ECAI 2012

Advances in Parallel Branch and Bound

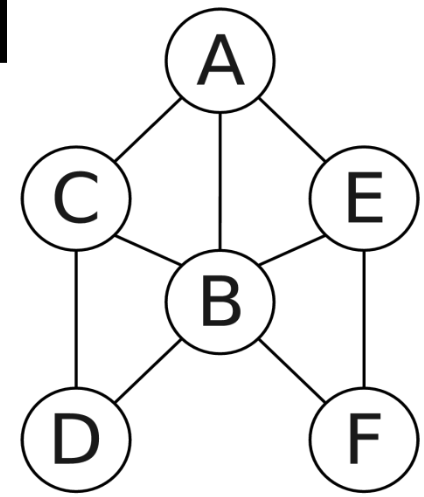
Lars Otten and Rina Dechter
Dept. of Computer Science
University of California, Irvine

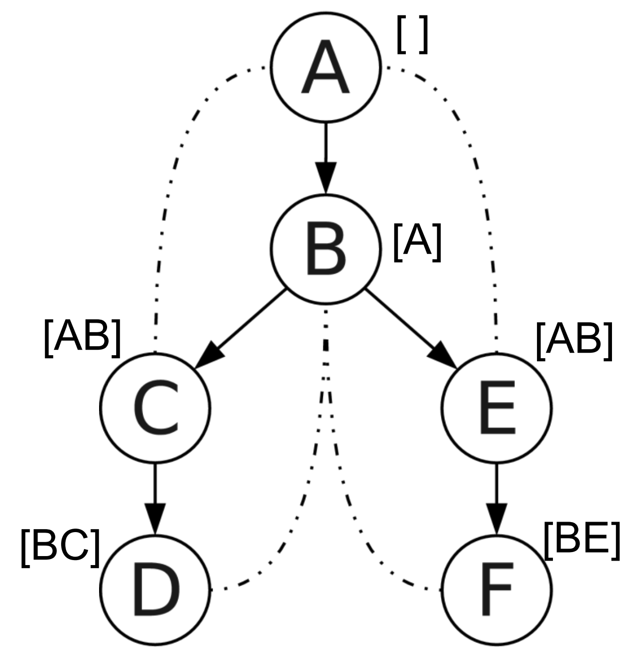
Summary

- 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, $O(n \cdot k^w)$.

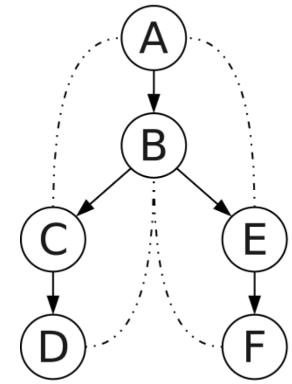


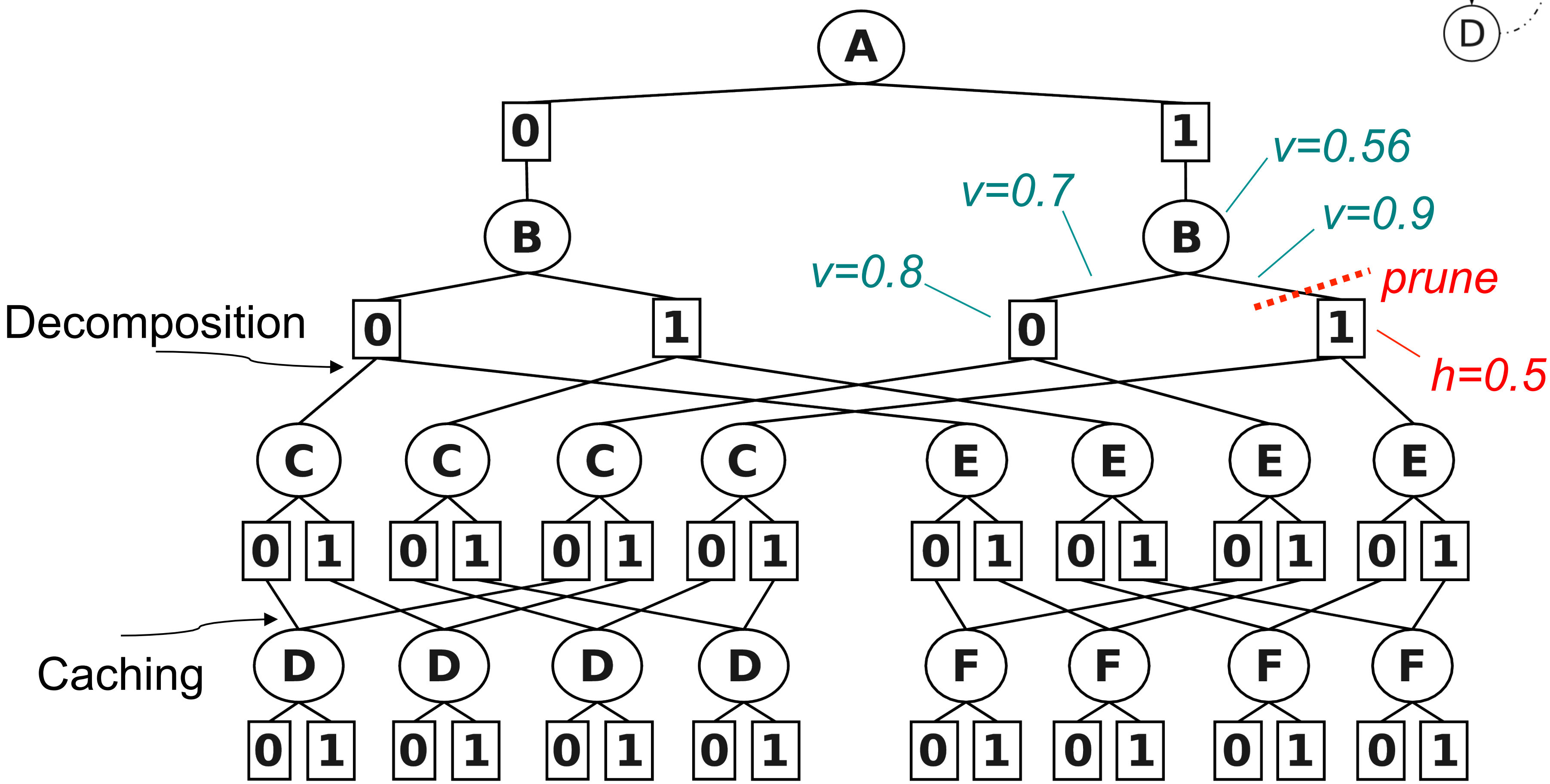


[Marinescu & Dechter 2009]
[Kask & Dechter 2001]

AND/OR Branch and Bound

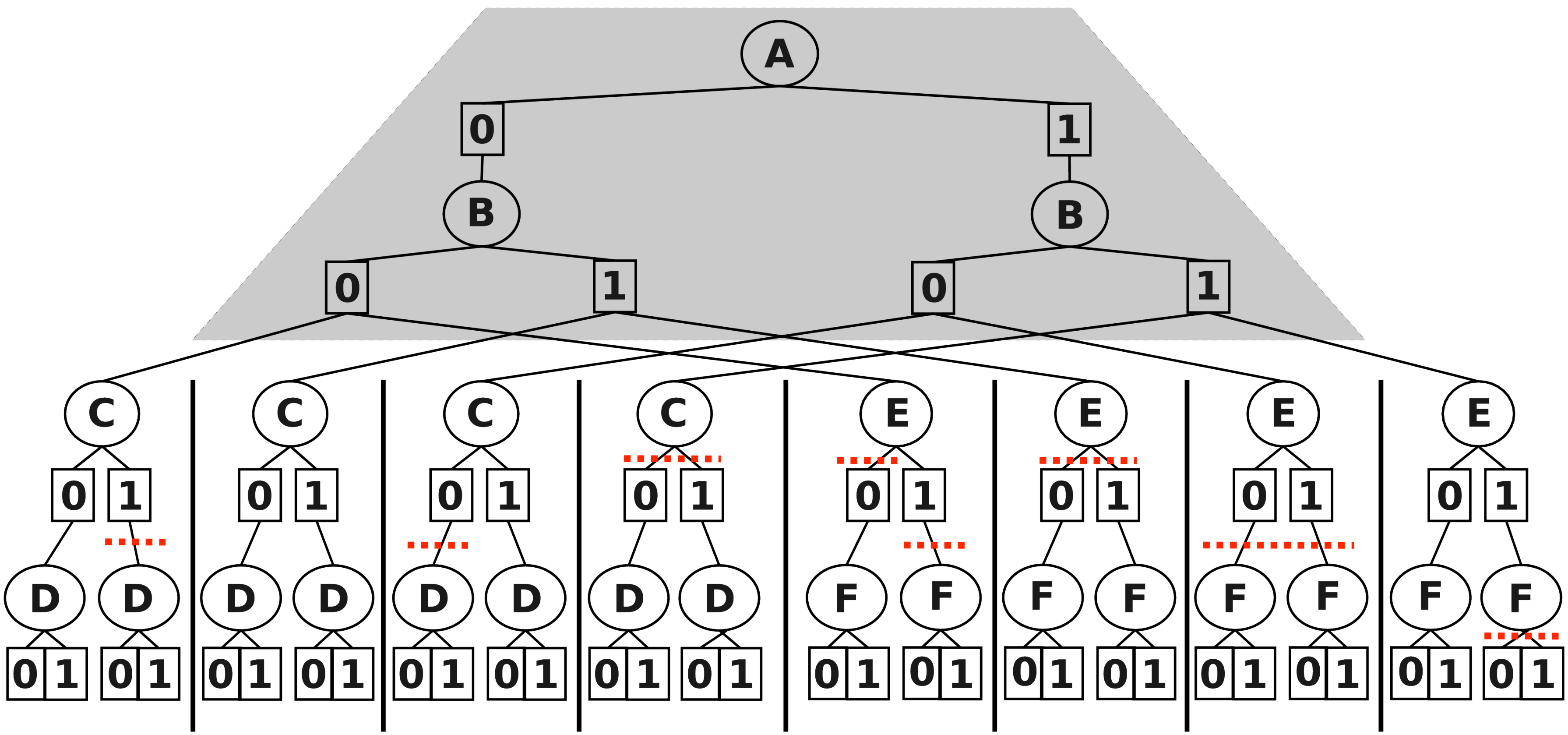
- Example AND/OR search space:





Parallelizing AOBB

- 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 $\mathcal{T}$ 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: $\quad n' \leftarrow \arg\max_{n \in F} \hat{N}(n)$
4: $\quad F \leftarrow F \setminus \{n'\}$
5: $\quad F \leftarrow F \cup children(n')$

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 λ_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 of 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 $\sqrt[\bar{d}]{5000}$.

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

Feature Informativeness

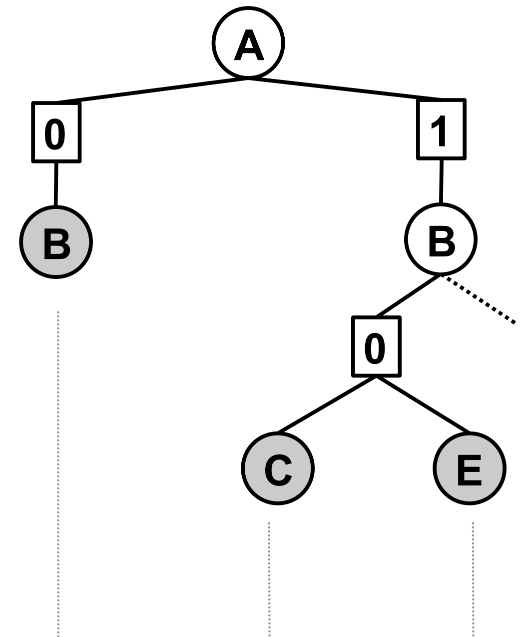
- *Lasso regularization* selects nine features:
  - $|\lambda_i|$: Weight in learned model.
  - *coo* : normalized cost of commission

relative error of model trained without φ_i

Feature φ_i	$|\lambda_i|$	***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

Feature Illustration

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



Feature φ_i	φ_i(B)	φ_i(C)	φ_i(E)
Average branching degree in probe	1.2	1.3	1.2
Average leaf node depth in probe	2.2	1.7	1.8
Subproblem upper bound minus lower bound	15.6	11.2	12.7
Ratio of nodes pruned by heuristic in probe	0.7	0.6	0.5
Max. context size minus mini bucket i-bound	1	1	1
Ratio of leaf nodes in probe	0.2	0.3	0.1
Subproblem upper bound	26.2	28.3	22.5
Std. dev. of subproblem pseudo tree leaf depth	0	0	0
Depth of subproblem root node in overall space	1	2	2

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.

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

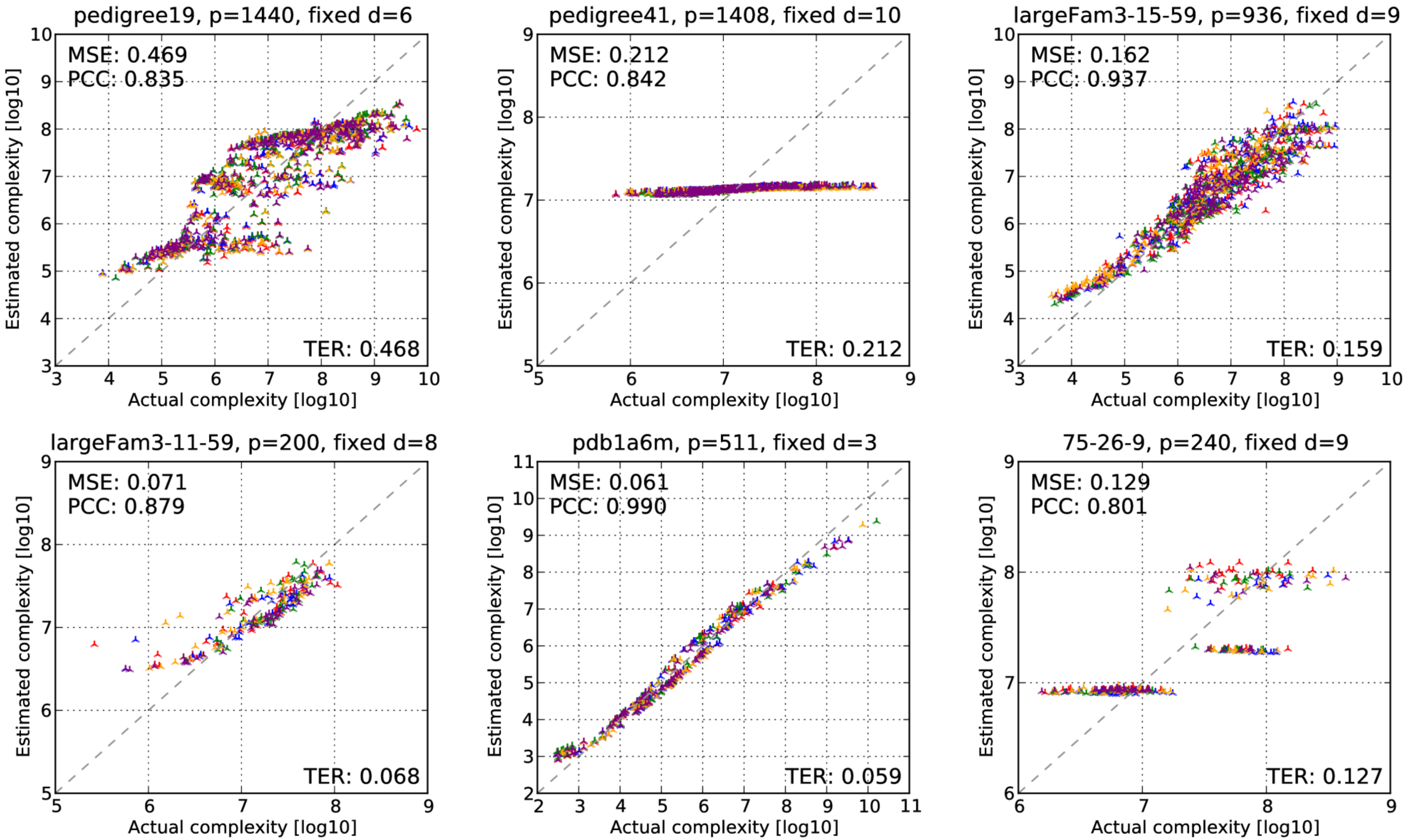
Evaluating Prediction Performance

- 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*).

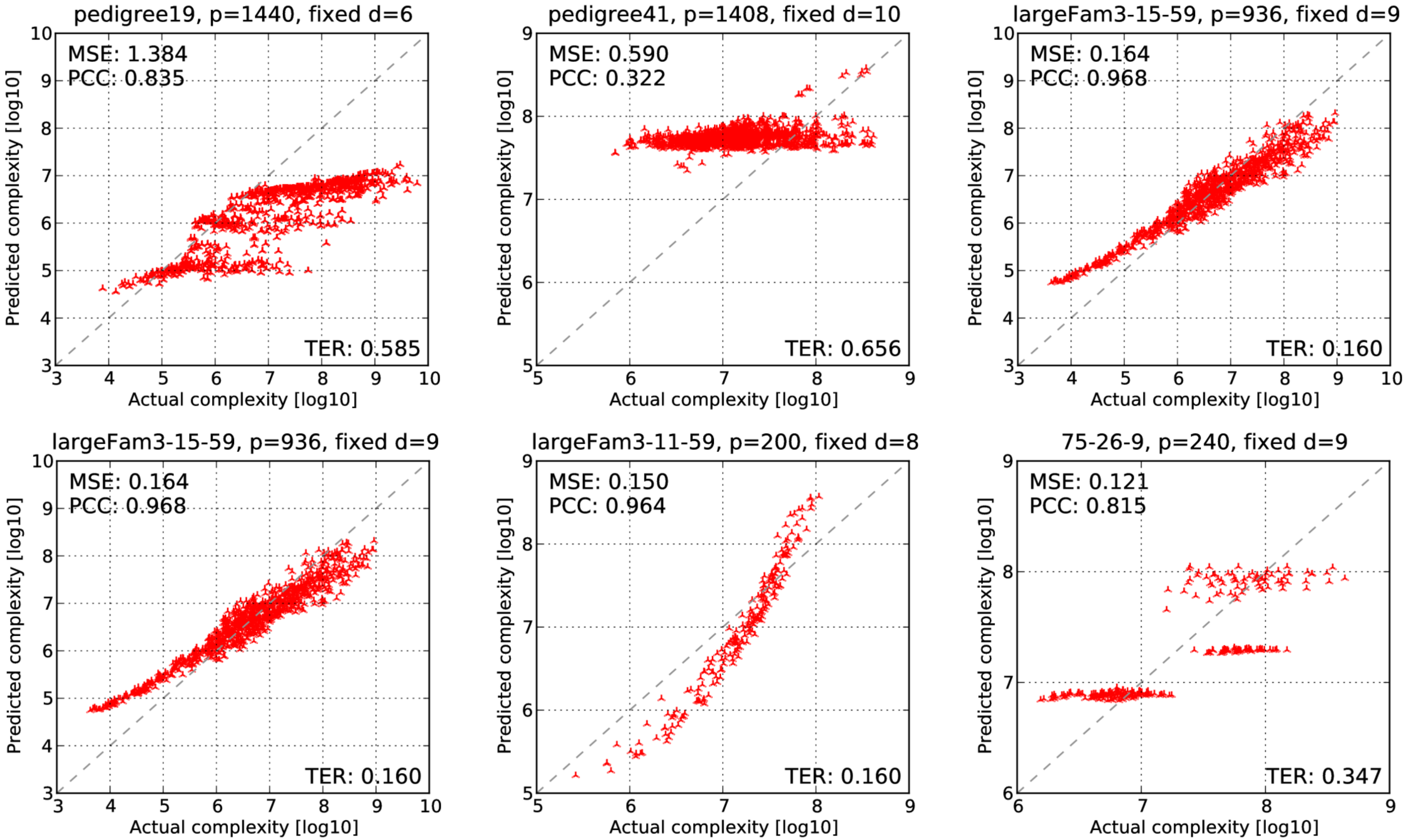
Metrics / Terminology

- ***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.

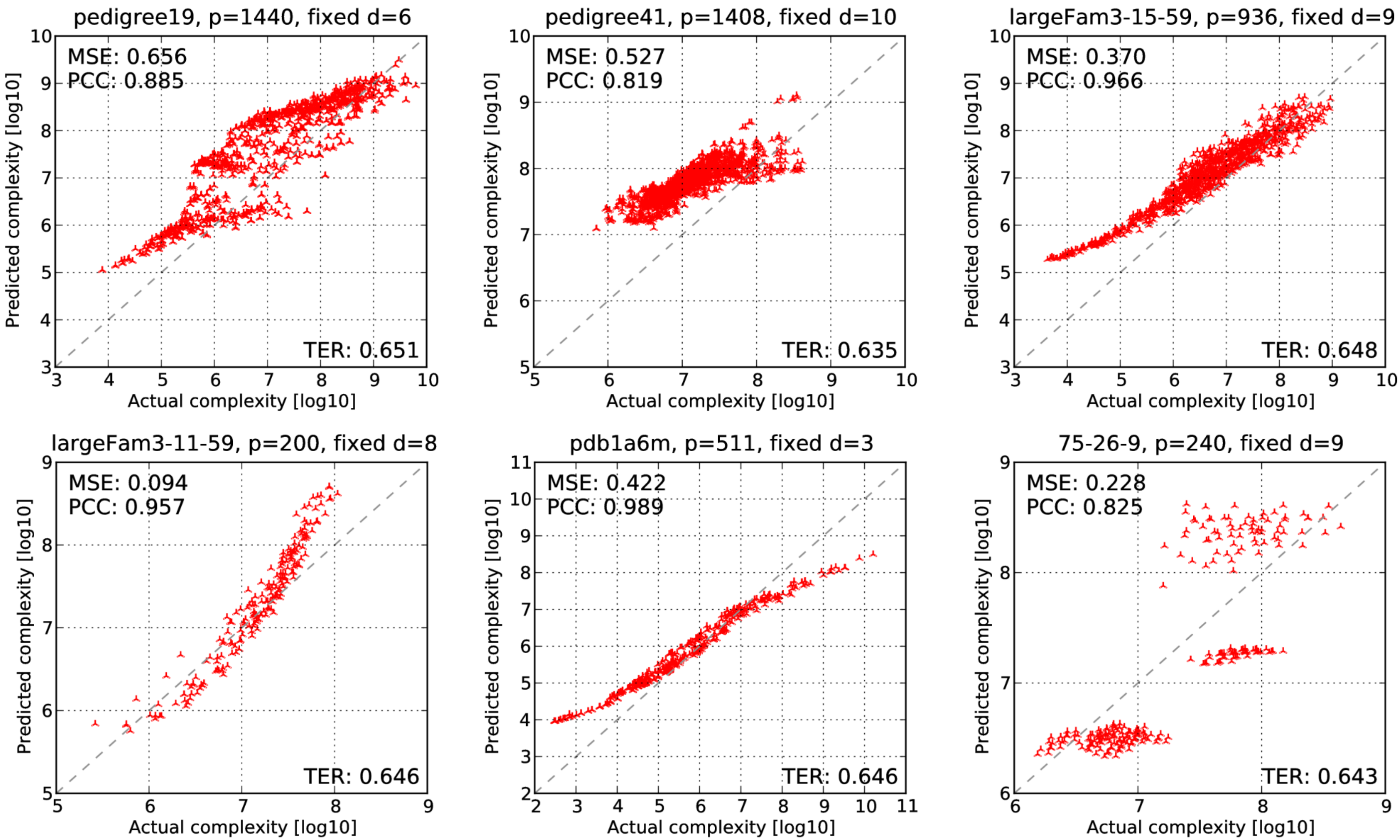
Learning per Instance (5-fold CV)



Learning per Problem Class



Learning across Problem Classes

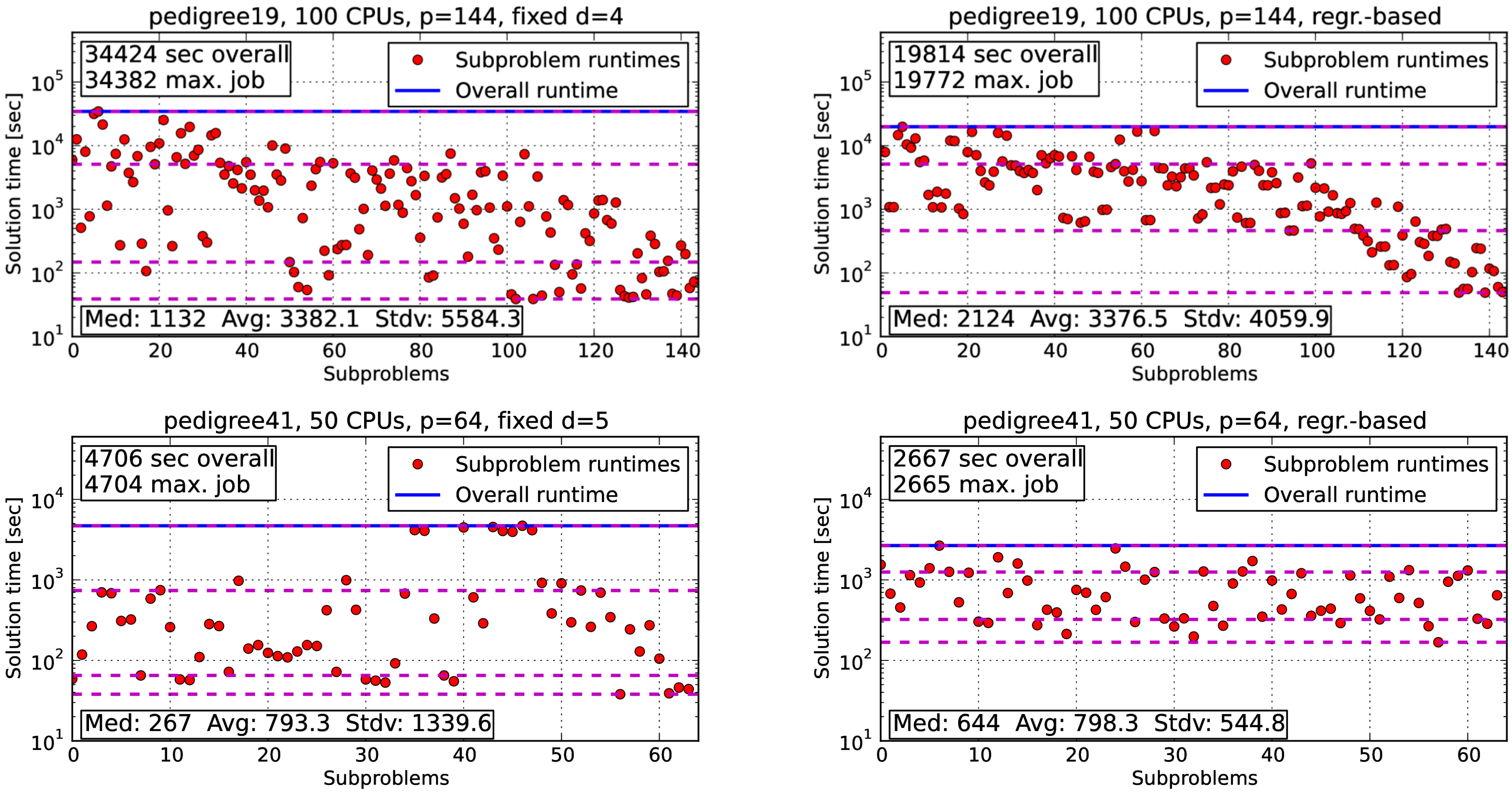


Prediction Performance Summary

- 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):



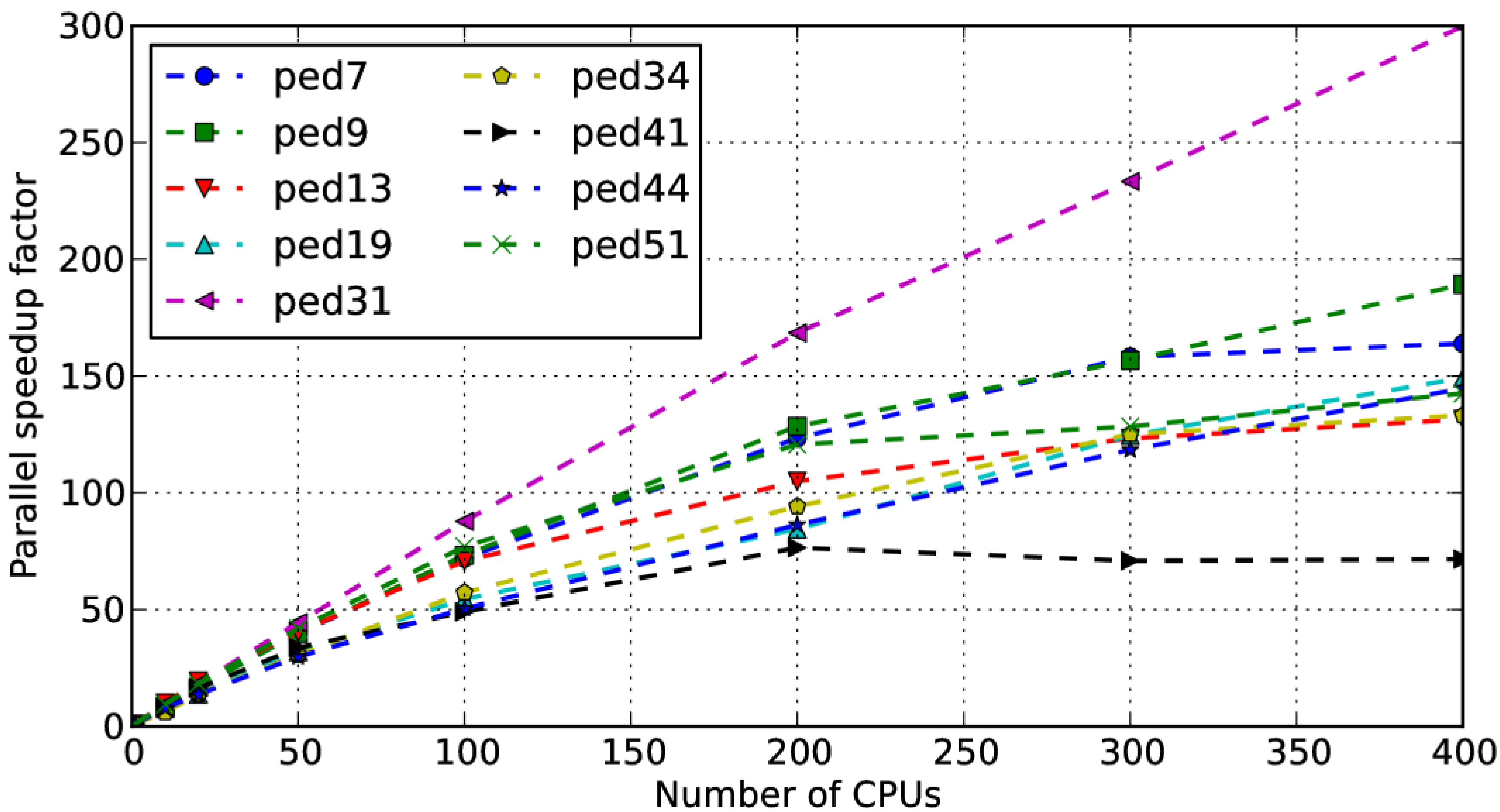
Parallel Runtime Summary

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

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

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.