Learning to Search in Branch-and-Bound Algorithms

He He, Hal Daumé III  
University of Maryland, College Park

Jason Eisner  
Johns Hopkins University

Overview

What is the best strategy to traverse the search tree in branch and bound?

- **Best** means to find a near-optimal solution as early as possible
- Different types of problems require different search strategies
- A single strategy usually does not work well throughout the search tree
- Our solution: automatically learns searching strategies that are adapted to a family of problems and different solving stages within one problem

Toy example: knapsack problem formulated as integer linear programming (ILP)

- Smart node selection/pruning speed up the solving process
- Good decisions come from experience — imitation learning

<table>
<thead>
<tr>
<th>Assumptions:</th>
</tr>
</thead>
<tbody>
<tr>
<td>- A small set of solved problems are given at training time</td>
</tr>
<tr>
<td>- Problems to be solved at test time are of the same type</td>
</tr>
<tr>
<td>- Finding a good feasible solution is enough — no need for proof of optimality</td>
</tr>
</tbody>
</table>

Node selection policy:

- A node comparator
- Node score = weight vector \times feature vector
- Higher score means the subtree is more likely to contain the optimal solution
- Always pop the node with the highest score

Both policies are learned by imitation learning (Dataset Aggregation)

Oracle:

- Expand optimal nodes first
- Prune all non-optimal nodes
- Provide training labels

Training examples:

\[ \omega \text{select} \cdot (\varphi(\omega) - \varphi(\omega)) = 0 \]
\[ \omega \text{prune} \cdot \psi(\omega) = 1 \]

Policy features (dynamic):

- Node: lower bound, estimated objective, depth, is child/sibling
- Branching: pseudocost, difference between current LP solution and root LP solution/current bound
- Tree: global bounds, integrality gap, number of solution found

Experiments

- Four Mixed-ILP libraries (# of vars: 300 - 1000; # of constrs: 100 - 500)
- Solver implemented based on SCIP
- Speedup with respect to SCIP

<table>
<thead>
<tr>
<th></th>
<th>P+S</th>
<th>P</th>
<th>SCIP</th>
<th>Gurobi</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regions</td>
<td>MIK</td>
<td>0.04‰</td>
<td>0.04‰</td>
<td>3.02‰</td>
</tr>
<tr>
<td>MIK</td>
<td>0.79‰</td>
<td>6.80‰</td>
<td>21.94‰</td>
<td></td>
</tr>
<tr>
<td>Hybrid</td>
<td>0.00‰</td>
<td>0.00‰</td>
<td>0.79‰</td>
<td>3.97‰</td>
</tr>
<tr>
<td>CORLAT</td>
<td>8.99%</td>
<td>8.91%</td>
<td>fail</td>
<td>fail</td>
</tr>
</tbody>
</table>

Optimality gap compared with SCIP (early stop at the same end time) and Gurobi (early stop at the same # of nodes explored)

- SCIP and Gurobi in their default settings work well on some datasets but not all; while our policy learns to adapt to specific problems
- Cross generalization — apply policies learned on one dataset to another