Finding Most Likely Haplotypes in General Pedigrees through Parallel Branch and Bound Search

Rina Dechter, UC Irvine
(joint work with Lars Otten)

Outline

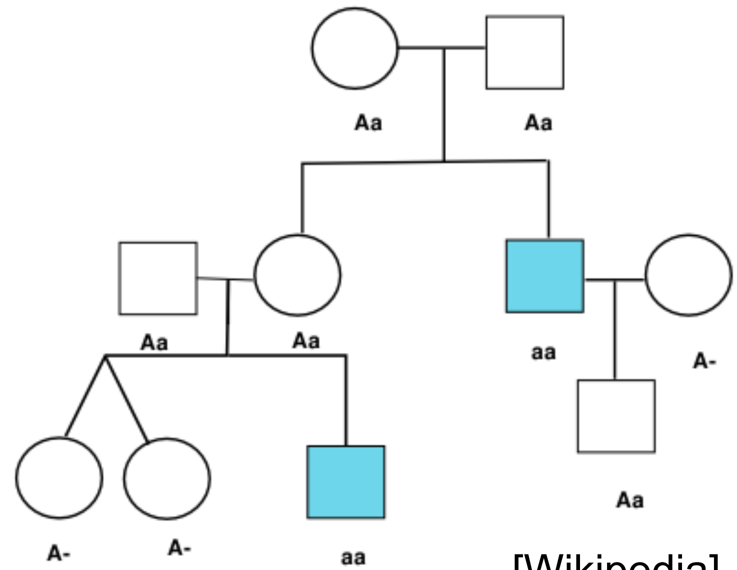
- Haplotype Inference as Bayes Net query.
- AND/OR Branch and Bound for Graphical Models.
  - State-of-the-art MPE solver. Won all three MPE tracks in PASCAL'11 Challenge.
  - Very complex instances necessitate parallelism. Run on grid of loosely coupled commodity hardware.
  - Pruning power causes significant job imbalance.
- Load Balancing through Complexity Estimation.
  - Learn linear regression models offline.
- Good parallel results on complex pedigree instances.

The Haplotype Configuration Problem

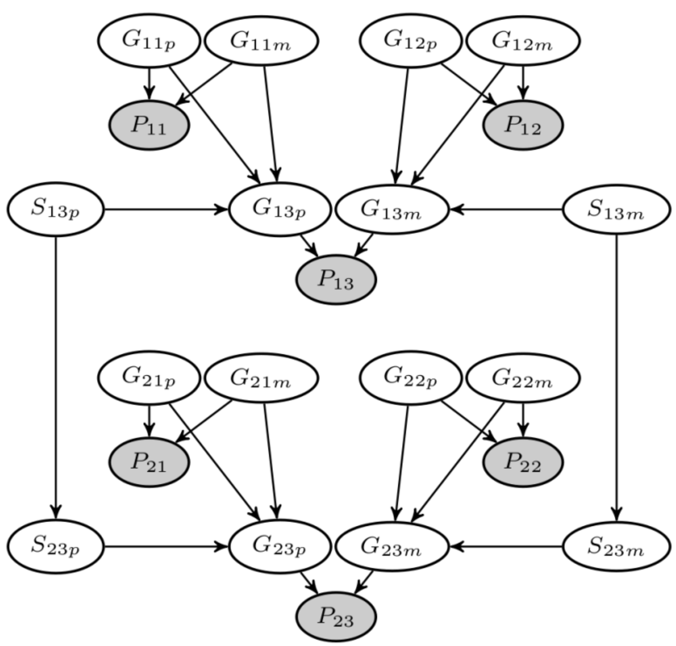
- Haplotype: the sequence of alleles at different loci inherited by an individual from one parent .
- Genotype: the two haplotypes of an individual constitute this individual's genotype. Measured genotypes results in a list of unordered pairs of alleles; one pair for each locus.
- A recombination occurrs between two loci, if an haplotype of an individual contains two alleles that resided in different haplotypes of the individual's parent.
- The **Maximum Likelihood Haplotype Configuration** problem, consists of finding a joint haplotype configuration for all members of the pedigree which maximizes the probability of the data.
- The haplotyping problem often does not have a unique solution.

Problem Statement

- Find most likely haplotype given partial genotypes.
  - *Pedigree* chart models ancestral relations.



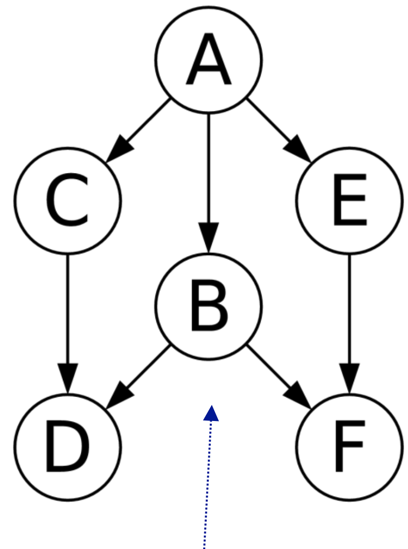
[Wikipedia]



- Encode problems as Bayesian Network.
  - "Most Probable Explanation" (MPE) yields haplotype.

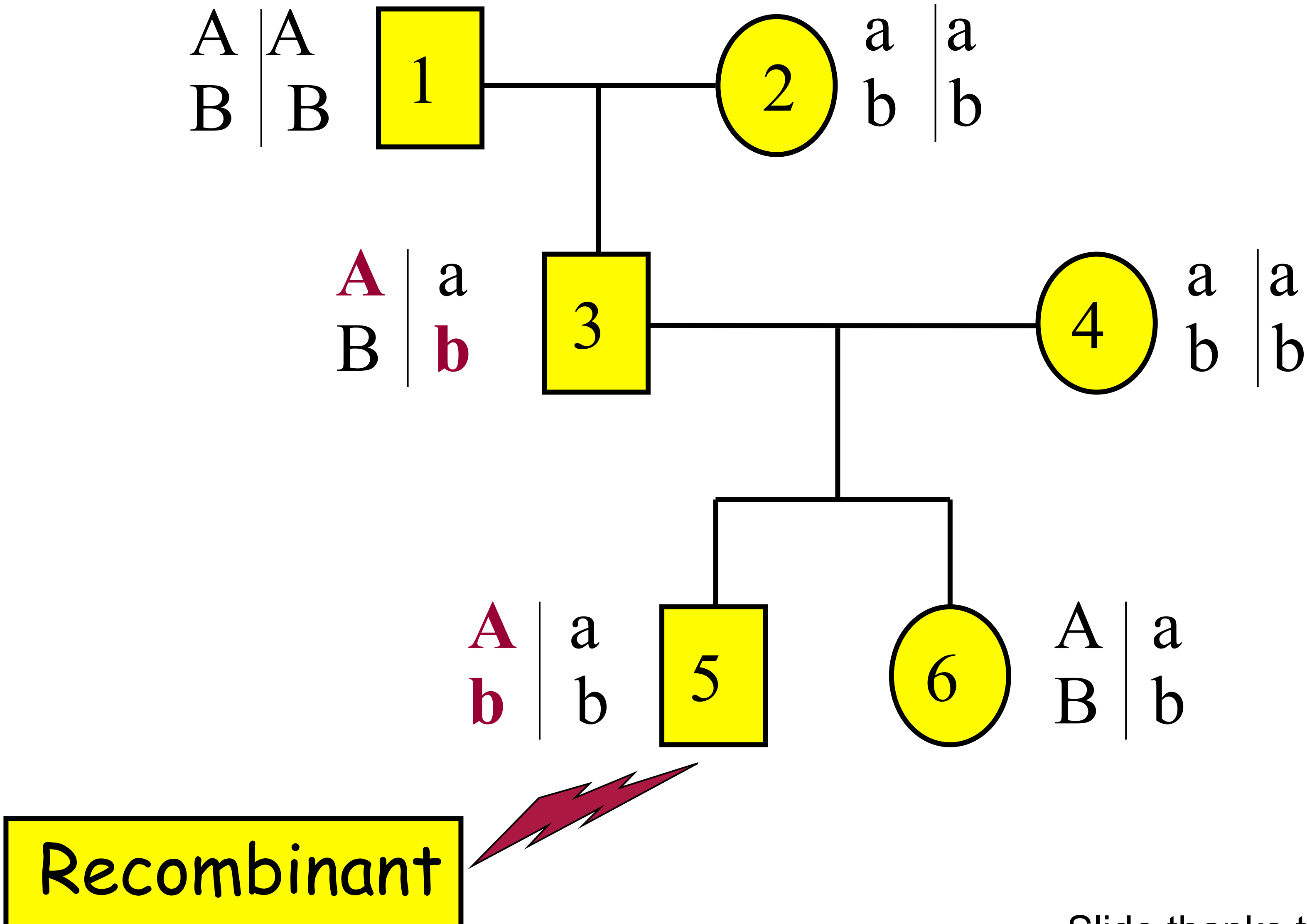
Bayesian Networks

- Given is a graphical model and a query:
  - Bayesian Network:
    - Variables *{X_i}* and conditional probability tables *{ $P(X_i|par_i)$ }* .
      - Factorizes joint probability distribution.
  - MPE Query:
    - Most Probable Explanation: Find assignment that maximizes joint probability.
  - Problem is NP-hard in general.
    - Advanced algorithms exist, expo-nential in tree width *w** of graph.



B	A	p(B|A)
0	0	0.8
1	0	0.2
0	1	0.4
1	1	0.6

Two Loci Inheritance



Slide thanks to Geiger

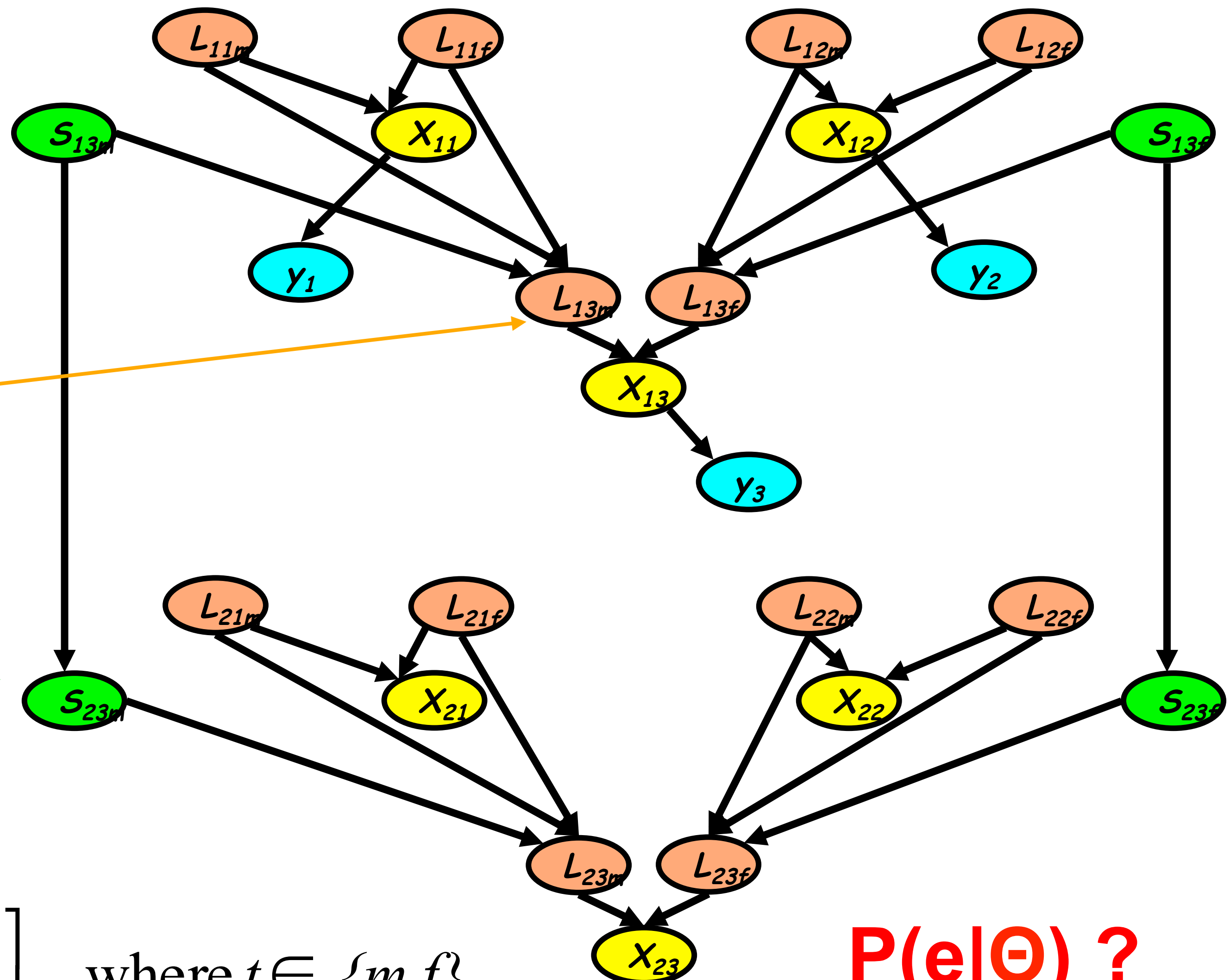
Bayesian Network for Recombination

Locus 1

Deterministic relationships

Probabilistic relationships

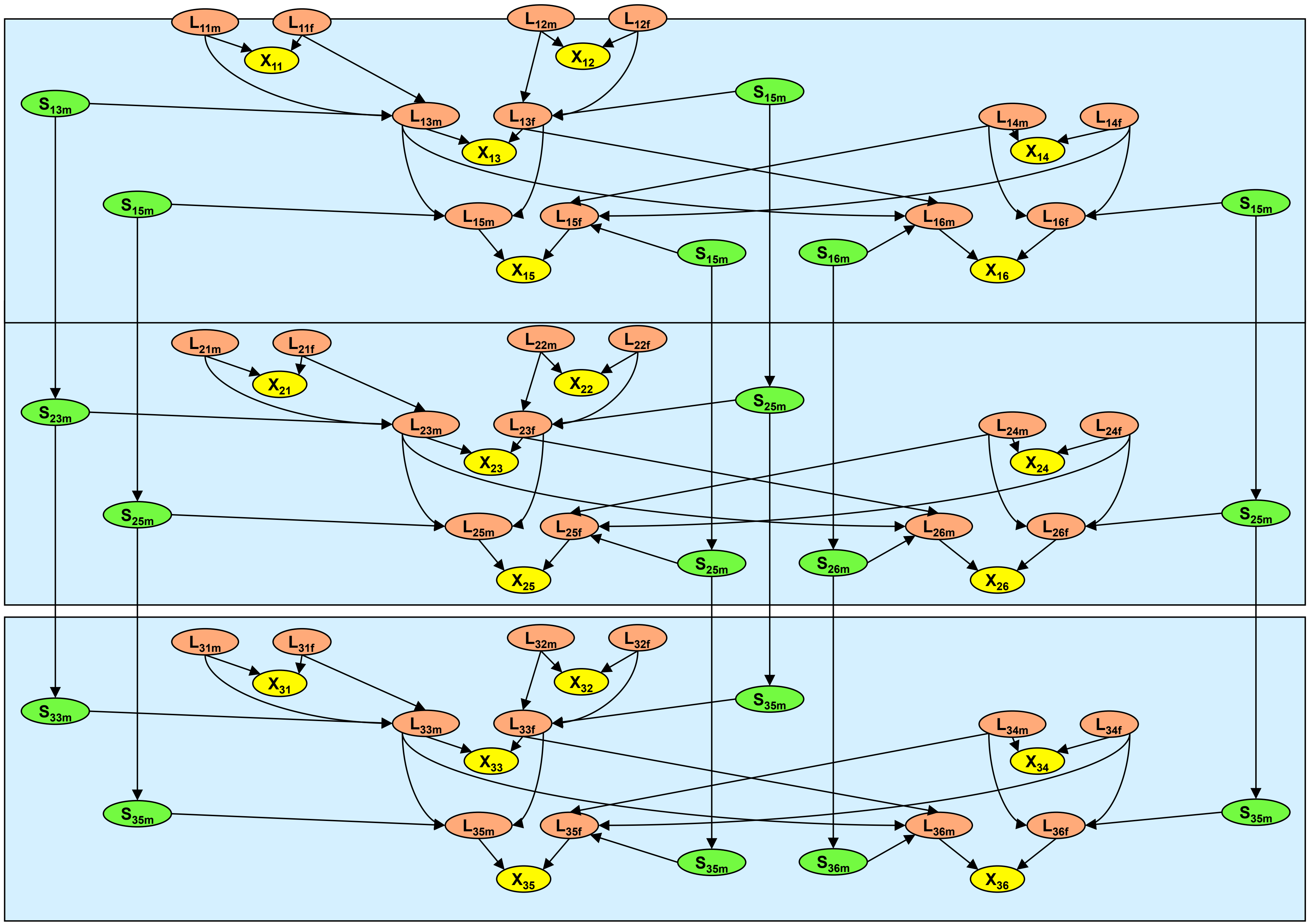
Locus 2



$$P(s_{23t} \mid s_{13t}, \theta) = \begin{bmatrix} 1-\theta & \theta \\ \theta & 1-\theta \end{bmatrix} \quad \text{where } t \in \{m,f\}$$

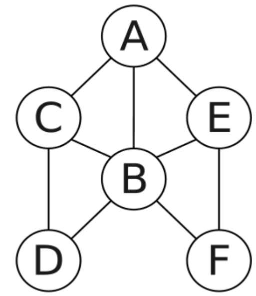
P(e|Θ) ?

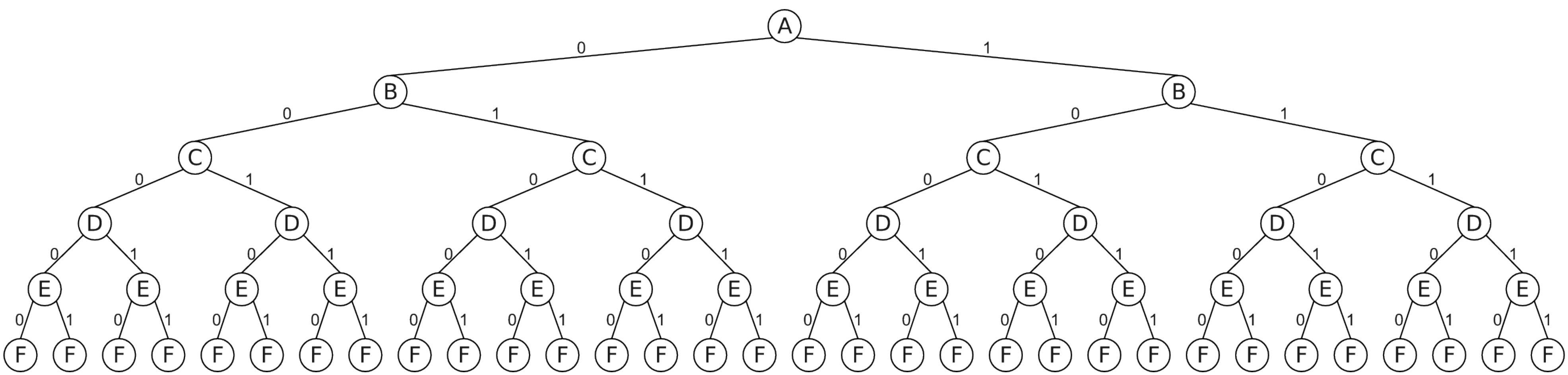
6 people, 3 markers



Searching the standard space (Depth-First Search)

- Standard depth-first search procedure:
  - Instantiate variables one at a time.
    - Backtrack in case of inconsistencies.
  - Time complexity: *exp(n)* .
    - Linear space.

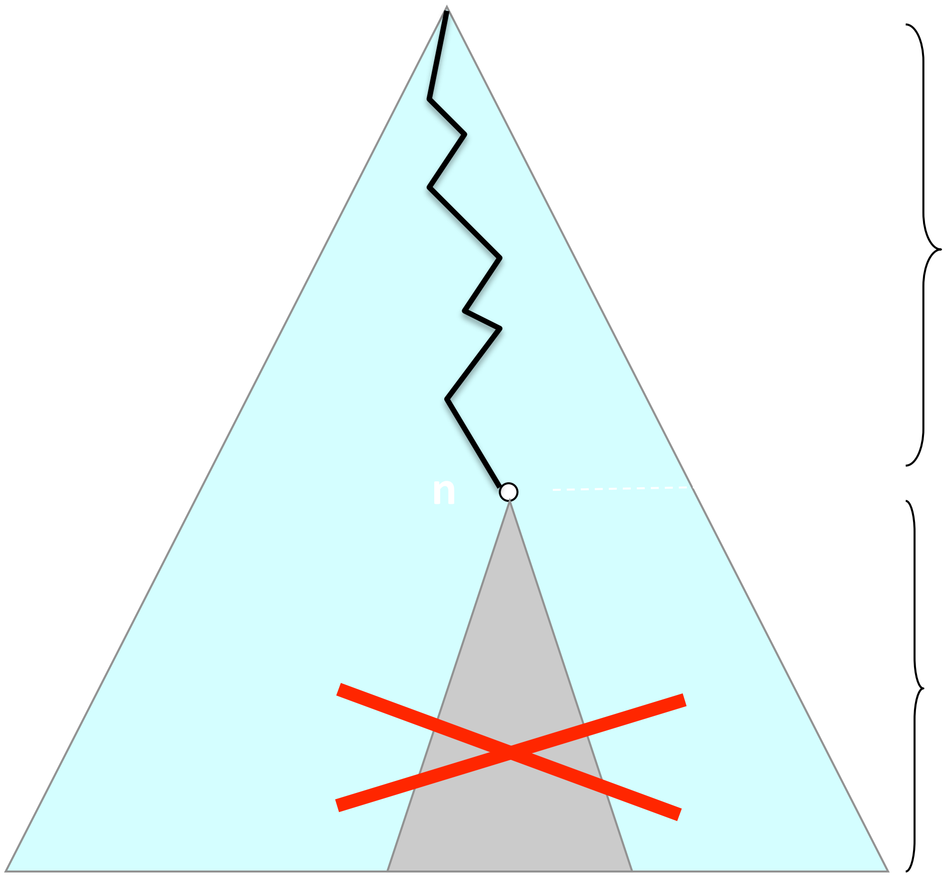




Branch-and-Bound Search

Upper Bound **UB**

Lower Bound **LB(n)**



g(n)=**cost of the search path to n**

Prune if LB(n) ≥ UB

H(n) = **estimates the optimal cost below n**

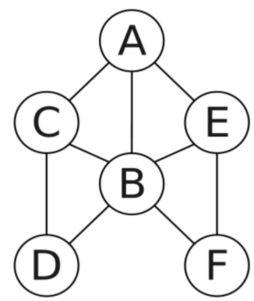
OR Search Tree

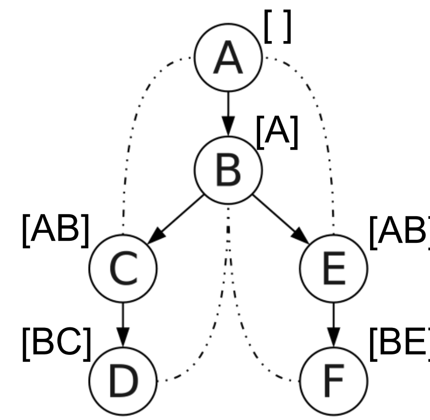
(Lawler & Wood66)

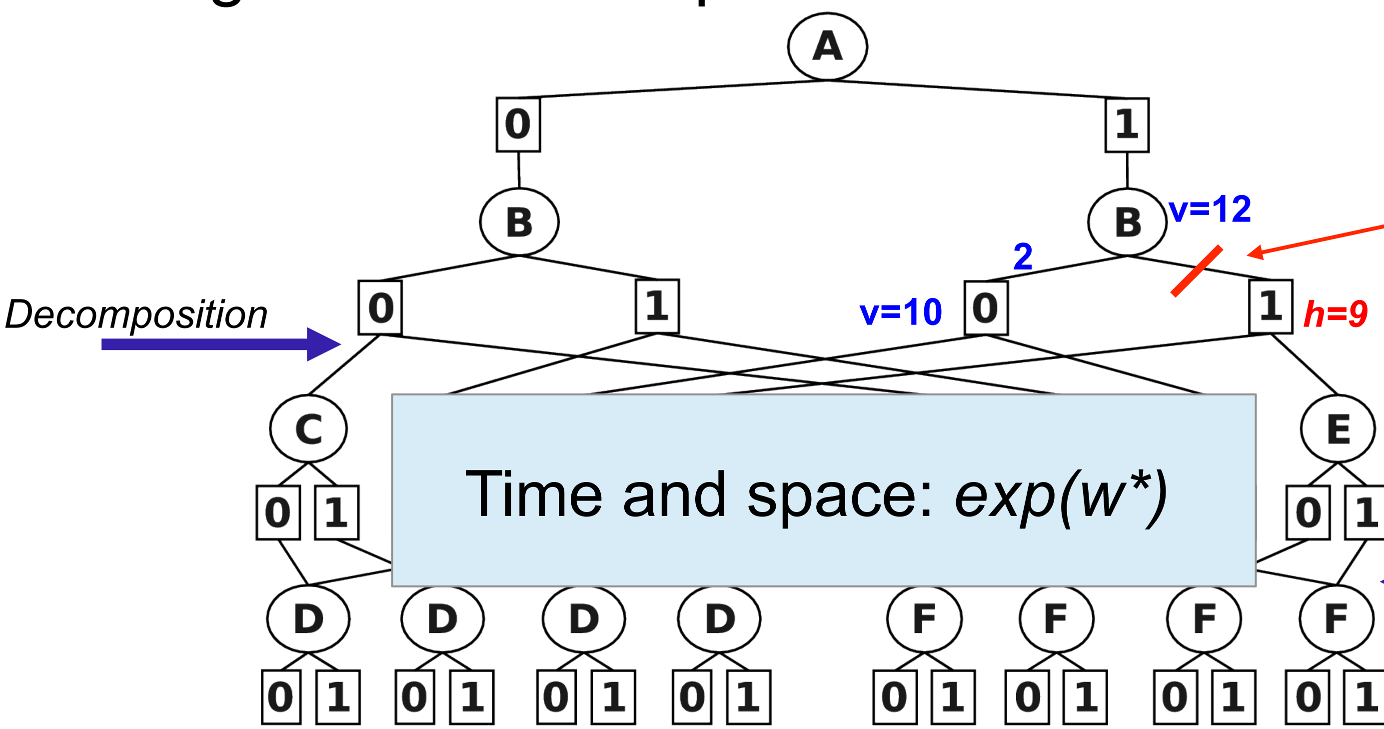
AND/OR Search Spaces

Marinescu & Dechter

- Improves upon standard search:
  - Decompose independent subproblems.
  - Merge unifiable subproblems.





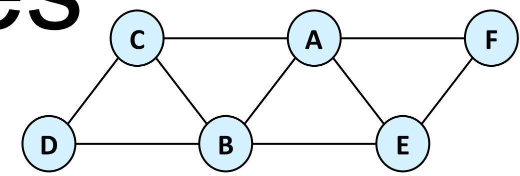


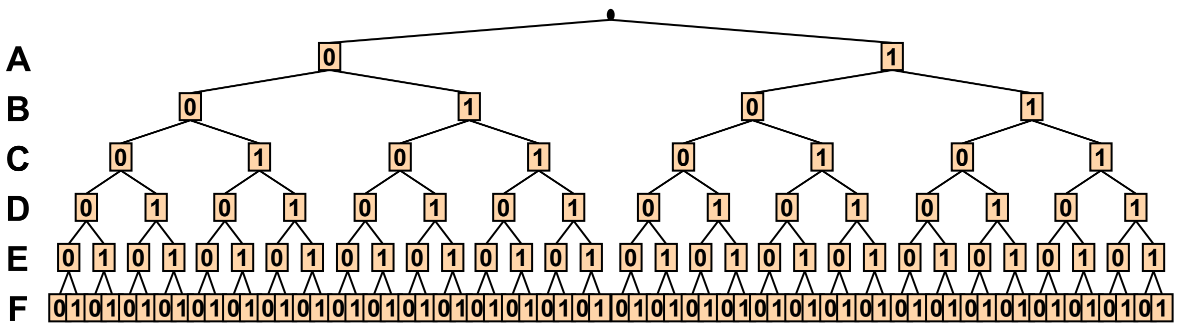
Prune based on current best solution and heuristic estimate. (mini-bucket heuristic)

Cachetable for F (independent of A)

B	E	*sol*
0	0	*0.8*
0	1	*0.3*
1	0	…
1	1	…

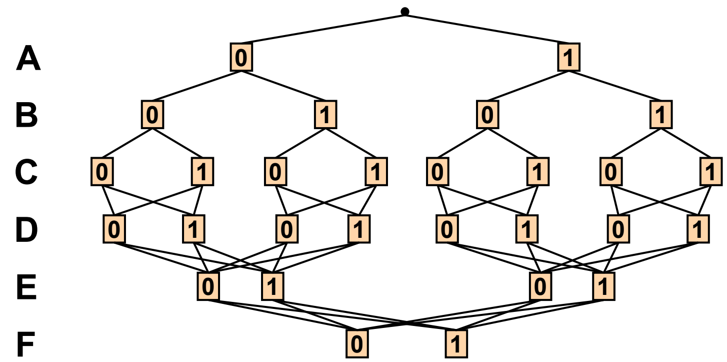
All Four Search Spaces





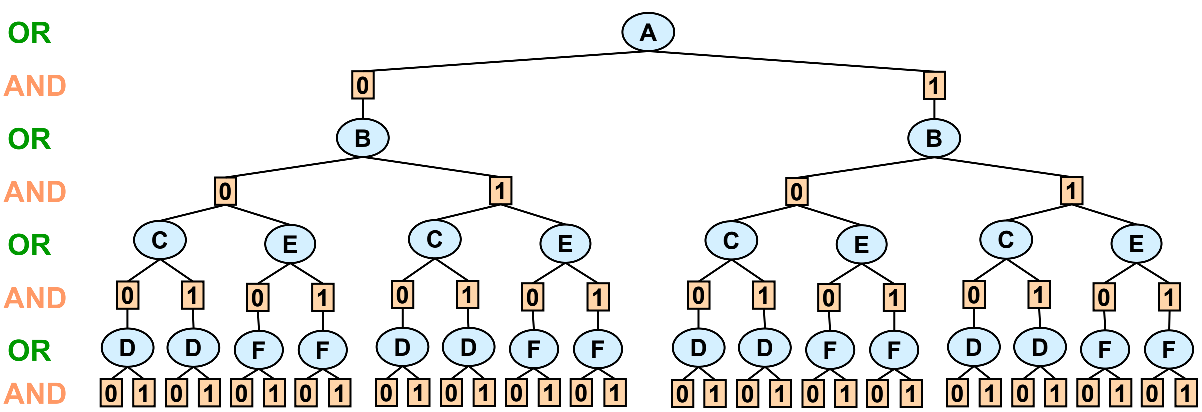
Full OR search tree

126 nodes



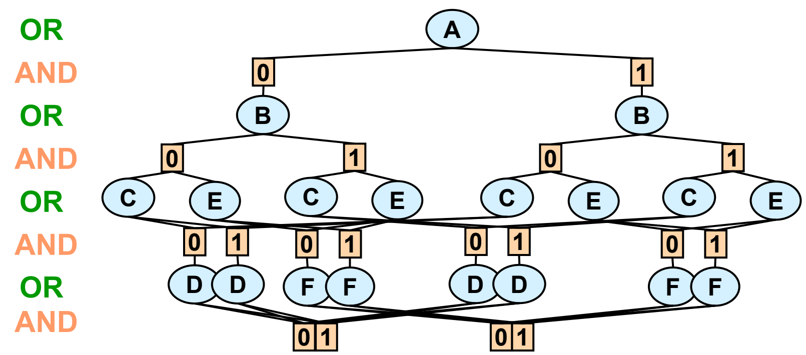
Context minimal OR search graph

28 nodes



Full AND/OR search tree

54 AND nodes

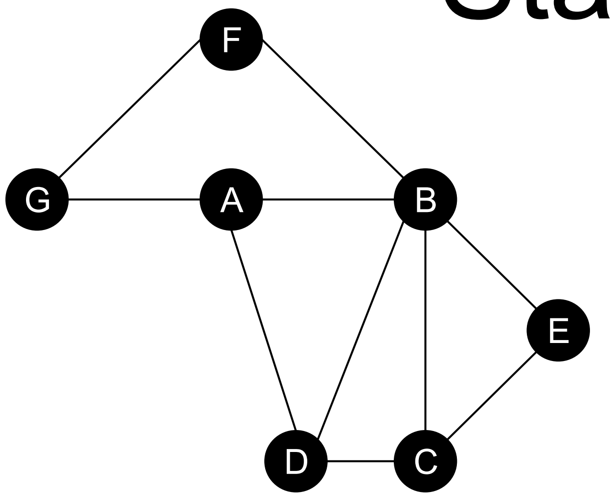


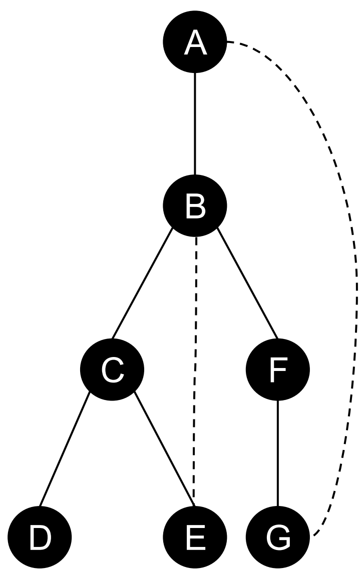
Context minimal AND/OR search graph

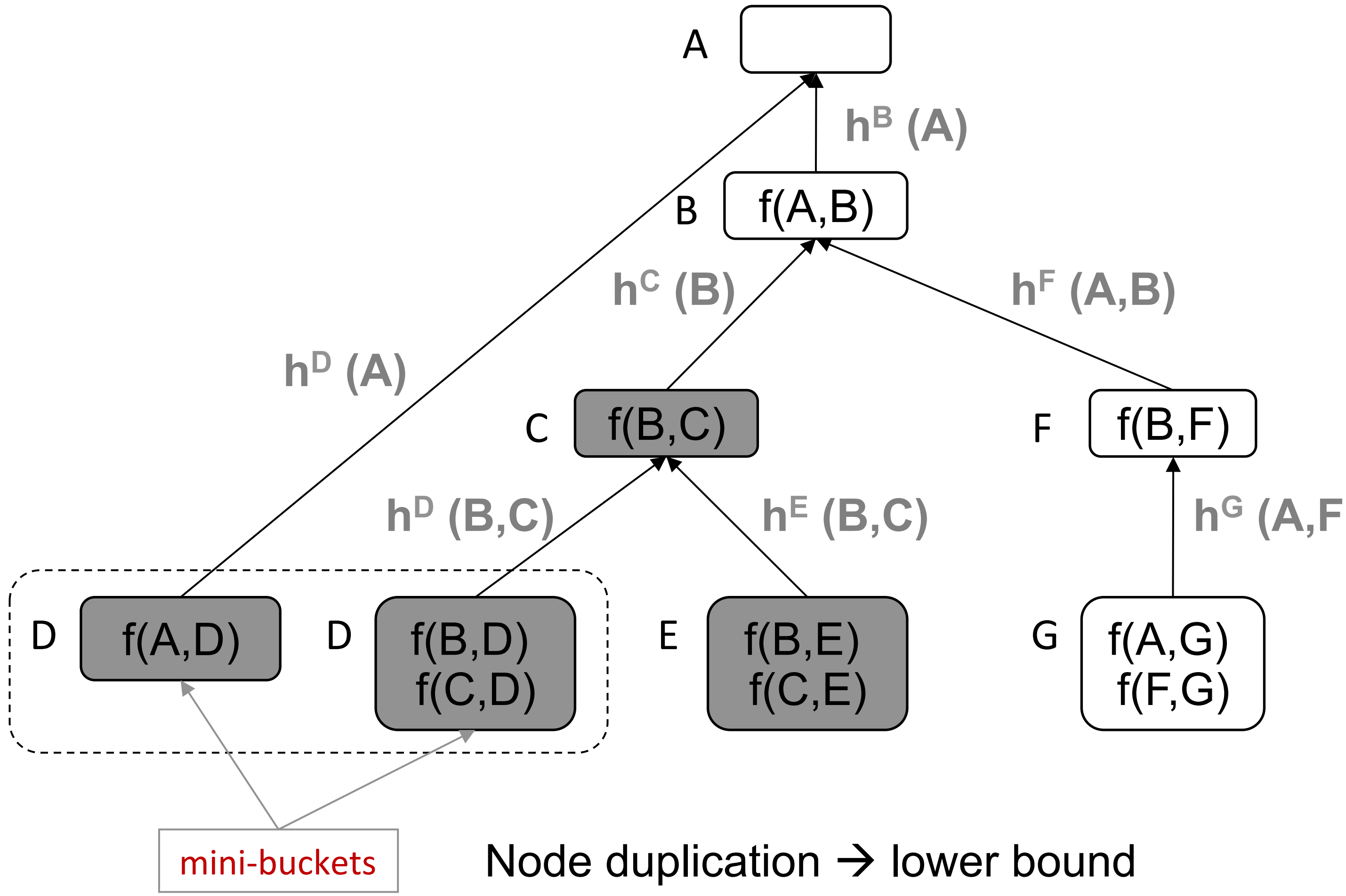
18 AND nodes

Any query is best computed
Over the c-minimal AO space

Static Mini-Bucket Heuristics







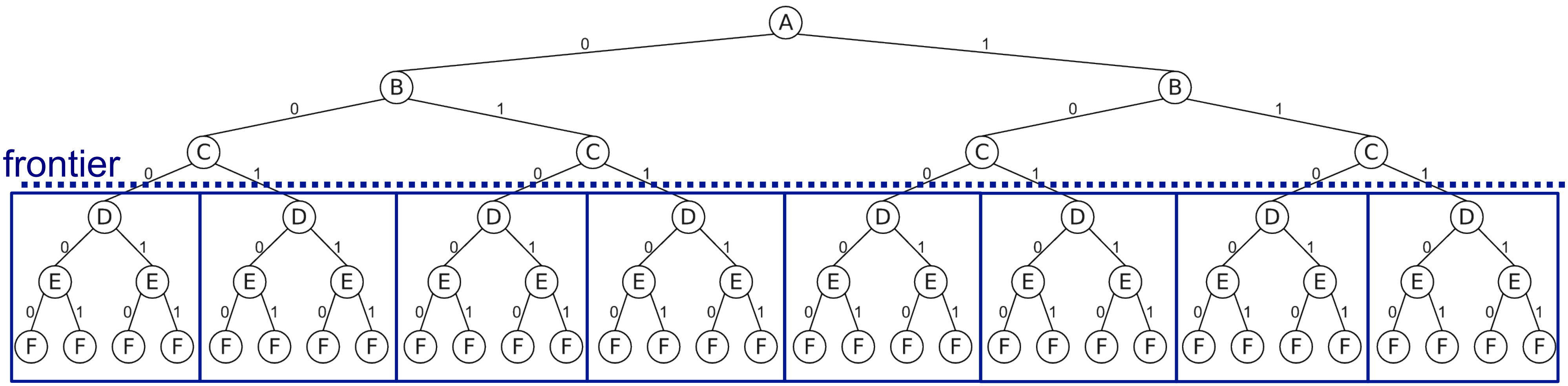
Node duplication → lower bound

$$h(a, b, c) = h^D(a) + h^D(b, c) + h^E(b, c) \leq h^*(a, b, c)$$

Ordering: (A, B, C, D, E, F, G)

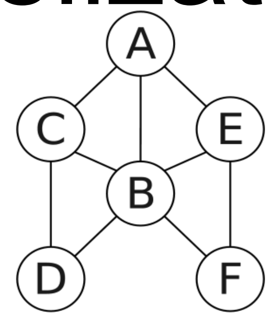
Searching in Parallel

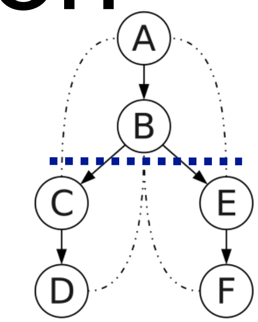
- Parallel tree search. [Kumar]
- Introduce *parallelization frontier* :
  - Condition on partial instantiations.
  - Solve subtrees in parallel and combine solutions.
    - Speedup at most linear.

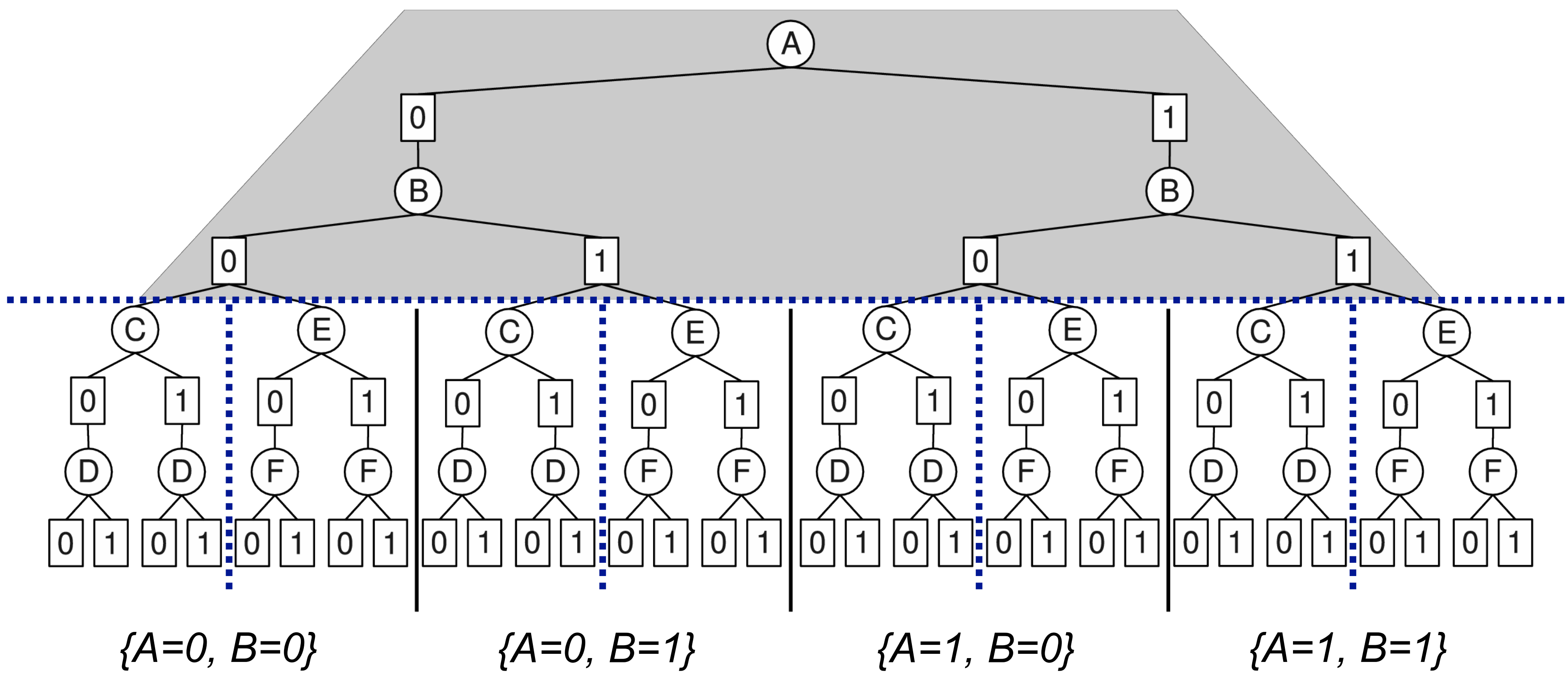


AND/OR Search Parallelization

- Depth 2 cutoff: **8** subproblems.
  - Conditioning <u>and</u> decomposition.
  - Full parallelization upfront (static).

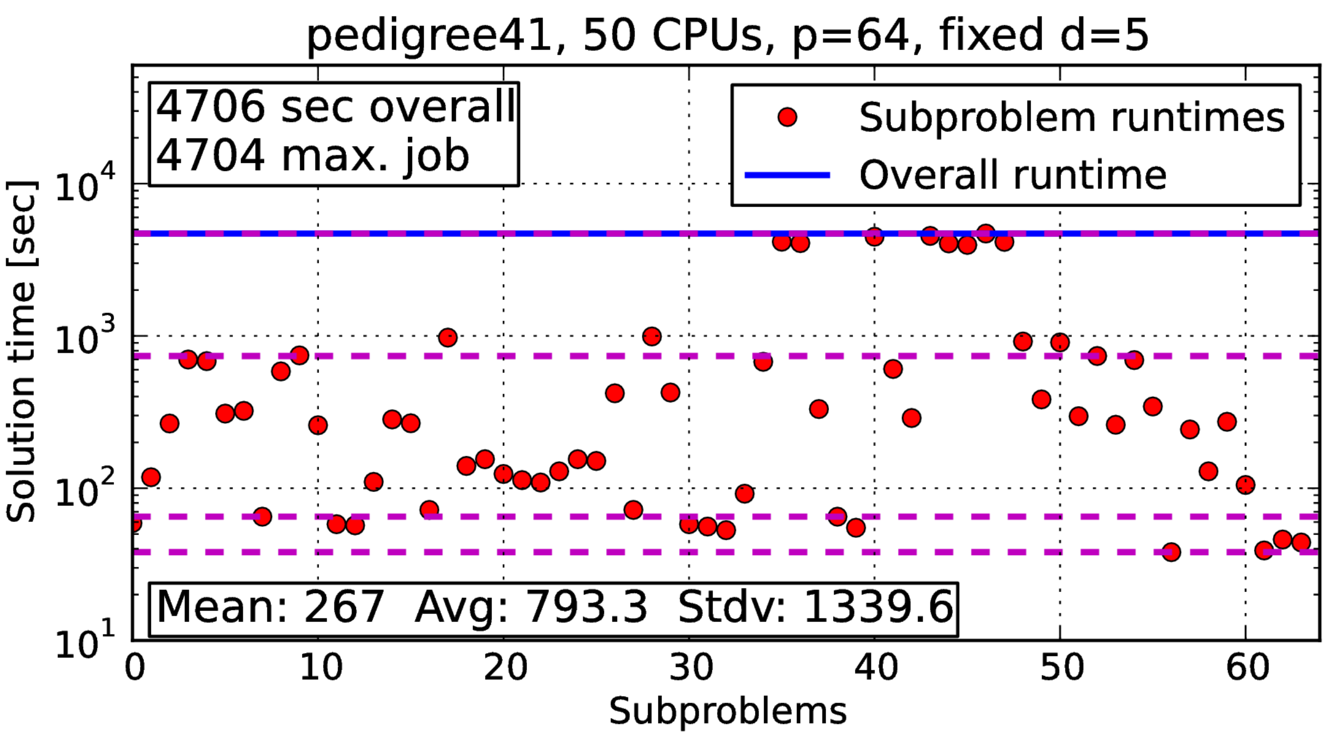






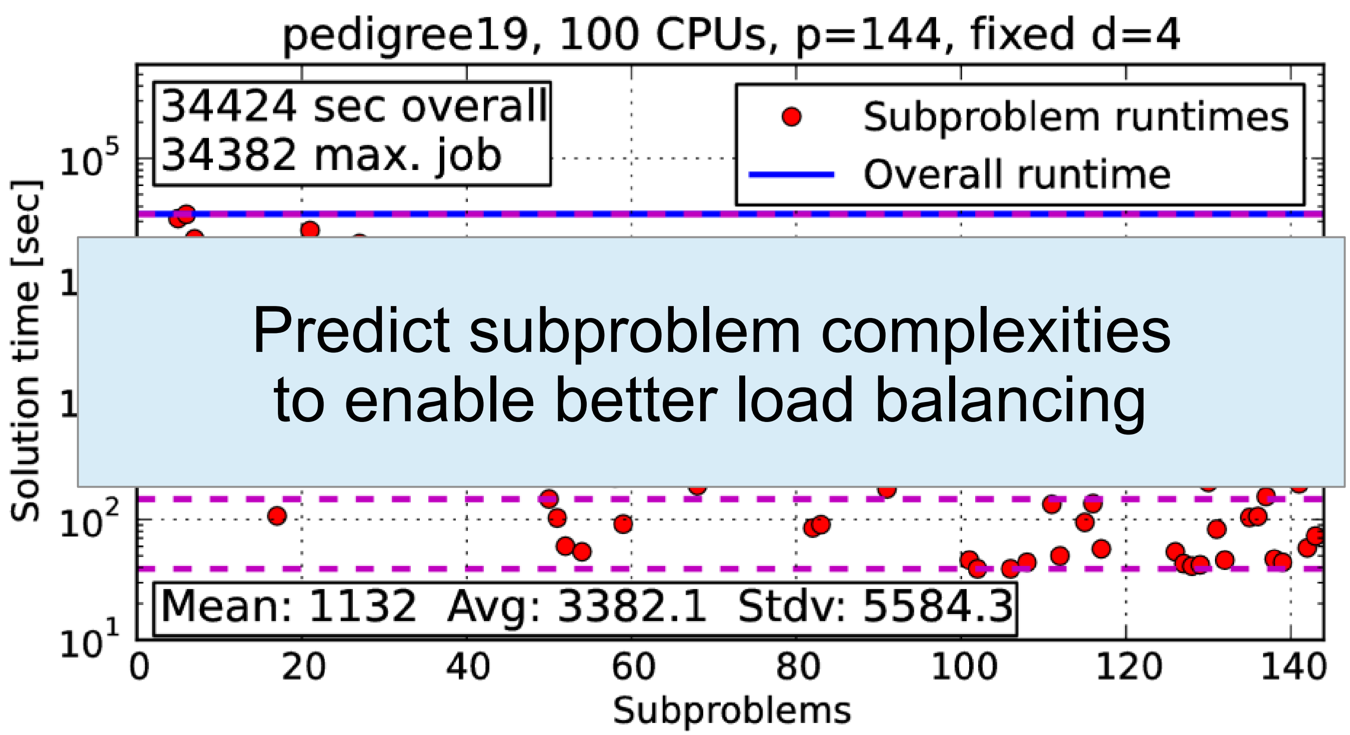
Subproblem Variance

- Fixed-depth cutoff:
  - Subproblems have identical structure.
  - But large variance in runtime complexity?



Subproblem Variance

- In spite of identical structure:
  - Effect of bounds and pruning differs vastly.
  - Few subproblems dominate overall performance.



Predict subproblem complexities to enable better load balancing

Subproblem Complexity Prediction

- 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 m sample subproblems n_k , minimize MSE:

$$\frac{1}{m} \sum_{k=1}^{m} \left(\sum_j \lambda_j \varphi_j(n_k) - \log N(n_k) \right)^2$$

34 Subproblem Features

- Static, structural properties:
  - Number of variables.
  - Avg. and max. width.
  - Height of sub pseudo tree.
  - Etc.
- Dynamic, runtime properties:
  - Upper and lower bound.
  - Pruning ratio and depth of small AOBB probe.
  - Etc.

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

Problem Features

- Subproblem variable statistics (static):
  - N: Number of variables in subproblem.
  - Min, Max, mean, average, and std. dev. of variable domain sizes in subproblem.
- Pseudo tree depth/leaf statistics (static):
  - h: Depth of subproblem root in overall search space.
  - Min, max, mean, average, and std. dev. of depth of subproblem pseudo tree leaf nodes, counted from subproblem root.
  - L: Number of leaf nodes in subproblem pseudo tree.

Problem Features

- Pseudo tree width statistics (static):
  - Min, max, mean, average, and std. dev. of induced width of variables within subproblem.
  - 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):
  - Lower bound *L* on subproblem solution cost, derived from current best overall solution.
  - Upper bound *U* on subproblem solution cost, provided by mini bucket heuristics.
  - Difference *U-L* between upper and lower bound, expressing "constrainedness" of the subproblem.

Problem Features

- Pruning ratios (dynamic), based on running 5000 node expansion probe of AOBB:
  - Ratio of nodes pruned using heuristic upper bound.
  - Ratio of nodes pruned due to determinism (zero probabilities, e.g.).
  - Ratio of nodes corresponding to pseudo tree leaf.
- Sample statistics (dynamic), based on running 5000 node expansion probe of AOBB:
  - Average depth of terminal search nodes within probe.
  - Average node depth within probe (denoted *d*).
  - Average branching degree, defined as $\sqrt[d]{5000}$

Problem Features

- Various:
  - Mini bucket *i*-bound parameter.
  - Max. subproblem variable context size minus mini bucket *i*-bound.

- In total 34 features.

Specifics of Learning

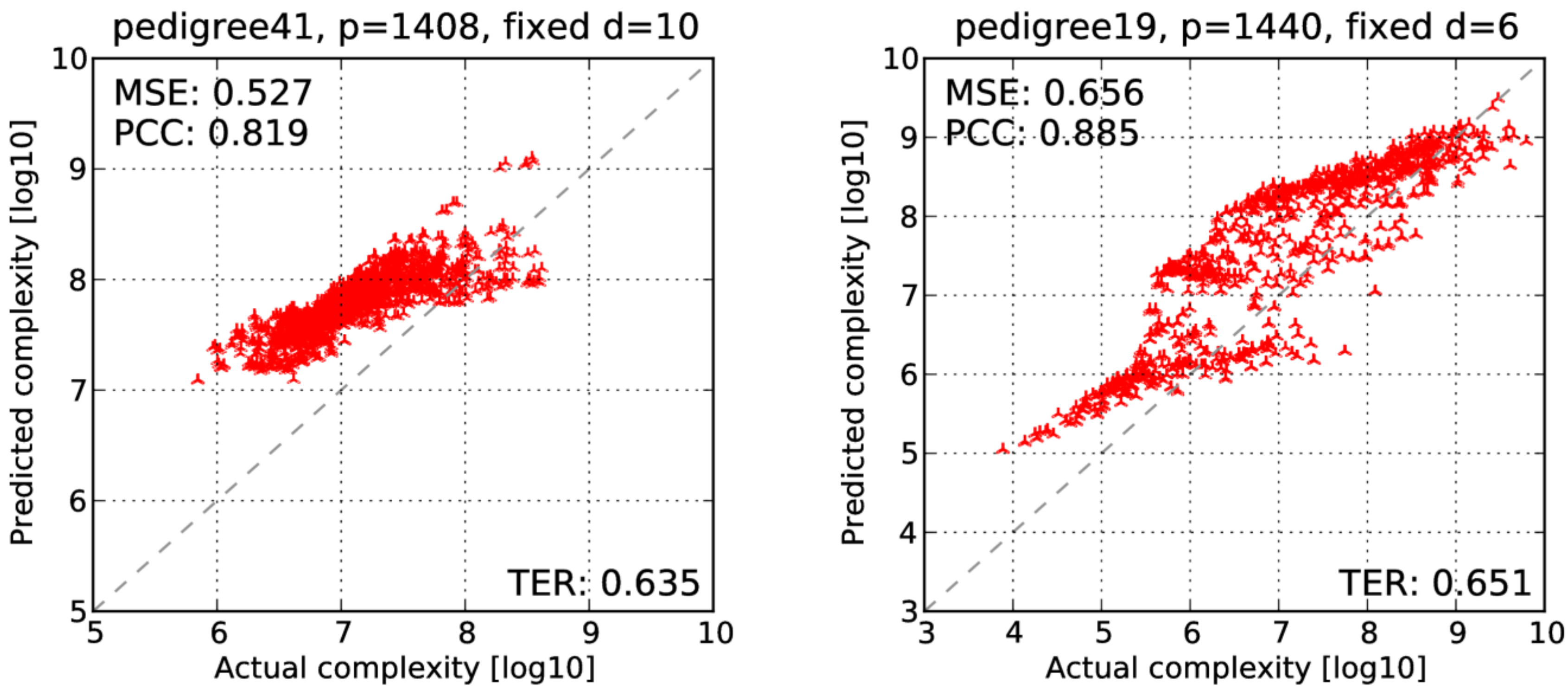
- Lasso learning to avoid overfitting.
  - Add regularization term to MSE.
    $$\frac{1}{m}\sum_{k=1}^{m}\left(\sum_{j}\lambda_j\,\varphi_j(n_k)-\log N(n_k)\right)^2+\alpha\,\|\lambda\|_1$$
  - Encourages sparsity, implicit feature selection.
  - α = 0.1 through cross validation.
- Measure:
  - MSE: Prediction error (MSE)
  - TER: Training error (MSE)
  - PCC: Pearson correlation coefficient (normalized cov.)

Regression Results

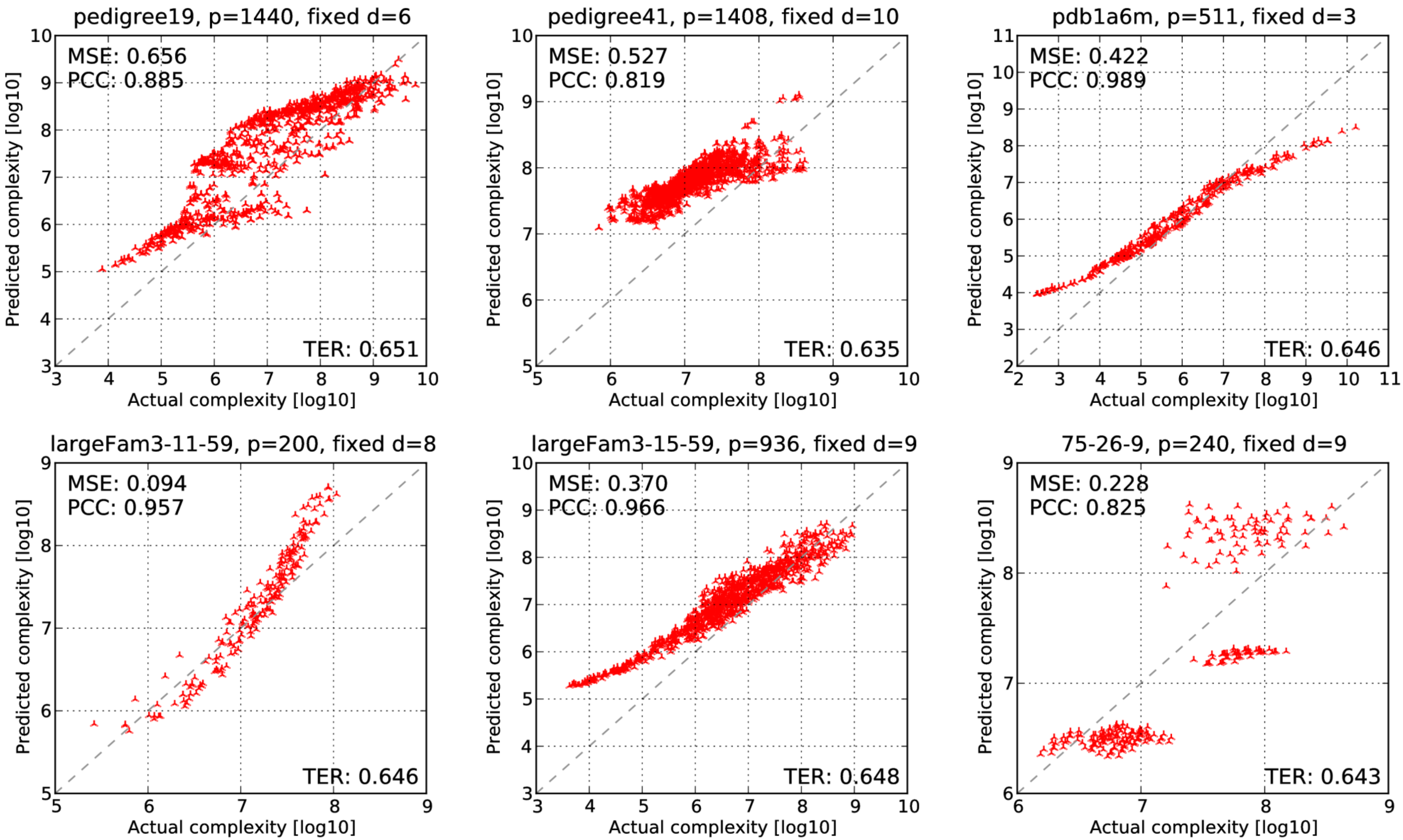
- 31 instances total (13 pedigrees) from 4 classes.
  - Run each with fixed-depth cutoff.
  - Choose up to 500 subproblem samples.
  - Yields 11,500 samples overall.
- Most general regression approach:
  - Train model on samples from 30 instances.
  - Test on samples from remaining instance.
- Other scopes of learning evaluated:
  - Per-instance and per-class, comparable results.

Regression Results

- Prediction on two pedigree examples:
  - Test error (MSE) close to training error (TER).
  - Fairly high correlation coefficient.



Across all Problems/Classes



Parallelization Scheme

- Iteratively split estimated largest subproblem.
  - Until desired number of subproblems is reached.

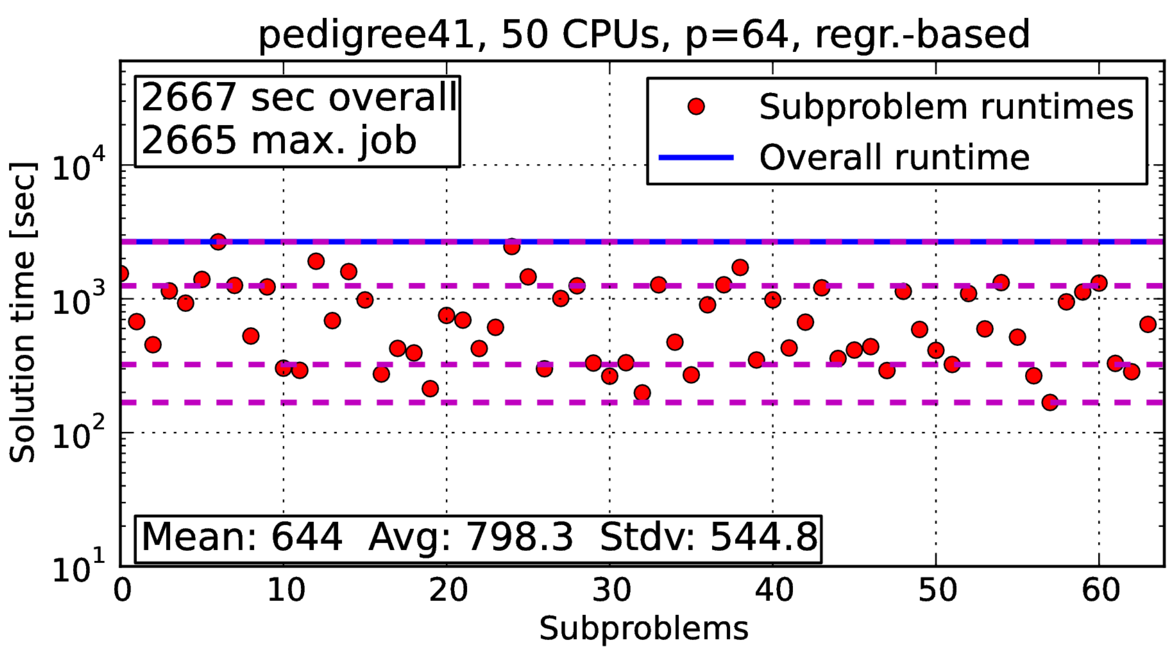
Algorithm 1 Finding the parallelization frontier

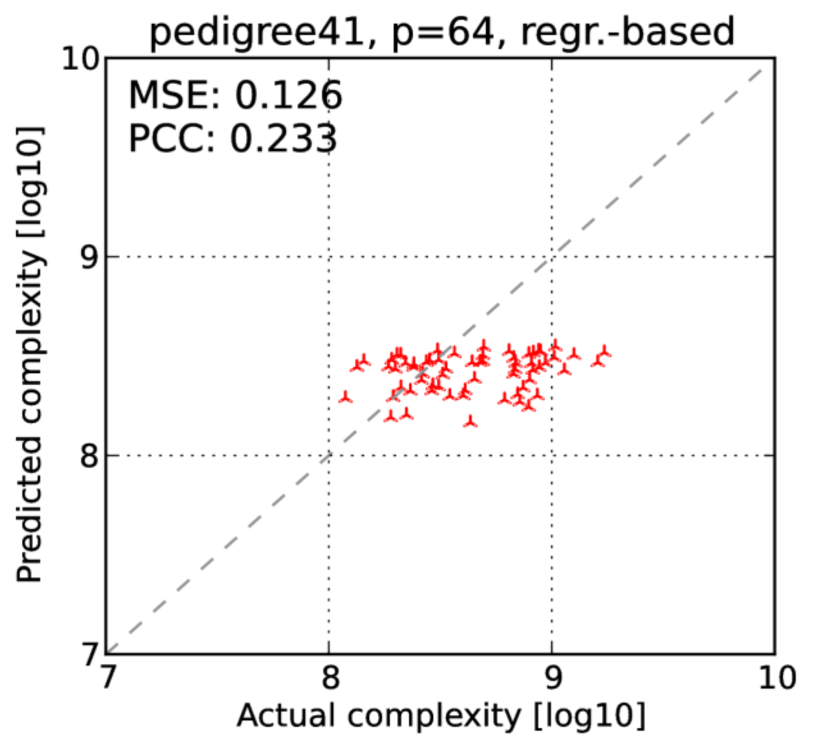
Input: Pseudo tree $\mathcal{T}$ with root X_0, subproblem count p, subproblem complexity estimator $\hat{N}$.
Output: Set F of subproblem root nodes with $|F| \geq p$.

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: $\quad F \leftarrow F \cup children(n')$

Detailed Parallel Results

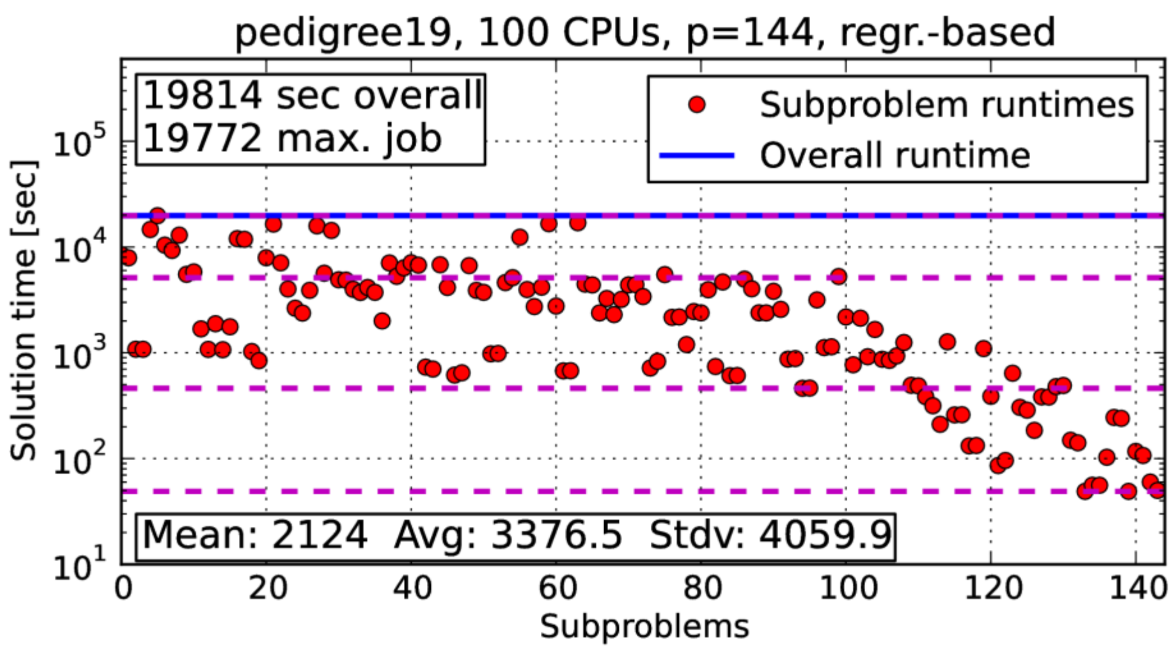
- Pedigree41:
  - Left: detailed subproblem statistics
  - Right: actual vs. predicted complexity

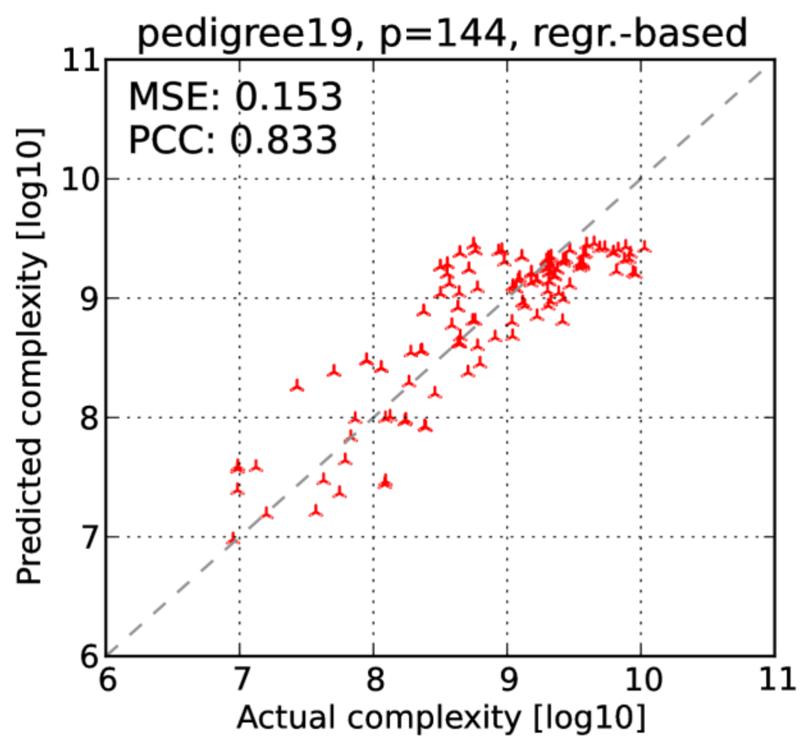




Detailed Parallel Results

- Pedigree19:
  - Left: detailed subproblem statistics.
  - Right: actual vs. predicted complexity.





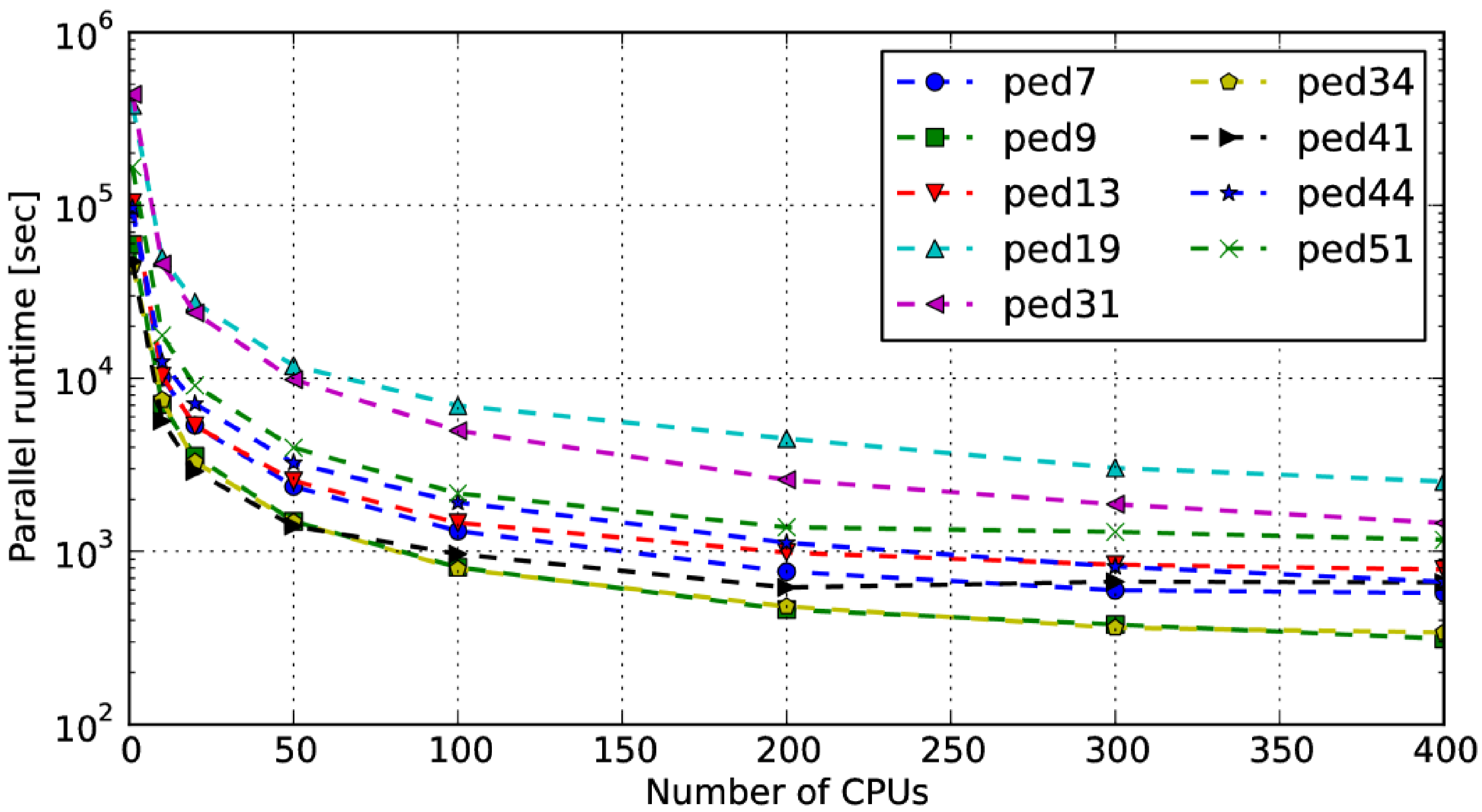
Overall Parallel Results

- Pedigrees with 20-25 individuals and 20-25 loci.
  - *n* is number of variables, *k* max. domain size, *w* induced width, h pseudotree height. Runtime in *hh:mm*.

inst	n	k	w	h	seq	Number of CPUs 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

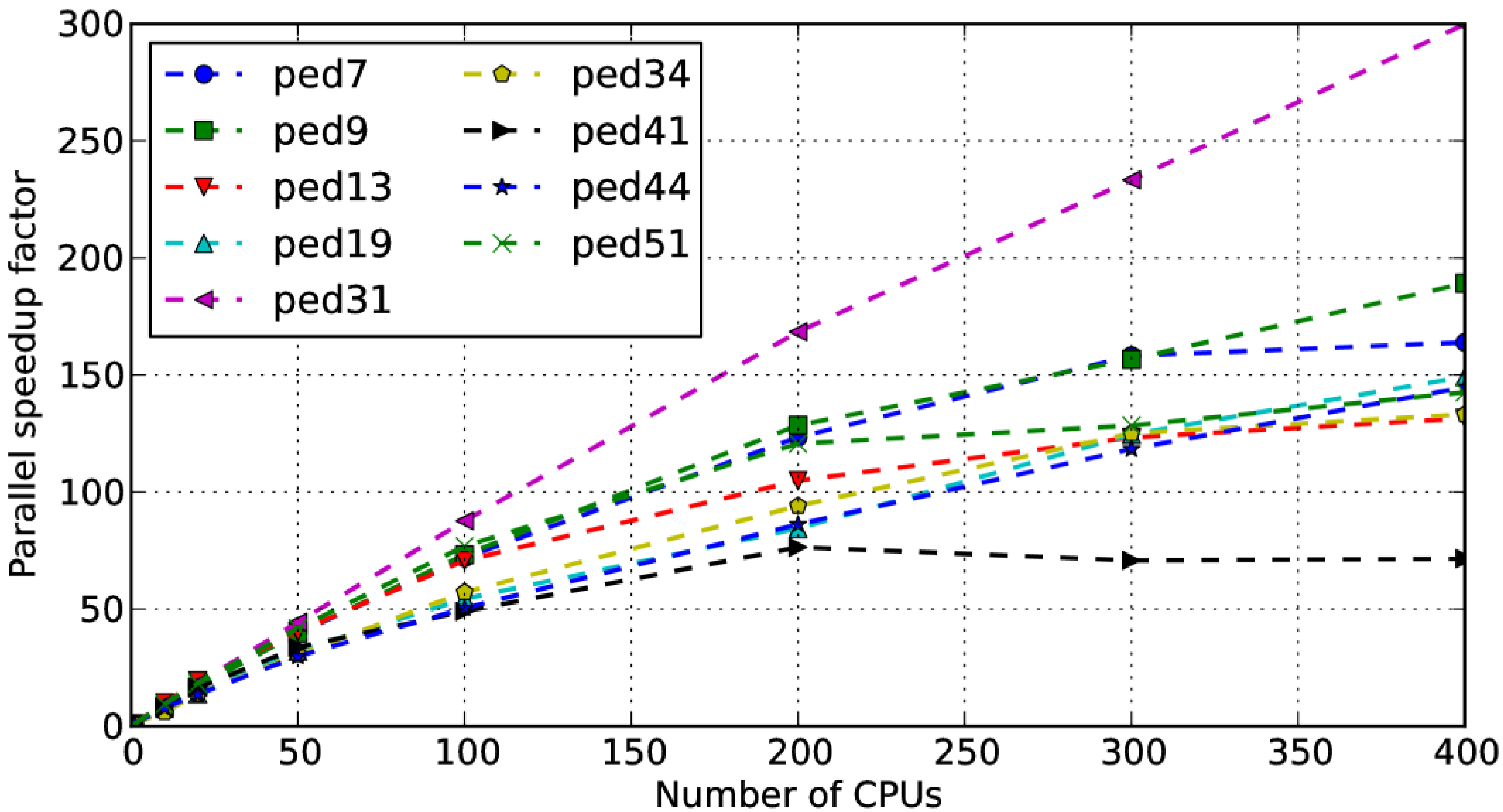
Overall Parallel Results

- Parallel runtimes plotted.



Parallel Speedup

- Speedup relative to sequential algorithm.
  - Highest potential with most complex problems.



Summary

- Express haplotype computation as MPE query.
  - Exploit graph structure and apply advanced AND/OR search algorithms (decomposition and caching and mini-bucket heuristics).
- Parallel AND/OR Branch and Bound:
  - Powerful pruning impedes load balancing.
  - Learn complexity regression model offline.
- Empirical results: Improved load balancing.
  - Good parallel performance and speedup on hard pedigree instances.
- Deployed in Superlink-Online SNP:
  - http://cbl-hap.cs.technion.ac.il/superlink-snp