Using Regression for Parallelizing AOBB

Lars Otten
Recap

• Static parallelization:
  • Fix number of subproblems.
    - Iteratively push frontier deeper.
  • Estimate complexity for each subproblem
    - Split subproblem with highest estimate.
• Previous metric:
  - \( N = b ^ { (U-L)^{\sqrt{h / \text{inc}}} } \)
New Model

- Use linear combination in exponent:
  - $N = \exp(\sum \lambda_i x_i)$

- To apply linear regression apply log:
  - $\log N = \sum \lambda_i x_i$

- Minimize error, given set of subproblems:

\[
MSE = \frac{1}{m} \sum_{k=1}^{m} \left( \sum_j \lambda_j x_j^{(k)} - N^{(k)} \right)^2
\]
Features

- $1$, constant value.
- $U$, upper bound on the subproblem solution, provided by the mini bucket heuristic,
- $L$, lower bound (for pruning) on the subproblem solution, calculated from the subproblem's partial solution tree and the current best-known solution,
- $U-L$, the “gap” between upper and lower bound.
- $h$, the height of the subproblem pseudo tree,
- $d$, the depth of the subproblem root node in the overall problem,
- $w$, the induced width of the subproblem,
- $n$, the number of subproblem variables,
- $\log(n)$, the log of the number of variables,
- $(U-L)*h$, as before.
Feature Selection

• Forward selection:
  • Start with just constant feature, compute RSS\(_0\).
  • Iterate:
    – For each unused feature, add to current feature set and compute updated RSS.
    – Pick feature that improves RSS the most (with threshold weighing number of features vs. sample size)

• Alternate with backward elimination:
  • Same principle, but removing the one feature that does not contribute enough.
Selected Results

- ped31, regression on all 320 subproblems

MSE: 0.0266491677243
Selected Results

- ped31, regression on 50/12800 subproblems

MSE: 0.107561693368
Selected Results

- ped13, all 640 subproblems

pedigree13_110408-130458_i20-w12-t0-p640

MSE: 0.80902036317
Selected Results

- ped19, all 640 subproblems
Runs with Regression

• Eventual goal:
  • Sample subproblems during preprocessing, select features and learn weights.

• For now:
  • Try different subproblem samples across range of runs off-line.
  • Perform “educated average”:
    - $2 \times (U-L) + h + 0.5 \times \log(n)$
Select Results

pedigree31_110506-100637_i18-w10-t0-p640

Log10 number of nodes per subproblem

Subproblems by increasing 'met'

MSE: 0.0395247182385
Select Results

pedigree19_110505-183523_i16-w10-t0-p640

Log10 number of nodes per subproblem

Subproblems by increasing 'met'

MSE: 0.434819374279
Solution Times

- Parallel performance
  - “old” vs. “new” metric.
  - ~320 CPUs

<table>
<thead>
<tr>
<th>instance</th>
<th>“old”</th>
<th>“new”</th>
</tr>
</thead>
<tbody>
<tr>
<td>ped13</td>
<td>144</td>
<td>59</td>
</tr>
<tr>
<td>ped19</td>
<td>231,454</td>
<td>33,755</td>
</tr>
<tr>
<td>ped31</td>
<td>1,631</td>
<td>519</td>
</tr>
<tr>
<td>ped51</td>
<td>22,306</td>
<td>5,198</td>
</tr>
</tbody>
</table>
Parallel Granularity

- Fix #CPUs, vary #subproblems
  - ped19, ~320 CPUs

<table>
<thead>
<tr>
<th>#subprob</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1280</td>
<td>11,867</td>
</tr>
<tr>
<td>2560</td>
<td>10,704</td>
</tr>
<tr>
<td>5120</td>
<td>7,506</td>
</tr>
</tbody>
</table>
Scaling

- Fix subproblem set, vary #CPUs
  - ped19, 2560 subproblems
    - | CPUs | time   |
    - |------|--------|
    - | 288  | 10,829 |
    - | 144  | 12,257 |
    - | 72   | 18,603 |
    - | 36   | 30,879 |
    - x 1.31, x 1.52, x 1.66

- ped31, 640 subproblems
  - | CPUs | time   |
  - |------|--------|
  - | 288  | 463    |
  - | 144  | 758    |
  - | 72   | 1,334  |
  - | 36   | 2,496  |
  - x 1.64, x 1.76, x 1.87