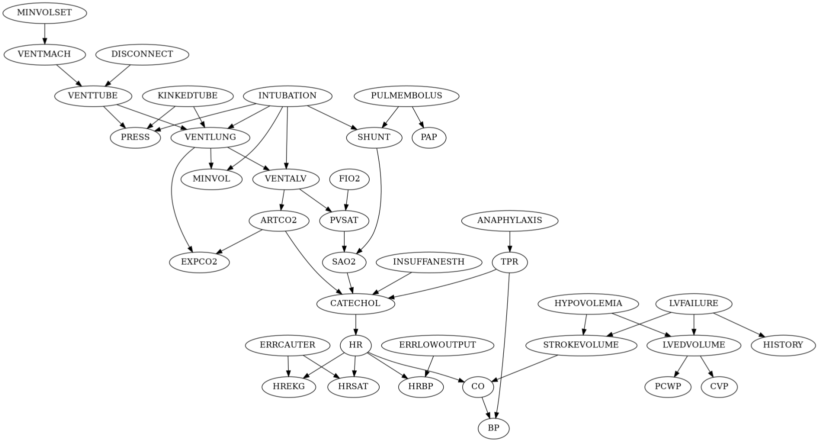
# Branch-price-and-cut for Bayesian network structure learning

James Cussens, University of Bristol

SCIP 20th anniversary workshop
4 November 2022, ZIB

# The Alarm Bayesian network



# BNSL ILP formulation

$$\textsc{Minimise} \sum_{\substack{i \in P \\ J \subseteq P \setminus \{i\}}} c_{i \leftarrow J} x_{i \leftarrow J}$$

SUBJECT TO:

$$\sum_{J \subseteq P \setminus \{i\}} x_{i \leftarrow J} = 1 \qquad i \in P$$

$$\sum_{i \in C} \sum_{\substack{J \subseteq P \setminus \{i\} \\ J \cap C \neq \emptyset}} x_{i \leftarrow J} \leq |C| - 1 \qquad C \subseteq P, |C| \geq 2$$

$$x_{i \leftarrow J} \in \{0, 1\}, i \in P, J \subseteq P \setminus \{i\}$$

# Pricing and cutting

- We have exponentially many decision variables which motivates pricing.
- We have exponentially many constraints which motivates a cutting plane approach.
- We could have only a quadratic number of acyclicity constraints but we choose to use the exponentially many *cluster constraints* since they are known to be facet-defining.
- In fact, any connected matroid defined on any subset of $P$ defines a facet.[1]

[1] Milan Studený. "How matroids occur in the context of learning Bayesian network structure". In: *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence (UAI 2015)*. Ed. by Marina Meila and Tom Heskes. AUAI Press, 2015, pp. 832–841.

# BNSL linear relaxation of combinatorial relaxation

$$\textsc{Minimise} \sum_{\substack{i \in P \\ J \subseteq P \setminus \{i\}}} c_{i \leftarrow J} x_{i \leftarrow J}$$

SUBJECT TO:

$$\sum_{J \subseteq P \setminus \{i\}} x_{i \leftarrow J} = 1 \qquad i \in P$$

$$\sum_{i \in C} \sum_{\substack{J \subseteq P \setminus \{i\} \\ J \cap C \neq \emptyset}} x_{i \leftarrow J} \leq |C| - 1 \qquad C \subseteq P, C \in \mathcal{C}$$

$$x_{i \leftarrow J} \in [0, 1], i \in P, J \subseteq P \setminus \{i\}$$

# A generic pricing problem

- Let $\lambda_i^*$ and $\lambda_C^*$ ($C \in \mathcal{C}$) be the dual values for the equations and cluster constraints, respectively.
- One natural approach is to look for a new family variable $x_{i \leftarrow J}$ with minimal negative reduced cost for each $i \in P$:

$$\begin{array}{ll} \textsc{Minimise}_J & z = c_{i \leftarrow J} - \lambda_i^* - \sum_{\substack{C \in \mathcal{C}, i \in C \\ C \cap J \neq \emptyset}} \lambda_C^* \\ \textsc{subject to} & z < 0 \end{array}$$

- Note that since $\lambda_C^* \leq 0$, it is harder for big parents sets $J$ to have negative reduced cost, since they 'make a cycle more likely'.

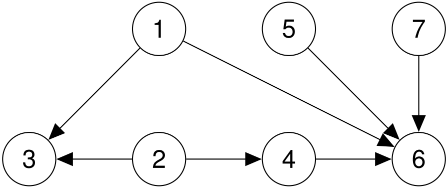
# Pricing for $\ell_0$ penalised Gaussian BNs

$$\text{MINIMISE}_J \quad z = n \log \sigma^2_{i \leftarrow J} + \Lambda^2 |J| - \sum_{\substack{C \in \mathcal{C}, i \in C \\ C \cap J \neq \emptyset}} \lambda^*_C$$

$$\text{SUBJECT TO} \quad z < \lambda^*_i$$

- Learning a $\ell_0$ penalised Gaussian BNs amounts to finding a 'good' $\ell_0$ linear regression model for each $i \in P$ without allowing cycles.
- We add a cycle-penalty to the $\ell_0$ penalty.
- $\sigma^2_{i \leftarrow J}$ denotes minimal squared error when predicting $i$ using predictors $J$ in a linear regression model. But this not a convex problem since we have $\log \sigma^2_{i \leftarrow J}$ rather than $\sigma^2_{i \leftarrow J}$.

# Example of price-and-cut




| LP | $\|\mathcal{C}\|$ | $\|\mathcal{V}\|$ | Rounds | Obj |
|---|---|---|---|---|
| 1 | 0 | 14 | 7,0 | -20954 |
| 2 | 20 | 35 | 4,4,5,4,3,1,0 | -38006 |
| 3 | 40 | 50 | 4,5,3,2,1,0 | -43158 |
| 4 | 60 | 52 | 2,0 | -45301 |

- Using pricing we end up with 52 ‘family’ variables rather than $7 \times 2^6 = 448$.

# Branch-price-and-cut

- We branch not on family variables but *arrow indicator variables* for a more balanced search tree.
- Instead of the constraint $x_{i \leftarrow j} = \sum_{J:j \in J} x_{i \leftarrow J}$,
- we post two *set partitioning constraints* $\sum_{J:j \in J} x_{i \leftarrow J} + \neg x_{i \leftarrow j} = 1$ and $\sum_{J:j \notin J} x_{i \leftarrow J} + x_{i \leftarrow j} = 1$.
- This allows SCIP to perform the desired propagations even when new $x_{i \leftarrow J}$ may be priced in.
- It is not too hard to alter the pricing algorithm to be consistent with the set of obligatory and forbidden arrows in any node.

# Partial order variables

- To facilitate propagation we also create *partial order variables* $x_{i \leftsquigarrow j}$.
- $x_{i \leftarrow j} + \neg x_{i \leftsquigarrow j} \leq 1$
- $x_{i \leftsquigarrow j} + x_{j \leftsquigarrow i} \leq 1$
- $x_{i \leftsquigarrow j} + x_{j \leftsquigarrow k} - x_{i \leftsquigarrow k} \leq 1.$

# What is/isn't in the LP

- Partial order variables and their constraints are not in the LP. An adaptation of Marc Pfetsch's LOP constraint handler is used for them.
- The constraints $\sum_{J:j \in J} x_{i \leftarrow J} + \neg x_{i \leftarrow j} = 1$ are in the LP and so have associated dual values $\lambda^*_{i \leftarrow j}$.

$$\text{MINIMISE}_J \quad z = c_{i \leftarrow J} - \sum_{j \in J} \lambda^*_{i \leftarrow j} - \sum_{\substack{C \in \mathcal{C}, i \in C \\ C \cap J \neq \emptyset}} \lambda^*_C$$

$$\text{SUBJECT TO} \quad z < \lambda^*_i$$

# Pricing via nonlinear optimisation

MINIMISE $$z = nx_{\log\sigma^2} + \sum_{j \in P \setminus \{i\}} (\Lambda^2 - \lambda^*_{i \leftarrow j}) y_j - \sum_{C \in \mathcal{C}} \lambda^*_C y_C$$

SUBJECT TO

$$\sum_{j \in P \setminus \{i\}} \gamma_j^2 + c \leq nx_{\sigma^2}$$

$$x_{\log\sigma^2} = \log x_{\sigma^2}$$

$$\gamma = S^{1/2}\beta - S^{-1/2}\boldsymbol{X}_{-i}^\top \boldsymbol{X}_i$$

$$(\beta_j, 1 - y_j) : \text{SOS-1} \qquad j \in P \setminus \{i\}$$

$$y_C = \bigvee_{j \in C} y_j \qquad C \in \mathcal{C}$$

$$z < \lambda_i^*$$

$$x_{\sigma^2} \in \mathbb{R}_+ \; x_{\log\sigma^2}, \gamma_j, \beta_j \in \mathbb{R} \; y_j, y_C \in \{0, 1\}$$

# How to interleave pricing and cutting?

- In the standard approach to cut-and-price each LP is solved to optimality (using pricing), and only once it is solved do we look for cuts to get a better LP (i.e. a tighter linear relaxation).
- But why bother solving an LP to optimality if it will soon be replaced by a better one? Also tight LPs make the pricing problem easier.
- So I add the option to only start pricing once we can find no more cuts.
- Thanks to Stephen Maher on ideas on how to 'trick' SCIP into doing delayed pricing!

# Creating initial parent sets

- A good idea to create, for each ‘child’ $i$ all necessary parent sets $J$ up to some size $k$.
- Sometimes one can establish that no bigger parent sets are needed for a particular child and so we can avoid futilely attempting to price in new parent sets.
- Also useful to find the parent set that would be the best for each child, if we did not have to worry about cycles.

# Does it work?

At time of writing my implementation is not entirely bug-free, but usually gives the correct answers - eventually!

| $k$ | NVars | Solving Time | Pricing time |
|---|---|---|---|
| 1 | 321 | 478 | 475 |
| 2 | 316 | 463 | 459 |
| 3 | 283 | 151 | 150 |
| 4 | 282 | 80 | 77 |
| 5 | - | - | - |
| 6 | 276 | 2 | 0 |

Table: Solving times for PRICEBNLEARN on the small `gaussian.test` dataset using BIC $\ell_0$ penalised log squared error. NVars indicates the number of IP variables in the problem at the point it is solved.

# With an easier objective . . .

| $k$ | NVars | Solving Time | Pricing time |
|---|---|---|---|
| 1 | 151 | 12.7 | 12.6 |
| 2 | 194 | 11.8 | 11.7 |
| 3 | 243 | 9.2 | 9.1 |
| 4 | 261 | 3.1 | 3.0 |
| 5 | 262 | 0.1 | 0 |
| 6 | 262 | 0.1 | 0 |

Table: Solving times for PRICEBNLEARN on the small `gaussian.test` dataset using BIC $\ell_0$ penalised squared error. NVars indicates the number of IP variables in the problem at the point it is solved.

# The benefits of delayed pricing

On a bigger problem with pricing as normal:

| $k$ | NVars | Solving Time | Pricing time |
|---|---|---|---|
| 3 | $\geq 3022$ | $> 2457$ | $> 2457$ |
| 6 | 19386 | 2206 | 2192 |

With delayed pricing:

| $k$ | NVars | Solving Time | Pricing time |
|---|---|---|---|
| 3 | 3035 | 1326 | 1324 |
| 6 | 19386 | 351 | 335 |

# General points

- There are big opportunities for MIP methods in machine learning.
- For example, pricing is needed for MIP learning of causal models where latent variables are allowed.[2].
- Naturally, we need a pricer that is fast (or infrequently called) for this approach to be a practical option.
- The interplay between pricing, cutting and branching requires careful consideration.

---

[2] Rui Chen, Sanjeeb Dash, and Tian Gao. "Integer Programming for Causal Structure Learning in the Presence of Latent Variables". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 18–24 Jul 2021, pp. 1550–1560. URL: https://proceedings.mlr.press/v139/chen21c.html.