HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI
MATEMAATTIS-LUONNONTIETEELLINEN TIEDEKUNTA
MATEMATISK-NATURVETENSKAPLIGA FAKULTETEN
FACULTY OF SCIENCE

PREDICTING THE HARDNESS OF LEARNING BAYESIAN NETWORKS

Brandon Malone, Kustaa Kangas, Matti Järvisalo, Mikko Koivisto, Petri Myllymäki

Motivation: There are various algorithms for finding a Bayesian network structure that is optimal with respect to a given scoring function. Due to the chaotic nature of the running times of such algorithms, it is *a priori* not clear which algorithm will solve a given problem instance fastest.
Results: 1) We can train models that predict the running time of an algorithm on a given instance with reasonable accuracy based on *features* of the instance. 2) Even very simple features admit an efficient hybrid algorithm, or *portfolio*, that runs the algorithm predicted to be fastest.

INTRODUCTION

BAYESIAN NETWORKS

A Bayesian network is a graphical model on random variables $X_1, \ldots, X_n$.

The *structure* of a Bayesian network is a directed **acyclic** graph (DAG) G.

A *scoring function* s measures how well G fits observed data on the variables. Typical scoring functions decompose into a sum

$$s(G) = \sum_{i=1}^{n} s_i(G_i) \, ,$$

where G_i is the set of parents of X_i in G.

Common s: penalized likelihood, minimum description length, BDeu, etc.

STRUCTURE LEARNING PROBLEM

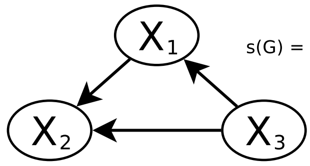
Input: A set $\mathcal{G}_i$ of candidate parent sets for each variable X_i and the local scores $s_i(G_i)$ for all $G_i \in \mathcal{G}_i$.

Task: Find a DAG G such that $G_i \in \mathcal{G}_i$ and the score $s(G)$ is maximized. (NP-hard)

G_1	$s_1(G_1)$
$\{X_2, X_3\}$	7
$\{X_2\}$	4
$\{X_3\}$	6
Ø	3

G_2	$s_2(G_2)$
$\{X_1, X_3\}$	8
$\{X_1\}$	3
$\{X_3\}$	2
Ø	1

G_3	$s_3(G_3)$
$\{X_1, X_2\}$	4
$\{X_1\}$	3
$\{X_2\}$	3
Ø	2



s(G) = 6 + 8 + 2 = 16

ALGORITHMS

Various exact algorithms are guaranteed to find an optimal G while avoiding exhaustive search in the space of all DAGs:

Dynamic programming over variable subsets finds an optimal ordering of variables that is compatible with an optimal DAG.

A* search formulates the DP approach as a shortest-path problem, uses admissible best-first heuristics to prune the search space.

Integer linear programming searches a convex polytope where each vertex is a feasible solution. Cutting planes are added during search to enforce acyclicity.

Branch and bound searches a relaxed space of cyclic graphs and breaks cycles by branching on arcs to remove in best-first order.

MODEL TRAINING

1. Select a set of training instances.
2. Select a set of instance *features*.
3. Compute the features of each instance.
4. Run all algorithms on all instances and record their running times.
5. Using the data, learn for each algorithm a model that maps a feature vector to a running time prediction.

FEATURES

We consider 74 features of various types:

1. Number of variables n, number of candidate parent sets $m = \sum_{i=1}^{n} |\mathcal{G}_i|$ (typically $|\mathcal{G}_i| \ll 2^{n-1}$ due to pruning).
2. Sizes of $G_i \in \mathcal{G}_i$: mean, variance, etc.
3. Properties of cyclic upper bound graphs: average degree, number of leaves, etc.
4. Probing: Properties extracted by running one algorithm for a few seconds: best network found, lower bounds, etc.

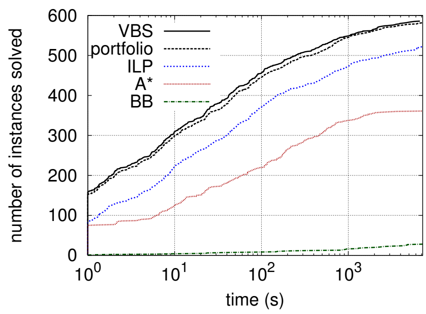
PREDICTORS

We use REP trees to train two predictors:

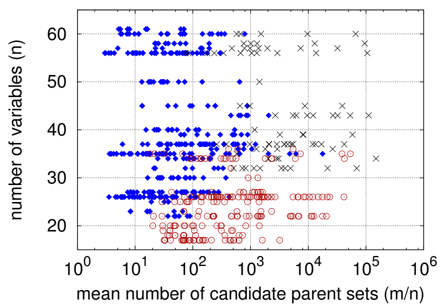
Predictor A: Uses the features n and m.
Predictor B: Uses all features.

PORTFOLIO

Given a new instance, a simple portfolio runs the algorithm predicted to be fastest by predictor A. Comparison to individual algorithms and the Virtual Best Solver that makes perfect predictions:

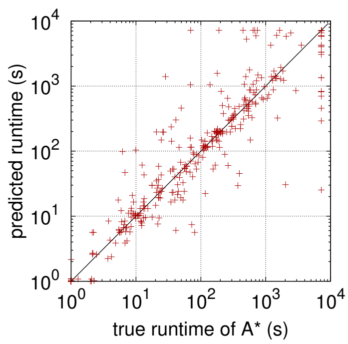


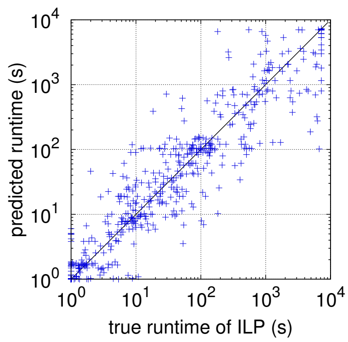
Orthogonality between dominant solvers w.r.t. n and m. Blue instances were solved faster by ILP, red ones by A*:



PREDICTION

Although the simple predictor A already admits an efficient portfolio algorithm, predictor B makes more accurate predictions:





HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY