Approximation Algorithms for Bayesian Network Structure Learning

Aurora Ingebrigtsen (and Adhanet Kiflemariam)

December 4, 2025

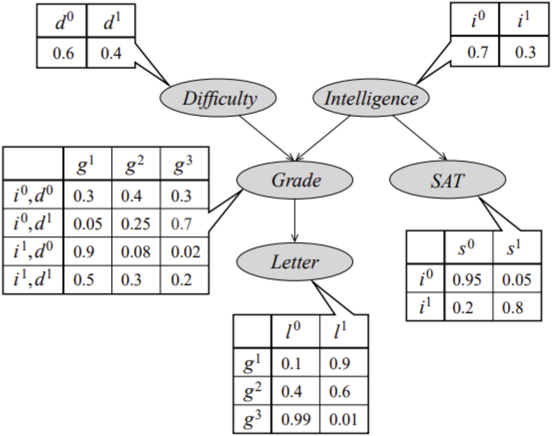
Bayesian networks

- **Bayesian Networks (BNs)** are *probabilistic graphical models* that represent joint probability distributions over a set of random variables.
- The joint distribution factorizes as:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i \mid \mathrm{Pa}(X_i))$$

- Enables efficient reasoning and interpretation of complex probabilistic systems.

Bayesian networks



d^0	d^1
0.6	0.4

i^0	i^1
0.7	0.3

	g^1	g^2	g^3
i^0,d^0	0.3	0.4	0.3
i^0,d^1	0.05	0.25	0.7
i^1,d^0	0.9	0.08	0.02
i^1,d^1	0.5	0.3	0.2

	s^0	s^1
i^0	0.95	0.05
i^1	0.2	0.8

	l^0	l^1
g^1	0.1	0.9
g^2	0.4	0.6
g^3	0.99	0.01

Example: the “Student” Bayesian Network

Bayesian network structure learning

- In many real-world problems, the structure of the network is unknown.
- **Bayesian network structure learning (BNSL)** aims to discover the DAG that best explains the observed data.
- **Score-based methods** assigns each DAG a score based on how well it fits the data and finds the DAG that maximizes the score.
- NP-hard problem

Project Overview

- Implemented two **exact** structure-learning algorithms:
  - Dynamic programming (Silander & Myllymäki)
  - Partial order approach
- Implemented one **approximation** algorithm:
  - Moderately exponential-time approximation
- Main goal:
  - Test how well the approximation performs in practice

Dynamic programming for exact structure learning

- With decomposable scores, the score of a DAG factorizes:

$$\text{Score}(G) = \sum_i s(X_i, \mathrm{Pa}(X_i))$$

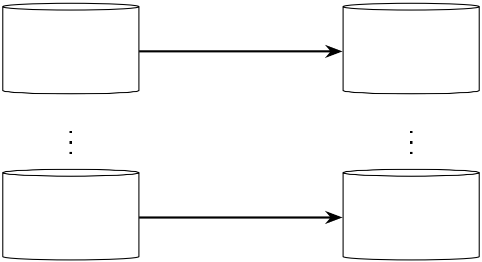
- **Silander & Myllymäki (2006):**
  - Precompute all local scores
  - Compute the best parent set for each variable and candidate set.
  - For every subset $W \subseteq V$, compute the best sink.
  - Recover the optimal variable ordering from sinks.
  - Build the optimal DAG.

Partial Order Approach

- **Parviainen & Koivisto (2013)**: Use precedence constraints (partial orders) to reduce computational complexity
- A **partial order** P on a set M is a relation that is reflexive, antisymmetric, and transitive.
- Let P be a partial order on M. A subset I of M is called an **ideal** of P if $y \in \mathcal{I}$ and $xy \in P$ imply that $x \in \mathcal{I}$
- Use DP over ideals of a partial order P

Partial Order Approach

- How to decide the partial orders?
- **Two-Bucket Scheme**: Partition nodes into front/back buckets
  - User defined parameters m (size of each bucket order), p (number of disjoint bucket orders)
  - Generate partial orders from bucket configurations

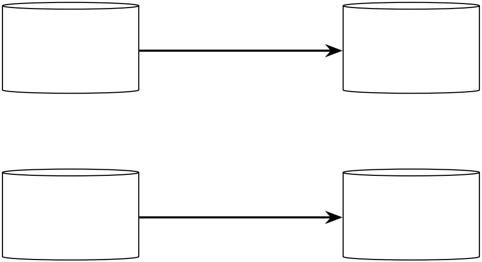


Partial Order Approach

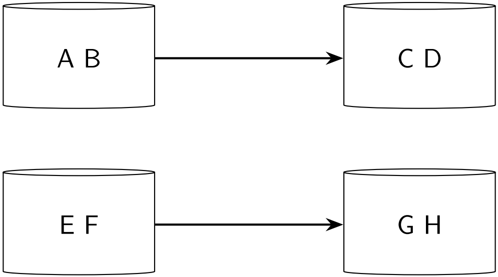
- $V = \{A, B, C, D, E, F, G, H\}$
- $m = 4$
- $p = 2$

$\{A, B, C, D\}$

$\{E, F, G, H\}$



Partial Order Approach



Partial Order Approach

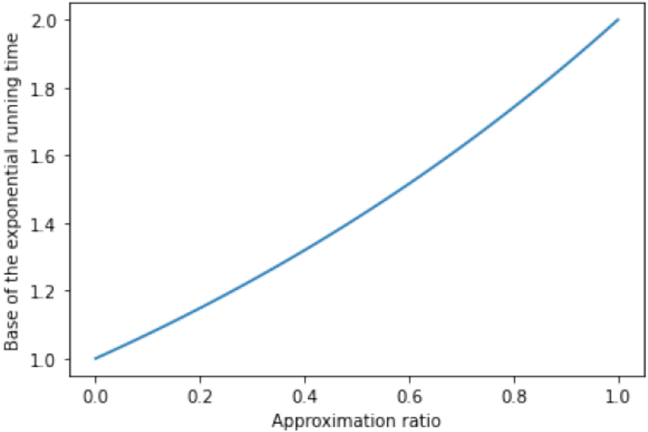
- **Guarantee**: Finds globally optimal DAG
- **Benefits:** space and time trade-off, parallelism, allows exploitation of prior knowledge

Approximation algorithms

- **Zeigler (2008)**: $\frac{1}{m}$-approximation in polynomial time where m is the maximum in-degree.
- **Kundu, Parviainen & Saurabh (2024)**

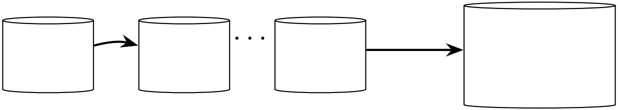
Moderately exponential-time approximation algorithm

- **Kundu, Parviainen & Saurabh (2024)**: Trade off between speed and approximation ratio $\frac{\ell}{k}$ where ℓ and k is user defined parameters.



Moderately exponential-time approximation algorithm

- Partition nodes into k equally sized sets; pick every combination of ℓ sets as the **last bucket** in a bucket order.
  - Find the optimal DAG for each bucket order
- Repeat for all $\binom{k}{\ell}$ choices; return the best DAG found.



Moderately exponential-time approximation algorithm

- Assumes local scores are non-negative:
  - Negative scores can be transformed into non-negative by adding a sufficiently large constant
  - The shift does not affect the output of the algorithms

How to interpret approximation ratios

- Suppose we have 5 variables

How to interpret approximation ratios

- Suppose we have 5 variables
- Local scores in $[-20, -10]$

How to interpret approximation ratios

- Suppose we have 5 variables
- Local score in $[-20, -10]$
- Shift by adding 20 $\rightarrow$ Local score in $[0, 10]$

How to interpret approximation ratios

- Suppose we have 5 variables
- Local score in $[-20, -10]$
- Shift by adding 20 $\rightarrow$ Local score in $[0, 10]$
- Suppose we run the algorithm with approximation ratio $1/2$ and find a DAG with shifted score 10.

How to interpret approximation ratios

- Suppose we have 5 variables
- Local score in $[-20, -10]$
- Shift by adding 20 $\rightarrow$ Local score in $[0, 10]$
- Suppose we run the algorithm with approximation ratio $1/2$ and find a DAG with shifted score 10.
  $\Rightarrow$ The shifted score of the optimal DAG is at most 20.

How to interpret approximation ratios

- Suppose we have 5 variables
- Local score in $[-20, -10]$
- Shift by adding 20 $\rightarrow$ Local score in $[0, 10]$
- Suppose we run the algorithm with approximation ratio $1/2$ and find a DAG with shifted score 10.
  $\Rightarrow$ The shifted score of the optimal DAG is at most 20.
- Invert shifting:
  - The found DAG has score $10 - (5 \cdot 20) = -90$.
  - The optimal DAG has a score at most $20 - (5 \cdot 20) = -80$.

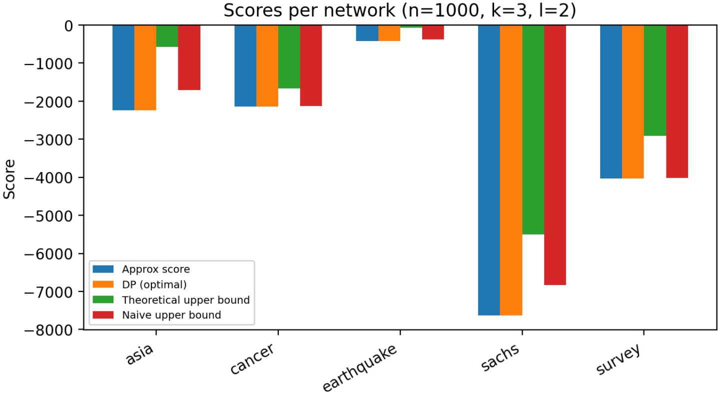
Experimental setup

- Data sizes: 100, 1k, 10k samples
- 3 seeds per configuration
- Approximation parameters: $(k, l) \in \{(5,1),(4,2),(3,2)\}$

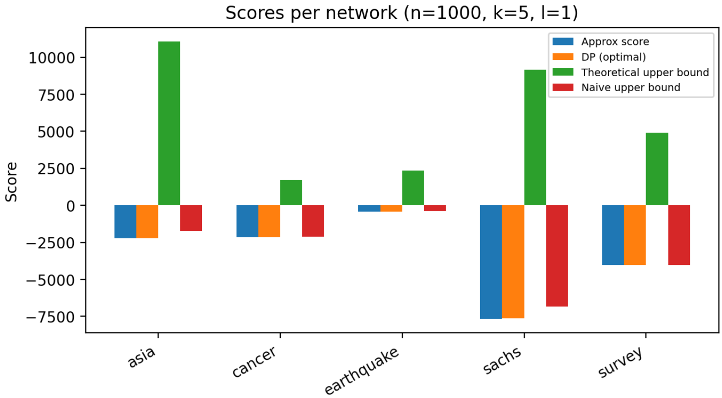
Name	**Nodes**	**Arcs**	**Parameters**
ASIA	8	8	18
CANCER	5	4	10
EARTHQUAKE	5	4	10
SACHS	11	17	178
SURVEY	6	6	21

Table: Networks used for experiments

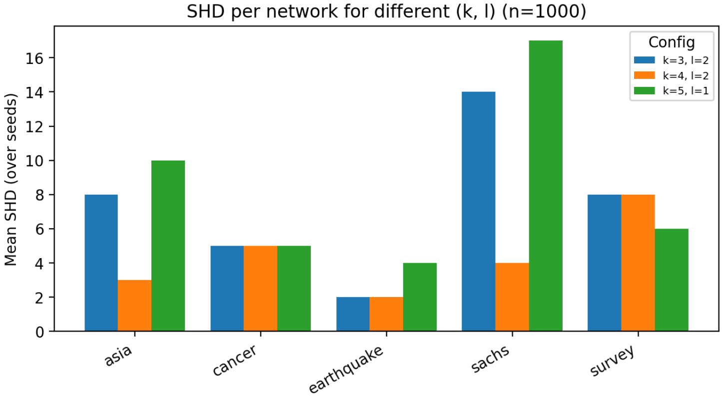
Results: Upper Bounds



Results: Upper Bounds



Results: SHD to true network



Further work

- Medium and large networks
- Running time
- Formalize results: look at trends and patterns

Thank you!