Pushing the Power of Stochastic Greedy Ordering Schemes for Inference in Graphical Models

Kalev Kask, Andrew Gelfand, Lars Otten, Rina Dechter
Dept. of Computer Science, UC Irvine

AAAI 2011 - San Francisco, CA
Tuesday, August 9th 2011
Graphical Model Inference

- Underlying graph structure encodes conditional independencies
  - Exploited in many inference algorithms:
    - Junction Tree (JT) [Lauritzen & Spiegelhalter 88]
      Bucket Elimination (BE) [Dechter 99]
    - Generalized BP [Yedidia, Freeman, & Weiss 05]
      AND/OR Sampling [Gogate & Dechter 08]
  - Complexity highly dependent on a given variable ordering and its (tree)width.
    - $O(k^w)$ – $k$ domain size, $w$ treewidth
Problem Decomposition

- Captured by elimination/variable ordering
  - Eliminate variable and connect neighbors, repeat

\[
\pi_A = x_1, x_2, x_3, x_4, x_5
\]

\[
C_1 = \{x_1, x_2, x_3, x_4, x_5\}
\]
\[
C_2 = \{x_2, x_3, x_4, x_5\}
\]
\[
C_3 = \{x_3, x_4, x_5\}
\]
\[
C_4 = \{x_4, x_5\}
\]
\[
C_5 = \{x_5\}
\]

\[
\text{width}(\pi_A) = \max_i |C_i| - 1 = 4
\]

\[
\pi_B = x_2, x_5, x_3, x_4, x_1
\]

\[
C_1 = \{x_1, x_2, x_4\}
\]
\[
C_2 = \{x_1, x_3, x_5\}
\]
\[
C_3 = \{x_3, x_1, x_4\}
\]
\[
C_4 = \{x_1, x_4\}
\]
\[
C_5 = \{x_1\}
\]

\[
\text{width}(\pi_B) = \max_i |C_i| - 1 = 2
\]
Computing “good” orderings

- Finding minimal order is NP-hard [Arnborg et al. 87]
  - Many anytime and approximate algorithms
    - **B&B:** [Gogate & Dechter 04] [Bachoore & Bodlaender 06]
    - **Tabu Search:** [Clautiaux et al. 04]
    - **Simulated Annealing:** [Kjaerulff 92]
- Greedy schemes are effective and popular
  - Not yet pushed to their limits
  - Preview: $n=15,319$, domain size $k=5$
  - \[ w = 36 \div 19 \text{ TB} \rightarrow w = 30 \div 41 \text{ GB} \]
Key Contributions

• Present comprehensive overview

• Develop unifying algorithm *IGVO*
  • Algorithmic enhancements:
    • Randomization through pooling
    • Early termination
    • Optimized data structures
    • Parallelization

• Perform extensive empirical evaluation
  • Obtain significant improvements
Greedy Variable Ordering

• **Algorithm:** GVO
  
  • For $i = 1$ to number of variables
    
    • $\pi(i) \leftarrow$ variable with smallest elimination cost
    
    • Eliminate $\pi(i)$

• Cost functions to consider:
  
  • *Min-Fill*: number of fill edges added
  
  • *Min-Degree*: degree of node in current graph
  
  • *Min-Complexity*: cost of variable elimination
Empirical Observation

- “Smallest cost” leads to many ties
  - Large variance in quality of resulting orders
- 20K Min-Fill iterations, random tie breaking:
Iterative GVO (IGVO)

• Break ties randomly and repeat! [Fishelson & Geiger 03]

• **Algorithm:** *Iterative GVO (IGVO)*
  
  - For $n=1$ to number of iterations
    - $\pi_n \leftarrow \text{GVO}(G)$ with random tie breaking
    - If $C(\pi_n, G) < C(\pi^*, G)$, then $\pi^* \leftarrow \pi_n$

• Possible complexity objectives:
  
  - **Width:** $C(\pi, G) \equiv \text{width}(\pi, G)$
  
  - **State space:** $C(\pi, G) \equiv s(\pi, G) = \sum_i s(\pi(i), G_i)$
Pooling & Early Termination

- **Pooling** with parameters $p$ and $e$:
  - Select node $\pi(i)$ from pool $T$ of size $p$
    - Can include nodes with non-minimal cost
  - Non-uniform sampling distribution over $T$:
    - Sample node $v$ with probability $p(v) = \frac{VC(v)^e}{\sum_{t \in T} VC(t)^e}$
    - Similar in [Fishelson & Geiger 03]

- **Early Termination**:
  - Abort iteration if cost of new ordering exceeds current optimum.
Optimized Data Structures

1) Adding fill edges has complexity $O(\text{deg}^3)$
   - Sorting adjacency lists reduces this to $O(2 \cdot \text{deg}^2)$

2) Updating Min-Fill costs when eliminating $x$
   - Full reevaluation of $N[x]$ and $N[N[x]]$ expensive
   - Instead, start from previous Min-Fill costs:
     - If $(w, u) \in E, (u, x) \in E$ and $(w, x) \notin E$ subtract 1 from $u$
     - $\forall$ fill-edges, $(u, v)$ if $(w, u) \in E$ and $(w, v) \notin E$ add 1 to $u$
     - $\forall$ fill-edges $(u, v)$ if $(w, u) \in E$ and $(w, v) \in E$ not added as fill-edge, subtract 1 from $w$
Experiments

- Large set of real-world benchmarks:
  - “largeFam”, 242 problems, haplotype queries
    - 2000-6000 variables, domain size 2-6
  - “type4”, 82 problems, genetic linkage analysis
    - Up to 15,000 variables, domain size 2-5
  - “protein”, 138 problems, side-chain prediction
    - Up to 2000 variables, max. domain size $k=81$
- Compare against baseline implementation
  - Standard Min-Fill with tie breaking
Comparing Ranking Functions

- Cumulative IGVO results (1 hour, largeFam)
  - 242 problems, 2000-6000 variables, $k = 6$
Effect of Randomization

- Comparing pool sizes (30 minutes, *largeFam*)
  - 242 problems, 2000-6000 variables, $k = 6$
Effect of Randomization

- Comparing pool sizes (30 minutes, largeFam)
  - 242 problems, 2000-6000 variables, $k = 6$
Effect of Parallelization

- Single- vs. 12-threaded (30 minutes, *type4*)
  - 82 problems, up to 15,000 variables, $k = 5$
## Effect of Parallelization

- Select results (30 minutes, *type4*)

<table>
<thead>
<tr>
<th>instance</th>
<th>$n$</th>
<th>iter</th>
<th>$w$</th>
<th>iter</th>
<th>$w$</th>
<th>iter</th>
<th>$w$</th>
<th>spd</th>
</tr>
</thead>
<tbody>
<tr>
<td>100-18</td>
<td>7,435</td>
<td>6,430</td>
<td>51</td>
<td>26,689</td>
<td>48</td>
<td>324,664</td>
<td>48</td>
<td>12.2</td>
</tr>
<tr>
<td>110-19</td>
<td>7,303</td>
<td>3,852</td>
<td>54</td>
<td>13,005</td>
<td>52</td>
<td>158,806</td>
<td>51</td>
<td>12.2</td>
</tr>
<tr>
<td>120-18</td>
<td>8,656</td>
<td>6,594</td>
<td>47</td>
<td>17,604</td>
<td>45</td>
<td>211,830</td>
<td>44</td>
<td>12.0</td>
</tr>
<tr>
<td>120-25</td>
<td>9,171</td>
<td>3,789</td>
<td>57</td>
<td>14,576</td>
<td>56</td>
<td>176,156</td>
<td>54</td>
<td>12.1</td>
</tr>
<tr>
<td>130-20</td>
<td>9,328</td>
<td>3,167</td>
<td>60</td>
<td>12,541</td>
<td>58</td>
<td>154,647</td>
<td>57</td>
<td>12.3</td>
</tr>
<tr>
<td>130-22</td>
<td>10,271</td>
<td>3,747</td>
<td>56</td>
<td>13,107</td>
<td>52</td>
<td>168,635</td>
<td>52</td>
<td>12.9</td>
</tr>
<tr>
<td>140-23</td>
<td>10,998</td>
<td>2,318</td>
<td>61</td>
<td>7,654</td>
<td>60</td>
<td>91,576</td>
<td>57</td>
<td>12.0</td>
</tr>
<tr>
<td>150-22</td>
<td>11,799</td>
<td>2,636</td>
<td>57</td>
<td>8,423</td>
<td>54</td>
<td>99,949</td>
<td>53</td>
<td>11.9</td>
</tr>
<tr>
<td>170-18</td>
<td>12,186</td>
<td>2,202</td>
<td>59</td>
<td>6,913</td>
<td>55</td>
<td>82,756</td>
<td>55</td>
<td>12.0</td>
</tr>
<tr>
<td>170-22</td>
<td>14,641</td>
<td>2,795</td>
<td>58</td>
<td>8,147</td>
<td>56</td>
<td>97,423</td>
<td>54</td>
<td>12.0</td>
</tr>
<tr>
<td>190-19</td>
<td>15,433</td>
<td>3,044</td>
<td>56</td>
<td>6,473</td>
<td>54</td>
<td>77,287</td>
<td>52</td>
<td>11.9</td>
</tr>
<tr>
<td>190-21</td>
<td>15,125</td>
<td>5,284</td>
<td>43</td>
<td>9,545</td>
<td>42</td>
<td>115,048</td>
<td>40</td>
<td>12.1</td>
</tr>
</tbody>
</table>
Pushing Feasibility

- **BEEM**: Bucket Elimination with External Memory [Kask, Gelfand, & Dechter 10]
  - Utilizes hard drive storage to store tables
  - Four previously infeasible instances now solvable

<table>
<thead>
<tr>
<th>instance</th>
<th>$n$</th>
<th>$k$</th>
<th>$w$</th>
<th>space</th>
<th>$w$</th>
<th>space</th>
</tr>
</thead>
<tbody>
<tr>
<td>110-21</td>
<td>7,675</td>
<td>5</td>
<td>37</td>
<td>16 TB</td>
<td>33</td>
<td>215 GB</td>
</tr>
<tr>
<td>140-20</td>
<td>9,355</td>
<td>5</td>
<td>35</td>
<td>10 TB</td>
<td>28</td>
<td>4 GB</td>
</tr>
<tr>
<td>180-21</td>
<td>14,157</td>
<td>5</td>
<td>38</td>
<td>9 TB</td>
<td>31</td>
<td>67 GB</td>
</tr>
<tr>
<td>200-18</td>
<td>15,319</td>
<td>5</td>
<td>36</td>
<td>19 TB</td>
<td>30</td>
<td>41 GB</td>
</tr>
</tbody>
</table>
Summary

- **Iterative Greedy Variable Ordering (IGVO):**
  - Unifying framework for finding orderings
  - Flexible yet simple and easily parallelizable
  - Implementation engineered for efficiency, algorithmic optimizations

- Often yields significantly better orderings
  - Allowed solving previously infeasible instances