiles
- Compute Pearson’s Coefficients
  - Edge-Weighted Graph
- Filter With Threshold Value
- Pre-Processing Tools
  - e.g., Graph Separators and Partitioning
- Clique-Centric Toolkit
  - Unweighted Graph
  - Cycles of Potential Interest
- Clique Extraction
  - e.g., Maximum Clique*, All Maximal Cliques*
  - *NP-complete
- Post-Processing Tools
  - e.g., Neighborhood Search and Components
- Validation