Learning the structure of Bayesian Networks using constraint programming

George Katsirelos
Based on PhD work of Fulya Trösser
cosupervised with Simon de Givry

22/06/2022

Introduction

Bayesian Network

Directed graphical model that captures conditional dependencies

This talk

Learning Bayesian Networks from data

Bayesian Networks

- A directed graph G
- Vertex $\leftrightarrow$ Random Variable
- Conditional probability distributions for each vertex

Bayesian Networks

- A directed graph G
- Vertex $\leftrightarrow$ Random Variable
- Conditional probability distributions for each vertex
- Normalized decomposed representation of a joint probability distribution

Bayesian Networks

- A directed graph G
- Vertex $\leftrightarrow$ Random Variable
- Conditional probability distributions for each vertex
- Normalized decomposed representation of a joint probability distribution
- Absence of an arc $\rightarrow$ conditional independence

Learning BNs

Bayesian Network Learning

Given random variables X and a set of observations, find a **simple** BN of **maximum likelihood**, i.e., that maximizes the probability of generating this data

Learning BNs

Bayesian Network Learning

Given random variables X and a set of observations, find a **simple** BN of **maximum likelihood**, i.e., that maximizes the probability of generating this data

- Once we know the graphical structure, the conditional probability tables are easy to compute

Learning BNs

Bayesian Network Learning

Given random variables X and a set of observations, find a **simple** BN of **maximum likelihood**, i.e., that maximizes the probability of generating this data

- Once we know the graphical structure, the conditional probability tables are easy to compute
- But computing the optimum structure is NP-hard

Applications

- Gene Regulatory Networks
- Risk Analysis
- Image processing

Learning BNs

- From data, precompute *scores*

$$s(x, p)\ \forall x \in X, p \in P(x)$$

  where $P(x) = 2^{X \setminus \{x\}}$
  - p: parent set
- Minimum score maximizes likelihood, minimizes complexity penalty

Learning BNs

Bayesian Network Structure Learning

Given random variables X and a score function over the parent sets of the variables, compute the acyclic DAG which minimizes the sum of the scores over all variables.

Learning BNs

Bayesian Network Structure Learning

Given random variables X and a score function over the parent sets of the variables, compute the acyclic DAG which minimizes the sum of the scores over all variables.

$\Rightarrow$ May not choose each edge individually – must choose among the given parent sets.

A simple observation

Characterization of DAGs

If a directed graph is a DAG then it has a root

Theorem – informal

A directed graph G is a DAG iff every subgraph C has a root

Theorem 1

A directed graph is a DAG iff every subset of vertices $C \subseteq V$ has a vertex with no parents in C

Subsets C are called *clusters*

Constraint Programming - CPBayes

- Variables: $O_1, \ldots, O_n$, to capture the order of the random variables, $P_1, \ldots, P_n$ for parent sets
- Global constraint to ensure acyclicity
- Symmetry breaking and dominance constraints
- Pattern database for lower bounds
- Component caching to avoid failing for the same reason over and over
  - Very efficient cache lookup

Acyclicity global constraint

- Direct application of Theorem 1

Acyclicity global constraint

- Direct application of Theorem 1
- 1. Find a root for the remaining subgraph
  2. If found, add it to *fixed* order
  3. If not found, remaining vertices form a *violated cluster*

Acyclicity global constraint

- Direct application of Theorem 1
- 1. Find a root for the remaining subgraph
  2. If found, add it to *fixed* order
  3. If not found, remaining vertices form a *violated cluster*
- $O(n^2 d)$ to detect unsatisfiability
- GAC by probing $\rightarrow O(n^3 d^2)$

Integer Programming - GOBNILP

- Implicit exponentially large representation
- y_{xp} *family* variables

$$\min \sum_{x \in X, p \in P(x)} score(x, p) y_{xp}$$

$s.t.$

$$\sum_{p \in P(x)} y_{xp} \geq 1 \qquad \forall x \in X$$

Graph is acyclic

Integer Programming - GOBNILP

$$\min \sum_{x \in X, p \in P(x)} score(x, p) y_{xp}$$

$s.t.$

$$\sum_{p \in P(x)} y_{xp} \geq 1 \qquad \forall x \in X$$

$$\sum_{x \in C, p \in P(x), p \cap C = \emptyset} y_{xp} \geq 1 \qquad \forall C \subseteq X \qquad (1)$$

(1): *Cluster* constraints

Integer Programming - GOBNILP

- Cluster constraints are added dynamically in branch-and-cut
- Actual solver incorporates many more types of cuts and specialized branching

+ facets of the polytope

- Need to solve an NP-hard problem to find them
- LP is very dense

+ Very good in practice

Aside – LP tutorial

$$\min \sum_i c_i x_i$$

s.t.

$$\sum_i a_{ij} x_i \geq b_j$$

$$x_i \geq 0$$

LP tutorial – duality

$$
\begin{array}{ll|l}
 & \min \sum_i c_i x_i & \\
\text{s.t.} & & \\
 & \sum_i a_{i1} x_i \geq b_1 & \\
 & \ldots & \\
 & \sum_i a_{im} x_i \geq b_m & \\
 & x_i \geq 0 &
\end{array}
$$

LP tutorial – duality

$$
\begin{array}{lll|l|l}
 & \min \sum_i c_i x_i & & & \\
\text{s.t.} & & & & \\
 & \sum_i a_{i1} x_i \geq b_1 & & y_1 & \\
 & \dots & & \dots & \\
 & \sum_i a_{im} x_i \geq b_m & & y_m & \\
 & x_i \geq 0 & & & y_j \geq 0
\end{array}
$$

$$
\sum_j \sum_i a_{ij} x_i y_j \geq \sum_j b_j y_j \quad \rightarrow
$$

$$
\sum_i (\sum_j a_{ij} y_j) x_i \geq \sum_j b_j y_j
$$

LP tutorial – duality

$$
\begin{array}{lll|l|l}
 & \min \sum_i c_i x_i & & & \\
\text{s.t.} & & & & \\
 & \sum_i a_{i1} x_i \geq b_1 & & y_1 & \\
 & \dots & & \dots & \\
 & \sum_i a_{im} x_i \geq b_m & & y_m & \\
 & x_i \geq 0 & & & y_j \geq 0
\end{array}
$$

$$\sum_j \sum_i a_{ij} x_i y_j \geq \sum_j b_j y_j \quad \rightarrow$$

$$\sum_i (\sum_j a_{ij} y_j) x_i \geq \sum_j b_j y_j \quad \rightarrow$$

$$\sum_j b_j y_j \leq \sum_i (\sum_j a_{ij} y_j) x_i \quad ? \qquad \sum c_i x_i$$

LP tutorial – duality

$$
\begin{array}{l|l|l}
\min \sum_i c_i x_i & & \\
\text{s.t.} & & \\
\sum_i a_{i1} x_i \geq b_1 & y_1 & \sum_j a_{1j} y_j \leq c_1 \\
\dots & \dots & \dots \\
\sum_i a_{im} x_i \geq b_m & y_m & \sum_j a_{nj} y_j \leq c_n \\
x_i \geq 0 & & y_j \geq 0
\end{array}
$$

$$
\sum_j \sum_i a_{ij} x_i y_j \geq \sum_j b_j y_j \quad \rightarrow
$$

$$
\sum_i (\sum_j a_{ij} y_j) x_i \geq \sum_j b_j y_j \quad \rightarrow
$$

$$
\sum_j b_j y_j \leq \sum_i (\sum_j a_{ij} y_j) x_i \quad \leq \quad \sum c_i x_i
$$

LP tutorial – duality

$$
\begin{array}{l|l|l}
\min \sum_i c_i x_i & & \max_j \sum b_j y_j \\
\text{s.t.} & & \\
\sum_i a_{i1} x_i \geq b_1 & y_1 & \sum_j a_{1j} y_j \leq c_1 \\
\ldots & \ldots & \ldots \\
\sum_i a_{im} x_i \geq b_m & y_m & \sum_j a_{nj} y_j \leq c_n \\
x_i \geq 0 & & y_j \geq 0
\end{array}
$$

$$\sum_j \sum_i a_{ij} x_i y_j \geq \sum_j b_j y_j \quad \rightarrow$$

$$\sum_i (\sum_j a_{ij} y_j) x_i \geq \sum_j b_j y_j \quad \rightarrow$$

$$\sum_j b_j y_j \leq \sum_i (\sum_j a_{ij} y_j) x_i \quad \leq \quad \sum c_i x_i$$

LP tutorial – duality

Primal		Dual
$\min \sum_i c_i x_i$		$\max_j \sum b_j y_j$
s.t.		
$\sum_i a_{i1} x_i \geq b_1$	y_1	$\sum_j a_{1j} y_j \leq c_1$
$\ldots$	$\ldots$	$\ldots$
$\sum_i a_{im} x_i \geq b_m$	y_m	$\sum_j a_{nj} y_j \leq c_n$
$x_i \geq 0$		$y_j \geq 0$

$$\sum_j \sum_i a_{ij} x_i y_j \geq \sum_j b_j y_j \quad \rightarrow$$

$$\sum_i (\sum_j a_{ij} y_j) x_i \geq \sum_j b_j y_j \quad \rightarrow$$

$$\sum_j b_j y_j \leq \sum_i (\sum_j a_{ij} y_j) x_i \quad \leq \quad \sum c_i x_i$$

LP tutorial – duality

- $opt(Dual) \leq opt(Primal)$

LP tutorial – duality

- $opt(Dual) \leq opt(Primal)$
- Strong duality $\rightarrow opt(Dual) = opt(Primal)$

LP tutorial – duality

- $opt(Dual) \leq opt(Primal)$
- Strong duality $\rightarrow$ $opt(Dual) = opt(Primal)$
- optimum of LP relaxation $\rightarrow$ lower bound for integer program

LP tutorial – duality

- $opt(Dual) \leq opt(Primal)$
- Strong duality $\rightarrow$ $opt(Dual) = opt(Primal)$
- optimum of LP relaxation $\rightarrow$ lower bound for integer program
- Feasible primal $\rightarrow$ no information

LP tutorial – duality

- $opt(Dual) \leq opt(Primal)$
- Strong duality $\rightarrow$ $opt(Dual) = opt(Primal)$
- optimum of LP relaxation $\rightarrow$ lower bound for integer program
- Feasible primal $\rightarrow$ no information
- Feasible dual $\rightarrow$ lower bound for integer program

LP tutorial – reduced cost

	Primal		Dual
	$\min \sum_i c_i x_i$		$\max_j \sum b_j y_j$
s.t.			
	$\sum_i a_{i1} x_i \geq b_1$	y_1	$\sum_j a_{1j} y_j \leq c_1$
	$\ldots$	$\ldots$	$\ldots$
	$\sum_i a_{im} x_i \geq b_m$	y_m	$\sum_j a_{nj} y_j \leq c_n$
	$x_i \geq 0$		$y_j \geq 0$

LP tutorial – reduced cost

	Primal		Dual
	$\min \sum_i c_i x_i$		$\max_j \sum b_j y_j + y_{m+1}$
s.t.			
	$\sum_i a_{i1} x_i \geq b_1$	y_1	$\sum_j a_{1j} y_j + y_{m+1} \leq c_1$
	$\ldots$	$\ldots$	$\ldots$
	$\sum_i a_{im} x_i \geq b_m$	y_m	$\sum_j a_{nj} y_j \leq c_n$
	$x_1 \geq 1$	y_{m+1}	
	$x_i \geq 0$		$y_j \geq 0$

LP tutorial – reduced cost

	Primal		Dual
	$\min \sum_i c_i x_i$		$\max_j \sum b_j y_j + y_{m+1}$
s.t.			
	$\sum_i a_{i1} x_i \geq b_1$	y_1	$\sum_j a_{1j} y_j + y_{m+1} \leq c_1$
	$\ldots$	$\ldots$	$\ldots$
	$\sum_i a_{im} x_i \geq b_m$	y_m	$\sum_j a_{nj} y_j \leq c_n$
	$x_1 \geq 1$	y_{m+1}	
	$x_i \geq 0$		$y_j \geq 0$

Let $rc_i(y) = c_i - \sum_j a_{ij} y_j$

Adding	$x_i \geq 1$
improves dual bound by	$rc_i(y)$

LP tutorial – reduced cost

	Primal		Dual
	$\min \sum_i c_i x_i$		$\max_j \sum b_j y_j + y_{m+1}$
s.t.			
	$\sum_i a_{i1} x_i \geq b_1$	y_1	$\sum_j a_{1j} y_j + y_{m+1} \leq c_1$
	$\ldots$	$\ldots$	$\ldots$
	$\sum_i a_{im} x_i \geq b_m$	y_m	$\sum_j a_{nj} y_j \leq c_n$
	$x_1 \geq 1$	y_{m+1}	
	$x_i \geq 0$		$y_j \geq 0$

Let $rc_i(y) = c_i - \sum_j a_{ij} y_j$

Adding	$\sum_{i \in S} x_i \geq 1$
improves dual bound by	$\min_{i \in S} rc_i(y)$

Greedy dual LP solving

$$\min \sum_i c_i x_i$$

$$\text{s.t.}$$

$$\sum_i x_i \geq 1$$

$$x_i \geq 0$$

1. Initialize $y_j = 0$ for all j
2. Pick constraint c_j such that all variables $x_i \in c_j$ have $rc_i(y) > 0$
3. Set $y_j = \min_{x_i \in c_j} rc_i(y)$
4. Repeat

Back to BNSL

- The BNSL LP has exactly the right form for our greedy dual LP solver

$$\min \sum_{x \in X, p \in P(x)} score(x, p) y_{xp}$$

$$s.t.$$

$$\sum_{p \in P(x)} y_{xp} \geq 1 \qquad \forall x \in X$$

$$\sum_{x \in C, p \in P(x), p \cap C = \emptyset} y_{xp} \geq 1 \qquad \forall C \subseteq X$$

- But no primal solution $\rightarrow$ cannot use GOBNILP's method for finding violated cluster inequalities
- How do we find relevant inequalities?

Adding cluster inequalities

- Find DAG using only parent sets with reduced cost 0
- If none exists $\rightarrow$ can find cluster C and derive

$$\sum_{x \in C, p \in P(x), p \cap C = \emptyset} y_{xp} \geq 1$$

Adding cluster inequalities

- Find DAG using only parent sets with reduced cost 0
- If none exists $\rightarrow$ can find cluster C and derive

$$\sum_{x \in C, p \in P(x), p \cap C = \emptyset} y_{xp} \geq 1$$

$\rightarrow$ Guaranteed that $rc_{xp} > 0$ for all $x \in C, p \in P(x), p \cap C = \emptyset$

Adding cluster inequalities

- Find DAG using only parent sets with reduced cost 0
- If none exists $\rightarrow$ can find cluster C and derive

$$\sum_{x \in C, p \in P(x), p \cap C = \emptyset} y_{xp} \geq 1$$

$\rightarrow$ Guaranteed that $rc_{xp} > 0$ for all $x \in C, p \in P(x), p \cap C = \emptyset$

$\rightarrow$ Can increase lower bound

GAC on Acyclicity

- Probing GAC propagator
  - $O(n^3 d^2)$
  - Too expensive in practice
- We propose an $O(n^3 d)$ GAC propagator
  - Significant improvement

Domain representation

- Domain values are sets
- Hottest loop:

```
for val :  D(v)
     if (val ⊆ S)
          ...
```

$\Rightarrow$ We use a set-of-sets data structure to speed this up

Decision tree as set of sets

- A directed tree
  - Each node labeled with a random variable
  - Each arc labeled with true or false
- Each path from the root to a leaf describes the characteristic function of a set in the domain
- We mask subtrees to capture current domains/0 reduced cost
- Replace hot loop with DFS in masked decision tree

ELSA solver

- Based on CPBayes
- Extra propagation step: greedily solve cluster LP
- Keep a pool of inequalities discovered during search
- Clusters are eagerly minimized
- GAC on acyclicity constraint
- Decision trees for domain representation

Empirical Performance

- **GOBNILP**: ILP solver (using CPLEX)
- **CPBayes**: CP-based solver
- **ELSA** : our solver based on CPBayes

Data sets from:
UCI Machine Learning Repository,
Bayesian Network Repository,
Bayesian Network Learning and Inference Package.

- **54** medium (less than 64 variables) data sets: BDeu and BIC scores, 1-hour CPU time limit
- **15** large (more than 64 variables) data sets: Only BIC score, max number of parents $= 5$, 10-hour CPU time limit

ELSA vs. CPBayes

Time Limit	Data Set	n	$\sum d$	CPBayes Total Time	ELSA Total Time
1 hour	carpo100_BIC	60	423	76.7 (27.5)	**52.5(0.0)**
	alarm1000_BIC	37	1002	191.1 (159.1)	**37.9(1.9)**
	flag_BDe	29	1324	16.6 (15.6)	**1.3(0.2)**
	wdbc_BIC	31	14613	459.4 (398.0)	**61.7(1.7)**
	kdd.ts	64	43584	†	**1355.2(141.3)**
	steel_BIC	28	93026	1265.6 (1196.1)	**100.6 (45.7)**
	kdd.test	64	152873	†	**1519.6 (48.9)**
	mushroom_BDe	23	438185	167.0 (4.9)	**150.1 (16.7)**
10 hours	bnetflix.ts	100	446406	1086.9 (876.3)	**557.9 (358.4)**
	plants.test	111	520148	†	**35961.7(33712.7)**
	jester.ts	100	531961	†	**7951.4 (7301.6)**
	accidents.ts	100	568160	†	†
	plants.valid	111	684141	†	**19819.2(14547.9)**
	jester.test	100	770950	†	**9644.5 (8742.8)**
	baudio.test	100	1016403	†	**31077.1 (29028.1)**
	bnetflix.test	100	1103968	5794.5 (5486.2)	**1448.8 (1137.7)**
	bnetflix.valid	111	1325818	**998.1 (451.0)**	1476.5 (1041.5)
	accidents.test	100	1425966	†	**8434.1(4723.0)**
	jester.valid	100	1463335	†	**31949.5 (30624.2)**
	accidents.valid	100	1617862	†	†

ELSA vs. GOBNILP

				GOBNILP	ELSA
Time Limit	Data Set	n	$\sum d$	Total Time	Total Time
1 hour	carpo100_BIC	60	423	**0.5**	52.5 (0.0)
	alarm1000_BIC	37	1002	**1.3**	37.9 (1.9)
	flag_BDe	29	1324	4.0	**1.3(0.2)**
	wdbc_BIC	31	14613	86.3	**61.7(1.7)**
	kdd.ts	64	43584	**508.8**	1355.2 (141.3)
	steel_BIC	28	93026	†	**100.6 (45.7)**
	kdd.test	64	152873	3178.0	**1519.6 (48.9)**
	mushroom_BDe	23	438185	†	**150.1 (16.7)**
10 hours	bnetflix.ts	100	446406	†	**557.9 (358.4)**
	plants.test	111	520148	†	**35961.7(33712.7)**
	jester.ts	100	531961	†	**7951.4 (7301.6)**
	accidents.ts	100	568160	**1932.2**	†
	plants.valid	111	684141	†	**19819.2(14547.9)**
	jester.test	100	770950	†	**9644.5 (8742.8)**
	baudio.test	100	1016403	†	**31077.1 (29028.1)**
	bnetflix.test	100	1103968	†	**1448.8 (1137.7)**
	bnetflix.valid	111	1325818	†	**1476.5(1041.5)**
	accidents.test	100	1425966	14453.1	**8434.1(4723.0)**
	jester.valid	100	1463335	†	**31949.5 (30624.2)**
	accidents.valid	100	1617862	**27730.5**	†

ELSA Variants*

				ELSA	ELSA \ GAC
Time Limit	Data Set	n	$\sum d$	Total Time	Total Time
1 hour	carpo100_BIC	60	424	**40,6**	40,7
	alarm1000_BIC	37	1003	**27,8**	28,8
	flag_BDe	29	1325	**0,9**	0,9
	wdbc_BIC	31	14614	**48,9**	49,1
	kdd.ts	64	43584	**1 314,5**	1 405,4
	steel_BIC	28	93027	**98,0**	99,2
	kdd.test	64	152873	**1 475,3**	1 515,9
	mushroom_BDe	23	438186	**135,4**	137,0
10 hours	bnetflix.ts	100	446406	**1 065,1**	1 111,4
	plants.test	69	520148	**18 981,9**	30 791,2
	jester.ts	100	531961	**10 166,0**	14 915,9
	accidents.ts	111	568160	**2 238,7**	2 260,3
	plants.valid	69	684141	**12 347,6**	19 853,1
	jester.test	100	770950	**17 637,8**	21 284,0
	bnetflix.test	100	1103968	8 197,7	**8 057,3**
	bnetflix.valid	100	1325818	**9 282,0**	10 220,5
	accidents.test	111	1425966	**3 661,7**	4 170,1

ELSA variants

				ELSA\DT	ELSA
Time Limit	Data Set	n	$\sum d$	Total Time	Total Time
1 hour	carpo100_BIC	60	423	52.6 (0.1)	**52.5(0.0)**
	alarm1000_BIC	37	1002	**34.4(1.0)**	37.9 (1.9)
	flag_BDe	29	1324	**1.0 (0.2)**	1.3 (0.2)
	wdbc_BIC	31	14613	**56.0 (2.4)**	61.7 (1.7)
	kdd.ts	64	43584	1452.3 (274.6)	**1355.2(141.3)**
	steel_BIC	28	93026	124.2 (71.8)	**100.6 (45.7)**
	kdd.test	64	152873	1594.3 (224.4)	**1519.6 (48.9)**
	mushroom_BDe	23	438185	182.6 (58.9)	**150.1 (16.7)**
10 hours	bnetflix.ts	100	446406	2103.1 (1900.9)	**557.9 (358.4)**
	plants.test	111	520148	**28049.6 (26312.9)**	35961.7 (33712.7)
	jester.ts	100	531961	21550.5 (21003.7)	**7951.4 (7301.6)**
	accidents.ts	100	568160	**2302.2 (930.0)**	†
	plants.valid	111	684141	**17801.6 (14080.2)**	19819.2 (14547.9)
	jester.test	100	770950	30186.8 (29455.0)	**9644.5 (8742.8)**
	baudio.test	100	1016403	†	**31077.1 (29028.1)**
	bnetflix.test	100	1103968	10333.1 (10096.5)	**1448.8 (1137.7)**
	bnetflix.valid	111	1325818	10871.7 (10527.7)	**1476.5 (1041.5)**
	accidents.test	100	1425966	**3641.7 (680.7)**	8434.1 (4723.0)
	jester.valid	100	1463335	†	**31949.5 (30624.2)**
	accidents.valid	100	1617862	†	†

Another aside: Connection to WCSP/MaxSAT/CP

- The set of variables with reduced cost 0 correspond to values that are kept in $Bool(P)$ in VAC
  - Also called *active* values
  - But VAC also propagates inequalities and may discover chains of inference
  - Exact LP solving may be even stronger
- MaxSAT cost-aware disjoint cores are almost exactly the same idea

Another aside: Connection to WCSP/MaxSAT/CP

- The set of variables with reduced cost 0 correspond to values that are kept in $Bool(P)$ in VAC
  - Also called *active* values
  - But VAC also propagates inequalities and may discover chains of inference
  - Exact LP solving may be even stronger
- MaxSAT cost-aware disjoint cores are almost exactly the same idea
- General method for handling global constraints

Insights

- Explanation-based bounding for CP
- General caching

Conclusions

- New algorithmic insights
  - Partially inspired by WCSP and MaxSAT
- Better tradeoff of speed vs inference than existing solvers
- All previous techniques were useful

Limitations

- Score computation
- LP solving
- Communication between caching/LP/pattern DB

Q?