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

LEARNING CHORDAL MARKOV NETWORKS BY DYNAMIC PROGRAMMING

Kustaa Kangas, Teppo Niinimäki, Mikko Koivisto
Department of Computer Science

Motivation: Structure learning in Markov networks asks for an undirected graph that maximizes a given decomposable scoring function. A special interest is in learning graphs that are *chordal*, since chordal Markov networks (CMNs) of low width admit efficient inference.

Our contribution: We present a dynamic programming algorithm that finds optimal CMNs on n variables in $O(4^n)$ time. Experiments demonstrate our implementation is competitive with recent algorithms based on constraint satisfaction [1] and linear programming [2].

INTRODUCTION

CHORDAL MARKOV NETWORKS

A *Markov network*:

- Undirected graph $\mathcal{G}$ on $V = \{1, \ldots, n\}$
- Represents a joint distribution

$$p(x_1, \ldots, x_n) = \prod_{C \in \mathcal{C}} \psi_C(x_C)\,,$$

where $\mathcal{C}$ is the set of (maximal) cliques of $\mathcal{G}$ and ψ_C are mappings to positive reals.

A *chordal* Markov network:

- Every cycle of length ≥ 4 has an edge between two non-consecutive vertices.
- Admits a clique tree decomposition.

STRUCTURE LEARNING PROBLEM

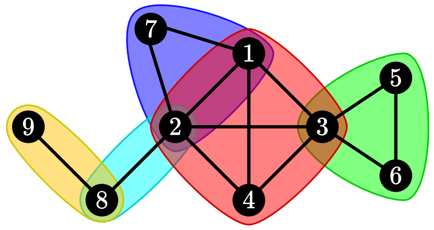
Given *data* D, i.e., samples on $x_1, \ldots, x_n$, a *scoring criterion* $s_D(\mathcal{G})$ measures how well a chordal graph $\mathcal{G}$ fits D.

Common scores (e.g. maximum likelihood, Bayesian Dirichlet) decompose as

$$s_D(\mathcal{G}) = \frac{\prod_{C \in \mathcal{C}} s_D(C)}{\prod_{S \in \mathcal{S}} s_D(S)}\,,$$

where $\mathcal{S}$ is the (multi)set of *separators*.

Separator: intersection of adjacent cliques in a clique tree decomposition of $\mathcal{G}$.



$$\frac{s_D(8,9)s_D(2,8)s_D(1,2,7)s_D(1,2,3,4)s_D(3,5,6)}{s_D(8)s_D(2)s_D(1,2)s_D(3)}$$

Input: $s_D(A)$ for every $A \subseteq V$.

Problem: find a chordal graph $\mathcal{G}$ of best fit, i.e., maximizing $s_D(\mathcal{G})$.

RECURRENCE

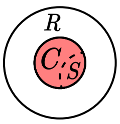
Chordal graphs admit a recursive characterization of the problem.

For $S \subset V$ and $\emptyset \subset R \subseteq V \setminus S$, let $f(S,R)$ be the maximum $s_D(\mathcal{G})$ over chordal $\mathcal{G}$ on $S \cup R$ s.t. S is a proper subset of a clique.

Then, the solution is given by $f(\emptyset, V)$ and

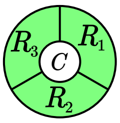
$$f(S,R) = \max_{\substack{S \subset C \subseteq S \cup R \\ \{R_1,\ldots,R_k\} \sqsubset R \setminus C \\ S_1,\ldots,S_k \subset C}} s_D(C) \prod_{i=1}^{k} \frac{f(S_i,R_i)}{s_D(S_i)}\,.$$

Dynamic programming runs in $O(4^n)$ time and $O(3^n)$ space on simplified recurrences:



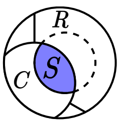
1. Given sets S and R, choose a clique C within R that contains S completely:

$$f(S,R) = \max_{S \subset C \subseteq S \cup R} s_D(C) g(C, R \setminus C)$$



2. Partition the remaining vertices into $R_1, \ldots, R_k$:

$$g(C,U) = \max_{\emptyset \subset R \subseteq U} h(C,R) g(C, U \setminus R)$$



3. For each part R choose a separator S contained in C. Recurse back to step 1:

$$h(C,R) = \max_{S \subset C} f(S,R)/s_D(S)$$

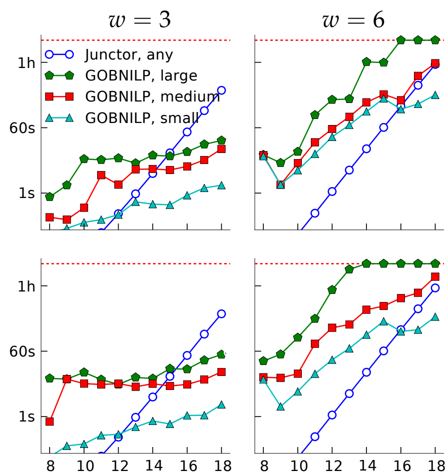
EXPERIMENTS

Compared to a recent constraint satisfaction based algorithm [1], our C++ implementation, `Junctor` (*), appears to be faster by several orders of magnitude.

We also compared against the freely available `GOBNILP` (**) on several instances.

SYNTHETIC INSTANCES

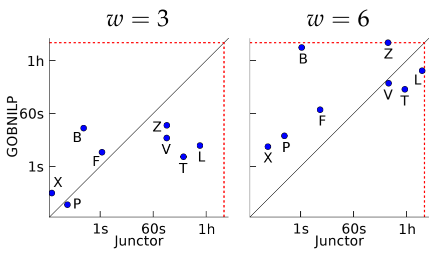
The running times (median for `GOBNILP`) as a function of n, on sparse (top) and dense (bottom) instances with 100 ("small"), 1000 ("medium"), and 10,000 ("large") data samples, bounding clique size by w. The top red line indicates timeout or memout.



BENCHMARK INSTANCES

The running times on the following benchmark instances from the UCI repository.

Dataset	Abbr.	n	Samples
Tic-tac-toe	X	10	958
Poker	P	11	10000
Bridges	B	12	108
Flare	F	13	1066
Zoo	Z	17	101
Voting	V	17	435
Tumor	T	18	339
Lymph	L	19	148



`Junctor` can solve instances of up to 22 variables within a few days for $w = 4$.

(*) `Junctor` is publicly available to download at `www.cs.helsinki.fi/u/jwkangas/junctor/`.

(**) `GOBNILP` by Bartlett and Cussens [2] uses integer linear programming for learning optimal Bayesian networks, but can also be restricted to learning chordal Markov networks.

[1] J. Corander, T. Janhunen, J. Rintanen, H. J. Nyman and J. Pensa. Learning chordal Markov networks by constraint satisfaction. NIPS, pages 1349–1357. 2013.

[2] M. Bartlett and J. Cussens. Advances in Bayesian network learning using integer programming. UAI, pages 182–191. 2013.

HELSINKI INSTITUTE FOR INFORMATION TECHNOLOGY