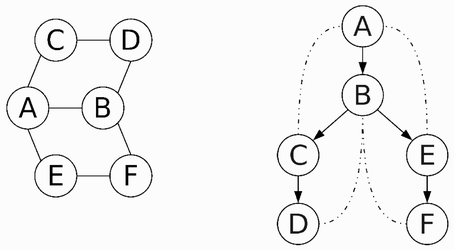
A Case Study in Complexity Estimation: Towards Parallel Branch-and-Bound over Graphical Models

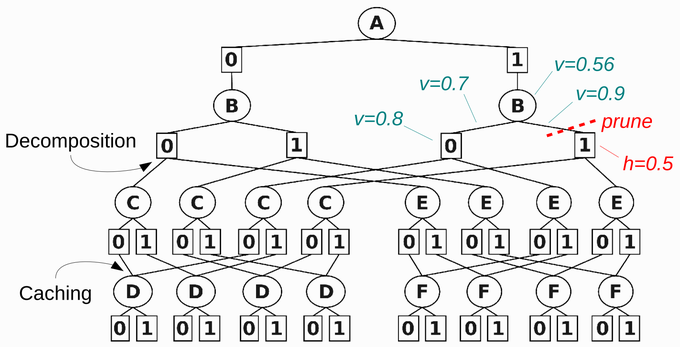
Lars Otten and Rina Dechter – *{lotten,dechter}@ics.uci.edu*
Dept. of Computer Science, University of California, Irvine

Abstract. We study the problem of complexity estimation in the context of parallelizing an advanced Branch and Bound-type algorithm over graphical models. The algorithm's pruning power makes load balancing, one crucial element of every distributed system, very challenging. We propose using a statistical regression model to identify and tackle disproportionally complex parallel subproblems, the cause of load imbalance, ahead of time. The proposed model is evaluated and analyzed on various levels and shown to yield robust predictions. We then demonstrate its effectiveness for load balancing in practice.

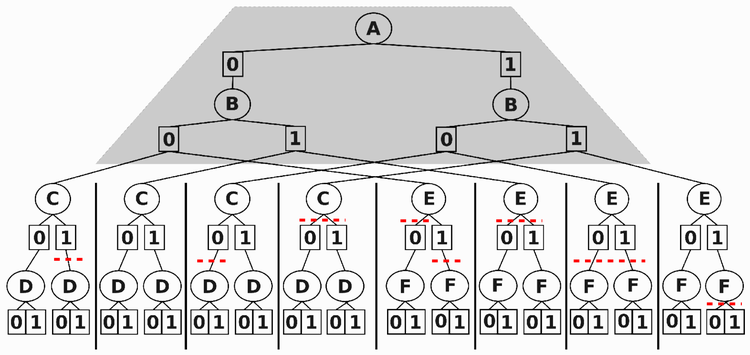
- AND/OR Branch and Bound search for optimization:
  - State-of-the-art combinatorial optimization algorithm.
  - PASCAL 2011 Inference Challenge: 1st in all MPE tracks.



- Guided by *pseudo tree*, captures conditional independencies in underlying graphical model.
  - Problem decomposition through *AND nodes*; *caching* avoids redundant computations using extra memory.
  - *Mini-bucket heuristic* solves relaxed problem to yield upper bounds on subproblem solutions.
    - Trade off accuracy and complexity via *i*-bound parameter.



- Parallelize AOBB using parallel tree search concept:
  - Enforce *parallelization frontier*, subproblems solved concurrently by independent worker nodes.
  - **Load Balancing** is crucial, but hard because of pruning.



- **Load balancing through learning:**
  - Subproblem runtime estimation via *regression model*.

Algorithm: Computing the parallelization frontier
Input: Pseudo tree T with root X_0, subproblem count p, complexity estimator N.
Output: Set F of subproblems root nodes with $|F| \geq p$.
1: $F \leftarrow \{ <X_0> \}$
2: **while** $|F| < p$ **:**
3: $n' \leftarrow \text{argmax}_{n \in F} N(n)$
4: $F \leftarrow F \setminus \{n'\}$
5: F $\leftarrow F \cup$ children(n')

- Experiments on 31 problems from 4 domains.
  - Run each with fixed-depth cutoff, use max. 500 subproblems.
  - About 11,500 subproblem samples overall.

domain	#	*n*	*k*	*w*	*h*
pedigree	13	137-1212	3-7	17-39	47-102
pdb	5	103-172	8	10-15	24-43
largeFam	8	2569-3730	3-4	28-37	73-108
grid	5	624-675	2	37-39	111-124

- Given subproblem n, compute **34** subproblem features:
  - Structural / cost-bound-based / from small AOBB probe.
  - Learn linear regression model for log complexity:

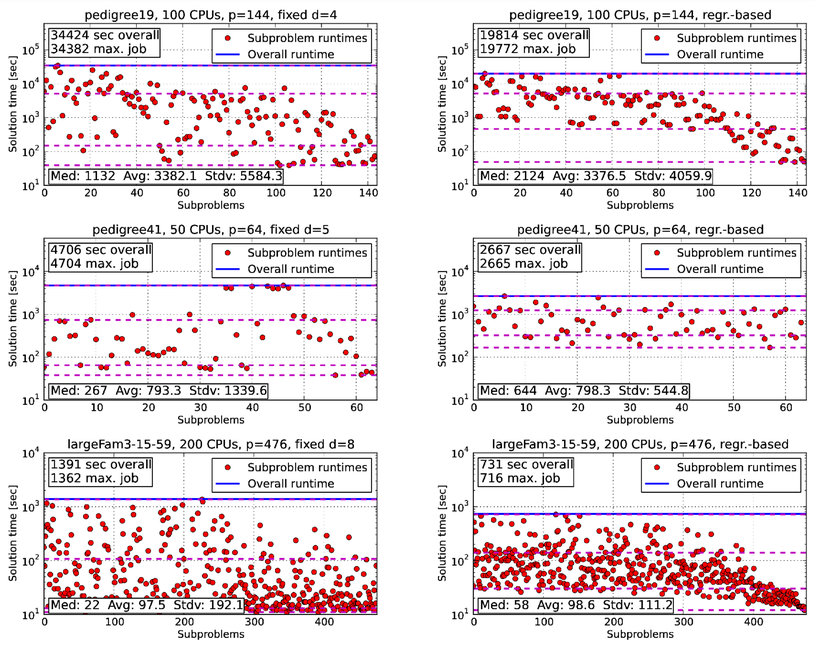
$$\log N(n) = \sum \lambda_i \cdot \varphi_i(n)$$

  - Nine features selected using *Lasso regularization*:

Feature φ_i	$\lvert\lambda_i\rvert$	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

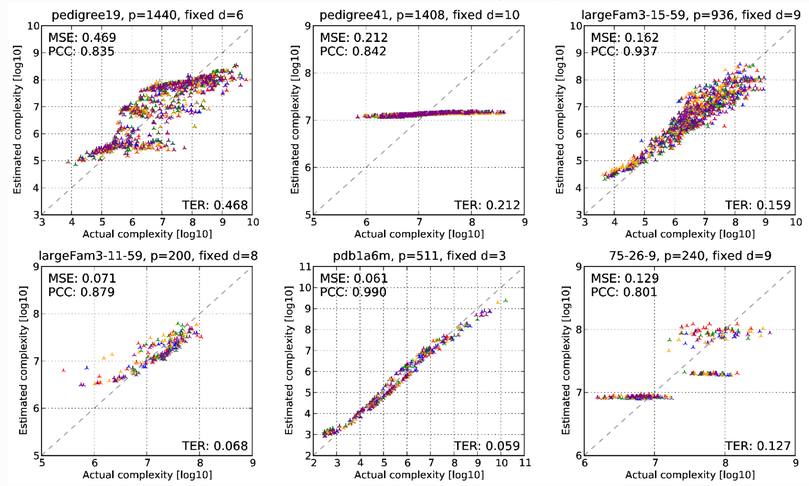
  - Experiments with higher-order models showed similar results.

- Load balancing before (left, fixed depth) and after (right):

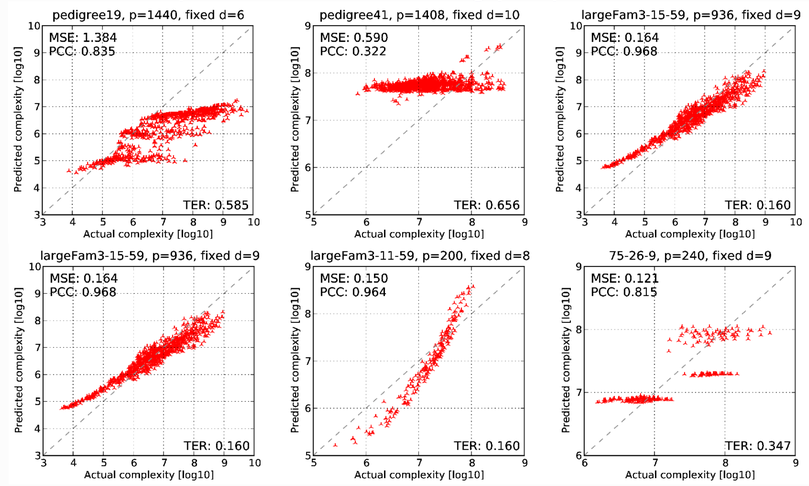


- Three incrementally more general levels of learning:
  - Sample subproblems from one instance only:
    - Need to learn new model for each instance.
  - Sample subproblem from problems from one class:
    - One model sufficient for one entire problem class.
  - Sample subproblems from problems across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

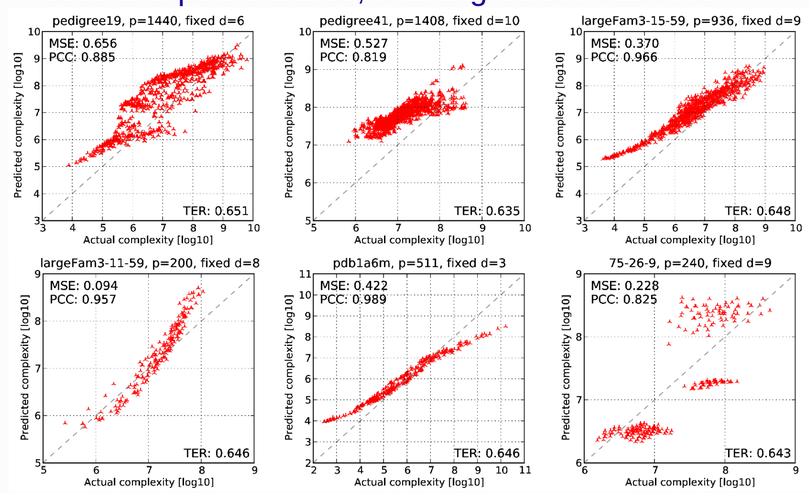
- Prediction performance, learning per problem instance:



- Prediction performance, learning per problem class:



- Prediction performance, learning across classes:



- Facilitate parallel speedup:
  - Before (dashed)
  - After (solid)
- Investigate parallel granularity next.