Structured Set Variable Domains in Bayesian Network Structure Learning using constraint programming

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

01/06/2023

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

Score-based approach

- 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
  - BIC, BDe, ...

Score-based approach

- 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
  - BIC, BDe, ...
- Large number of candidate parentsets

Score-based approach

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.

Score-based approach

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.

Constraint Program for BNSL

- *Decision Variables*: one *set variable* per vertex
- Goal: Assign a parent-set to each variable minimizing the score
- Such that: the resulting graph is acyclic

  $\Rightarrow$ *Global constraint*

Solving Constraint Programs

- Solved with branch-and-bound
- Strong bounds
  Given the current choice of parentsets, what is the best score I could possibly achieve?
  $\Rightarrow$ Relaxations
- Efficient propagation
  Given the current choice of parentsets, what choices can I prove are impossible or suboptimal?
  $\Rightarrow$ Feeds back into bound computation

The problem

- We have loops like these with set domains:

for $x \in P$ **do**
  for $S \in D(x)$ **do**
    if $S \subseteq W$ **then**
      something cheap ;

- We use decision trees to represent the domains to accelerate finding whether such an S exists

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*

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

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
- Step 1 implementation: Find variable with parent-set included in the order

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
- Step 1 implementation: Find variable with parent-set included in the order
- Also used as a subroutine in an algorithm that improves bounds

Domain representation

- Domain values are sets
- Hottest loop:

```
for S in D(v)
     if (S ⊆ W)
           ...
```

$\Rightarrow$ 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
  - CPBayes: basic CP model + many optimizations
- Stronger propagation on acyclicity
- Stronger bounds
- 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

				CPBayes	ELSA
Time Limit	Data Set	n	$\sum d$	Total Time	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**	†

The effect of decision trees

				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	†	†

Conclusions

- New algorithmic insights
- Better tradeoff of speed vs inference than existing solvers
- All previous techniques were useful

Future directions

- Better exploitation of domain structure
- Lazy Score computation

Q?