Advances in Parallel Branch and Bound

Lars Otten and Rina Dechter
Dept. of Computer Science
University of California, Irvine
Summary

- Parallelizing AND/OR Branch and Bound:
  - Advanced optimization scheme: problem decomposition, subproblem caching, mini-bucket heuristic.

- Load Balancing is hard due to pruning:
  - Learn regression model for runtime prediction:
    - 34 subproblem features, static and dynamic.
    - Different levels of learning, up to 11K samples.

- Results: good estimation performance leads to improved load balancing:
  - High correlation coefficient of predictions.
  - Close to linear speedup for hard problems.
AND/OR Branch and Bound

- Search for combinatorial optimization over graphical models.

- Guided by *pseudo tree*:
  - Subproblem decomposition.
  - Merge unifiable subproblems.

- Mini-bucket heuristic:
  - Solve relaxed problem exactly.
  - $i$-bound control parameter.

- Asymptotic search complexity:
  - Exp. in treewidth, $O(n \cdot k^w)$.

[Marinescu & Dechter 2009]
[Kask & Dechter 2001]
AND/OR Branch and Bound

- Example AND/OR search space:

```
0 1 1 0 1 0 1 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
Parallelizing AOB

- Partially explore central search space.
  - Remaining subtrees yield parallel subproblems.
  - Implies parallelization frontier. [Grama & Kumar 1999]

8 independent subproblems with varying pruning
Parallel Performance Bottleneck

- Crucial: balance parallel workload.
  - Avoid few subproblems dominating everything.
  - Approach: Iteratively split hardest subproblem.

- Central question: Which is hardest?
  - Need to predict subproblem complexity in advance.

Algorithm 1 Pseudo code for subproblem generation

| Input: | Pseudo tree $T$ with root $X_0$, minimum subproblem count $p$, complexity estimator $\hat{N}$. |
| Output: | Set $F$ of subproblem root nodes with $|F| \geq p$. |

1: $F \leftarrow \{\langle X_0 \rangle\}$
2: while $|F| < p$
3: $n' \leftarrow \arg\max_{n \in F} \hat{N}(n)$
4: $F \leftarrow F \setminus \{n'\}$
5: $F \leftarrow F \cup \text{children}(n')$
Subproblem Complexity Regression

- Model number of nodes $N(n)$ as exponential function of subproblem features $\varphi_j(n)$:
  \[ N(n) = b \sum_j \lambda_j \varphi_j(n) \]

- Then consider log number of nodes:
  \[ \log N(n) = \sum_j \lambda_j \varphi_j(n) \]

- Thus, finding parameter values $\lambda_j$ can be seen as a *linear regression* problem.

  - Given sample subproblems $n_k$, minimize MSE:
    \[
    \frac{1}{m} \sum_{k=1}^{m} \left( \sum_j \lambda_j \varphi_j(n_k) - \log N(n_k) \right)
    \]

  related: [Leyton-Brown, Nudelman, Shoham 2009]
Subproblem Features $\varphi_j(n)$

- Use both static and dynamic characteristics:
  - Structural,
  - Subproblem bounds,
  - Limited AOBB probe.

**Subproblem variable statistics (static):**
1: Number of variables in subproblem.
2-6: Min, Max, mean, average, and std. dev. of variable domain sizes in subproblem.

**Pseudotree depth/leaf statistics (static):**
7: Depth of subproblem root in overall search space.
8-12: Min, max, mean, average, and std. dev. of depth of subproblem pseudo tree leaf nodes, counted from subproblem root.
13: Number of leaf nodes in subproblem pseudo tree.

**Pseudo tree width statistics (static):**
14-18: Min, max, mean, average, and std. dev. of induced width of variables within subproblem.
19-23: Min, max, mean, average, and std. dev. of induced width of variables within subproblem, when conditioning on subproblem root conditioning set.

**Subproblem cost bounds (dynamic):**
24: Lower bound $L$ on subproblem solution cost, derived from current best overall solution.
25: Upper bound $U$ on subproblem solution cost, provided by mini bucket heuristics.
26: Difference $U - L$ between upper and lower bound, expressing “constrainedness” of the subproblem.

**Pruning ratios (dynamic), based on running 5000 node expansion probe of AOBB:**
27: Ratio of nodes pruned using the heuristic.
28: Ratio of nodes pruned due to determinism (zero probabilities, e.g.)
29: Ratio of nodes corresponding to pseudo tree leaf.

**Sample statistics (dynamic), based on running 5000 node expansion probe of AOBB:**
30: Average depth of terminal search nodes within probe.
31: Average node depth within probe (denoted $\bar{d}$).
32: Average branching degree, defined as $\sqrt[3]{5000}$.

**Various (static):**
33: Mini bucket $i$-bound parameter.
34: Max. subproblem variable context size minus mini bucket $i$-bound.
Feature Informativeness

- *Lasso regularization* selects nine features:
  - $|\lambda_i|$ : Weight in learned model.
  - $\text{coo}$ : normalized cost of commission

| Feature $\varphi_i$                                      | $|\lambda_i|$ | $\text{coo}$ |
|---------------------------------------------------------|---------------|--------------|
| Average branching degree in probe                       | 0.57          | 100          |
| Average leaf node depth in probe                         | 0.39          | 87           |
| Subproblem upper bound minus lower bound                 | 0.22          | 17           |
| Ratio of nodes pruned by heuristic in probe              | 0.20          | 27           |
| Max. context size minus mini bucket i-bound              | 0.19          | 16           |
| Ratio of leaf nodes in probe                             | 0.18          | 10           |
| Subproblem upper bound                                  | 0.11          | 7            |
| Std. dev. of subproblem pseudo tree leaf depth           | 0.06          | 2            |
| Depth of subproblem root node in overall space          | 0.05          | 2            |

*relative error of model trained without $\varphi_i$*
Feature Illustration

- Example parallelization frontier (right):
  - Possible feature values below.

<table>
<thead>
<tr>
<th>Feature $\phi_i$</th>
<th>$\phi_i(B)$</th>
<th>$\phi_i(C)$</th>
<th>$\phi_i(E)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average branching degree in probe</td>
<td>1.2</td>
<td>1.3</td>
<td>1.2</td>
</tr>
<tr>
<td>Average leaf node depth in probe</td>
<td>2.2</td>
<td>1.7</td>
<td>1.8</td>
</tr>
<tr>
<td>Subproblem upper bound minus lower bound</td>
<td>15.6</td>
<td>11.2</td>
<td>12.7</td>
</tr>
<tr>
<td>Ratio of nodes pruned by heuristic in probe</td>
<td>0.7</td>
<td>0.6</td>
<td>0.5</td>
</tr>
<tr>
<td>Max. context size minus mini bucket i-bound</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Ratio of leaf nodes in probe</td>
<td>0.2</td>
<td>0.3</td>
<td>0.1</td>
</tr>
<tr>
<td>Subproblem upper bound</td>
<td>26.2</td>
<td>28.3</td>
<td>22.5</td>
</tr>
<tr>
<td>Std. dev. of subproblem pseudo tree leaf depth</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Depth of subproblem root node in overall space</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Training the Models

- Experiments on 31 problems from 4 classes:
  - $n$: number of variables, $k$: max. domain size,
    $w$: induced width, $h$: pseudo tree height.

- About 11,500 subproblem training samples:
  - Run each instance with fixed-depth cutoff, use max. 500 subproblems.

<table>
<thead>
<tr>
<th>domain</th>
<th>#</th>
<th>$n$</th>
<th>$k$</th>
<th>$w$</th>
<th>$h$</th>
</tr>
</thead>
<tbody>
<tr>
<td>pedigree</td>
<td>13</td>
<td>137-1212</td>
<td>3-7</td>
<td>17-39</td>
<td>47-102</td>
</tr>
<tr>
<td>pdb</td>
<td>5</td>
<td>103-172</td>
<td>8</td>
<td>10-15</td>
<td>24-43</td>
</tr>
<tr>
<td>largeFam</td>
<td>8</td>
<td>2569-3730</td>
<td>3-4</td>
<td>28-37</td>
<td>73-108</td>
</tr>
<tr>
<td>grid</td>
<td>5</td>
<td>624-675</td>
<td>2</td>
<td>37-39</td>
<td>111-124</td>
</tr>
</tbody>
</table>
Evaluating Prediction Performance

- Incrementally more general levels of learning:
  - Sample subproblems from one instance only:
    - Need to learn new model for each instance.
  - Sample subproblem from problems from one class:
    - One model sufficient for one entire problem class.
  - Sample subproblems across all classes:
    - One model applies to all problems.
  - (Future work: Prediction for unseen classes?)

- Record prediction error ($MSE$), training error ($TER$), and correlation coefficient ($PCC$).
Metrics / Terminology

- **MSE**: Prediction Error
  - Model error on the training sample set.
  - Relates to generalization error.

- **TER**: Training Error / Sample Error
  - Model error on the test sample set.

- **PCC**: Pearson Correlation Coefficient
  - Covariance between actual and predicted complexities, normalized by product of respective standard deviation.
Learning per Instance (5-fold CV)

- **pedigree19**, p=1440, fixed d=6
  - MSE: 0.469
  - PCC: 0.835
  - TER: 0.468

- **pedigree41**, p=1408, fixed d=10
  - MSE: 0.212
  - PCC: 0.842
  - TER: 0.212

- **largeFam3-15-59**, p=936, fixed d=9
  - MSE: 0.162
  - PCC: 0.937
  - TER: 0.159

- **largeFam3-11-59**, p=200, fixed d=8
  - MSE: 0.071
  - PCC: 0.879
  - TER: 0.068

- **pdb1a6m**, p=511, fixed d=3
  - MSE: 0.061
  - PCC: 0.990
  - TER: 0.059

- **75-26-9**, p=240, fixed d=9
  - MSE: 0.129
  - PCC: 0.801
  - TER: 0.127
Learning per Problem Class

pedigree19, p=1440, fixed d=6

- MSE: 1.384
- PCC: 0.835
- TER: 0.585

pedigree41, p=1408, fixed d=10

- MSE: 0.590
- PCC: 0.322
- TER: 0.656

largeFam3-15-59, p=936, fixed d=9

- MSE: 0.164
- PCC: 0.968
- TER: 0.160

largeFam3-11-59, p=200, fixed d=8

- MSE: 0.150
- PCC: 0.964
- TER: 0.160

75-26-9, p=240, fixed d=9

- MSE: 0.121
- PCC: 0.815
- TER: 0.347
Learning across Problem Classes

- **pedigree19** (p=1440, fixed d=6)
  - MSE: 0.656
  - PCC: 0.885
  - TER: 0.651

- **pedigree41** (p=1408, fixed d=10)
  - MSE: 0.527
  - PCC: 0.819
  - TER: 0.635

- **largeFam3-15-59** (p=936, fixed d=9)
  - MSE: 0.370
  - PCC: 0.966
  - TER: 0.648

- **largeFam3-11-59** (p=200, fixed d=8)
  - MSE: 0.094
  - PCC: 0.957
  - TER: 0.646

- **pdb1a6m** (p=511, fixed d=3)
  - MSE: 0.422
  - PCC: 0.989
  - TER: 0.646

- **75-26-9** (p=240, fixed d=9)
  - MSE: 0.228
  - PCC: 0.825
  - TER: 0.643
Prediction Performance Summary

- No indication of overfitting:
  - Prediction error is fairly close to training error.
- Different levels of learning:
  - Per instance:
    - Limited practical relevance, requires extensive sampling.
  - Per problem class / across classes:
    - Allows reuse of learned models, useful in practice.
- Promising generalization performance:
  - Little increase in error across learning levels.
  - Very good correlation coefficients.
Improved Load Balancing

- Compare against naive, fixed-depth cutoff (left):

---

**pedigree19, 100 CPUs, p=144, fixed d=4**

<table>
<thead>
<tr>
<th>Overall</th>
<th>Max. Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>34424 sec</td>
<td>34382</td>
</tr>
</tbody>
</table>

Solution time [sec]

- Subproblem runtimes
- Overall runtime

- Med: 1132
- Ave: 3382.1
- Stdv: 5584.3

---

**pedigree19, 100 CPUs, p=144, regr.-based**

<table>
<thead>
<tr>
<th>Overall</th>
<th>Max. Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>19814 sec</td>
<td>19772</td>
</tr>
</tbody>
</table>

Solution time [sec]

- Subproblem runtimes
- Overall runtime

- Med: 2124
- Ave: 3376.5
- Stdv: 4059.9

---

**pedigree41, 50 CPUs, p=64, fixed d=5**

<table>
<thead>
<tr>
<th>Overall</th>
<th>Max. Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>4706 sec</td>
<td>4704</td>
</tr>
</tbody>
</table>

Solution time [sec]

- Subproblem runtimes
- Overall runtime

- Med: 267
- Ave: 793.3
- Stdv: 1339.6

---

**pedigree41, 50 CPUs, p=64, regr.-based**

<table>
<thead>
<tr>
<th>Overall</th>
<th>Max. Job</th>
</tr>
</thead>
<tbody>
<tr>
<td>2667 sec</td>
<td>2665</td>
</tr>
</tbody>
</table>

Solution time [sec]

- Subproblem runtimes
- Overall runtime

- Med: 644
- Ave: 798.3
- Stdv: 544.8
Parallel Runtime Summary

- Example: pedigree benchmarks
  - Sequential AOBB vs. parallel scheme w/ regression.
  - Harder instances profit the most.

<table>
<thead>
<tr>
<th>inst</th>
<th>n</th>
<th>k</th>
<th>w</th>
<th>h</th>
<th>seq</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
</tr>
</thead>
<tbody>
<tr>
<td>ped7</td>
<td>1068</td>
<td>4</td>
<td>32</td>
<td>90</td>
<td>26:11</td>
<td>02:49</td>
<td>01:29</td>
<td>00:39</td>
<td>00:21</td>
<td>00:12</td>
<td>00:09</td>
<td>00:09</td>
</tr>
<tr>
<td>ped9</td>
<td>1118</td>
<td>7</td>
<td>27</td>
<td>100</td>
<td>16:26</td>
<td>01:57</td>
<td>00:59</td>
<td>00:24</td>
<td>00:13</td>
<td>00:07</td>
<td>00:06</td>
<td>00:05</td>
</tr>
<tr>
<td>ped13</td>
<td>1077</td>
<td>3</td>
<td>32</td>
<td>102</td>
<td>28:42</td>
<td>02:51</td>
<td>01:28</td>
<td>00:42</td>
<td>00:24</td>
<td>00:16</td>
<td>00:13</td>
<td>00:13</td>
</tr>
<tr>
<td>ped19</td>
<td>793</td>
<td>5</td>
<td>25</td>
<td>98</td>
<td>105:11</td>
<td>13:48</td>
<td>07:38</td>
<td>03:17</td>
<td>01:56</td>
<td>01:14</td>
<td>00:50</td>
<td>00:42</td>
</tr>
<tr>
<td>ped31</td>
<td>1183</td>
<td>5</td>
<td>30</td>
<td>85</td>
<td>121:25</td>
<td>12:43</td>
<td>06:38</td>
<td>02:43</td>
<td>01:23</td>
<td>00:43</td>
<td>00:31</td>
<td>00:24</td>
</tr>
<tr>
<td>ped34</td>
<td>1160</td>
<td>5</td>
<td>31</td>
<td>102</td>
<td>12:34</td>
<td>02:05</td>
<td>00:54</td>
<td>00:24</td>
<td>00:13</td>
<td>00:08</td>
<td>00:06</td>
<td>00:05</td>
</tr>
<tr>
<td>ped41</td>
<td>1062</td>
<td>5</td>
<td>33</td>
<td>100</td>
<td>13:07</td>
<td>01:34</td>
<td>00:48</td>
<td>00:23</td>
<td>00:16</td>
<td>00:10</td>
<td>00:11</td>
<td>00:11</td>
</tr>
<tr>
<td>ped44</td>
<td>811</td>
<td>4</td>
<td>25</td>
<td>65</td>
<td>26:52</td>
<td>03:28</td>
<td>01:58</td>
<td>00:54</td>
<td>00:32</td>
<td>00:18</td>
<td>00:13</td>
<td>00:11</td>
</tr>
<tr>
<td>ped51</td>
<td>1152</td>
<td>5</td>
<td>39</td>
<td>98</td>
<td>46:13</td>
<td>04:54</td>
<td>02:31</td>
<td>01:06</td>
<td>00:36</td>
<td>00:22</td>
<td>00:21</td>
<td>00:19</td>
</tr>
</tbody>
</table>
Parallel Speedups

- Speedup vs. sequential algorithm (1 CPU)
Summary

- Parallelizing AND/OR Branch and Bound:
  - Advanced optimization scheme: problem decomposition, subproblem caching, mini-bucket heuristic.
- Load Balancing is hard due to pruning.
  - Learn regression model for runtime prediction:
    - 34 subproblem features, static and dynamic.
    - Different levels of learning, up to 11K samples.
- Results: good estimation performance leads to improved load balancing.
  - High correlation coefficient of predictions.
  - Close to linear speedup for hard problems.