Proceedings of Machine Learning Research vol 213:1–18, 2023 2nd Conference on Causal Learning and Reasoning

Branch-Price-and-Cut for Causal Discovery

James Cussens JAMES.CUSSENS@BRISTOL.AC.UK
Dept of Computer Science, University of Bristol, Bristol, UK

Editors: Mihaela van der Schaar, Dominik Janzing and Cheng Zhang

Abstract

We show how to extend the integer programming (IP) approach to score-based causal discovery by including *pricing*. Pricing allows the addition of new IP variables during solving, rather than requiring them all to be present initially. The dual values of acyclicity constraints allow this addition to be done in a principled way. We have extended the GOBNILP algorithm to effect a branch-price-and-cut method for DAG learning. Empirical results show that implementing a *delayed pricing* approach can be beneficial. The current pricing algorithm in GOBNILP is slow, so further work on fast pricing is required.

Keywords: Causal discovery, integer linear programming, pricing

1. Introduction

We consider score-based approaches to causal discovery. In the score-based approach each causal model has a score—determined by the observed data, and perhaps prior knowledge—and the goal is to find whichever causal model has the best score. Typical scores include posterior probability and log-likelihood with some suitable complexity penalty. The problem of causal discovery thus reduces to an optimisation problem.

The principal problem with this approach is that the resulting optimisation problem can be NP-hard. For example, Chickering et al. (2004) showed that Bayesian network learning (i.e. causal discovery assuming causal sufficiency) is NP-hard even with 'large' samples. Assuming $P \neq NP$ this means that solving the causal discovery optimisation problem exactly (i.e. finding a provably globally optimal solution) is impractical in the general case. As a result most approaches to causal discovery are inexact (sometimes called *heuristic*): a search for high-scoring causal models is conducted and the best-scoring model is returned, but there is no guarantee that a better-scoring model does not exist. Many inexact methods use local search.

In addition to the integer programming score-based approach considered here, there are very many approaches to causal discovery. For a good review please consult Drton and Maathuis (2017) which covers both score-based and constraint-based learning (where conditional independence constraints are inferred from data and graphical models consistent with these constraints are found).

The great advantage of resorting to inexact methods is that it is often possible in practice to find high-scoring models quickly even when learning from large datasets. However, when exact learning is possible it is clearly preferable; the goal of this paper is to explore whether a particular method—*pricing*—can extend the applicability of exact learning.

$$\text{MINIMISE} \quad \sum_{\substack{i \in P \\ J \subseteq P \setminus \{i\}}} c_{i \leftarrow J} x_{i \leftarrow J} \tag{1}$$

SUBJECT TO:

$$\sum_{J \subseteq P \setminus \{i\}} x_{i \leftarrow J} = 1 \qquad i \in P \tag{2}$$

$$\sum_{i \in C} \sum_{\substack{J \subseteq P \setminus \{i\} \\ J \cap C \neq \emptyset}} x_{i \leftarrow J} \leq |C| - 1 \qquad C \subseteq P, |C| \geq 2 \tag{3}$$

$$x_{i \leftarrow J} \in \{0, 1\}, i \in P, J \subseteq P \setminus \{i\} \tag{4}$$

Figure 1: Integer program for score-based BNSL with a decomposable cost

2. Bayesian network structure learning using integer programming

We will focus on learning DAGs under the assumption of causal sufficiency, postponing discussion of other casual discovery tasks until Section 7. This problem is often called *Bayesian network structure learning* and we will abbreviate it to BNSL throughout. We will formulate BNSL as a minimisation problem, so the goal is to find a DAG of minimal cost, rather than maximal score. Let P be the random variables observed in the data which correspond to the vertices of candidate DAGs. We will assume that the cost is *decomposable*. A decomposable cost is one which is a sum of *local costs*, one for each BN variable, where the local cost for each BN variable $i \in P$ is determined by the choice of *parents* $J \subseteq P \setminus \{i\}$ for that variable. BN variable j is a parent of BN variable i iff there is a directed edge $i \leftarrow j$ in the DAG in which case i is called a *child* of j.

Let $x_{i \leftarrow J}$ be the binary variable indicating that BN variable i has parent set J. The $x_{i \leftarrow J}$ will be called *family variables* since they indicate the choice of parents for a particular BN (child) variable. Clearly any directed graph has a unique encoding as a zero-one vector indexed by the family variables. Denote the local cost for i having parents J by $c_{i \leftarrow J}$. Each $c_{i \leftarrow J}$ can be computed from the data we are learning from. If a cost is decomposable it follows that the cost for any DAG is $\sum c_{i \leftarrow J} x_{i \leftarrow J}$ where the $x_{i \leftarrow J}$ encode the DAG.

Let x denote both the vector of all $x_{i \leftarrow J}$ family variables and also particular values of x—where the context will make clear which is meant. Finding an x which minimises $\sum c_{i \leftarrow J} x_{i \leftarrow J}$ does not solve the BNSL problem since typically such an x will not represent a DAG. It is necessary to add *constraints* to ensure that the only feasible values of x are those which represent DAGs. One suitable set of constraints is given in Fig 1 where score-based BNSL with a decomposable score is represented as an *integer linear program*, or *integer program (IP)* for short. Fig 1 represents an IP since the objective function is linear and constraints are either linear (2,3) or are integrality constraints (4). For a good introduction to the theory and practice of integer programming Wolsey (1998) is recommended.

In the IP in Fig 1 the equations (2) ensure that exactly one parent set J is selected for each BN variable i, so that each feasible x at least represents a directed graph. Next note that for $C \subseteq P, |C| \geq 2$ if there is a cycle with vertices C then each $i \in C$ has a parent in C and we

have $\sum_{i \in C} \sum_{J \subseteq P \setminus \{i\}, J \cap C \neq \emptyset} x_{i \leftarrow J} = |C|$, so the inequalities (3) are sufficient to ensure that only *acyclic* directed graphs are feasible. These inequalities (in a different formulation) were introduced by Jaakkola et al. (2010) under the name *cluster constraints*.

3. Solving the BNSL linear relaxation

Exact optimisation has two tasks. If we are, as here, minimising, then just like inexact optimisation it aims to find solutions with low cost. But unlike inexact optimisation it also computes a lower bound on the objective of an optimal solution. When we have a solution whose objective value equals this lower bound then we know that it is a globally optimal solution.

The best solution found at any point in the solving process is known as the *incumbent* and in modern IP solvers incumbents are found by all sorts of heuristic algorithms. The current version of the IP solver SCIP (Bestuzheva et al., 2021) has no fewer than 58 heuristic algorithms available for this purpose! Note that, in principle, *any* DAG learning algorithm could be used by an IP solver to provide an initial incumbent DAG, the task of the IP solver being then to search for yet better incumbents and to compute lower bounds in the hope of establishing optimality.

Lower bounds on optimal objective values are provided by solving *relaxations* of the original problem. A relaxation is produced by removing some constraints and solving the resulting problem to optimality. The objective value of an optimal solution of the relaxed problem provides the desired lower bound: it will always be no worse (i.e. not greater than) than the objective value of an optimal solution to the original problem, precisely because the former is a less constrained version of the latter. It is, of course, crucial that this optimal solving can be performed quickly. In addition, it is preferable if the relaxation is *tight*, so that the lower bound it provides is close to the true optimal objective value.

For an IP problem the standard approach is to solve its *linear relaxation* where integrality constraints are removed, thus producing a *linear program (LP)*. LPs can be solved to optimality in polynomial time, although in practice they are quickly solved by the simplex algorithm which has worst-case exponential time. The linear relaxation of our BNSL IP (Fig 1) just replaces $x_{i \leftarrow J} \in \{0, 1\}$ with $x_{i \leftarrow J} \in [0, 1]$.

However, solving this linear relaxation is non-trivial since there are $2^p - p - 1$ cluster constraints (where $p = |P|$) and $p2^{p-1}$ family variables. So unless p is small, constructing and solving an LP with all family variables and all cluster constraints present is impractical. Cussens (2011) addressed the too-many-cluster-constraints problem by adding these constraints as *cutting planes*. An initial LP is created with only a small set $\mathcal{C}$ of cluster constraints. This LP is solved and then a search is conducted for cutting planes—cluster constraints not satisfied by the solution to this LP. The set $\mathcal{C}$ of cluster constraints is then updated to include these cutting planes and the resulting new LP solved, and a new search for cutting planes is conducted. This iterative approach is continued until no cutting planes can be found or the linear relaxation provides a sufficiently good bound. Note that each cluster constraint $C \in \mathcal{C}$ defines a *facet*, it is a maximally tight inequality of the convex hull of zero-one vectors encoding DAGs (Cussens et al., 2017a,b). This is why they provide tight linear relaxations and why it is worth the effort to find them even though cycles can be ruled out with merely a quadratic number of linear constraints (Cussens, 2010).

Although the problem of finding these cutting planes is NP-hard (Cussens et al., 2017b), this has proved a reasonable approach to producing tight linear relaxations *if there are not too many*

family variables. The key contribution of this paper is to investigate the use of *pricing* to iteratively introduce family variables, when adding all of them initially is impractical.

3.1. Pricing-in variables

To understand how pricing can be inserted into the cutting plane approach for BNSL one needs to consider the *dual* of the linear program that is solved at each stage. Let $\mathcal{C}$ be the current set of cluster constraints. Reformulate the inequalities (3) as the lower bounds: $-\sum_{i \in C} \sum_{\substack{J \subseteq P \setminus \{i\} \\ J \cap C \neq \emptyset}} x_{i \leftarrow J} \geq -(|C| - 1)$. Consider multiplying the equations (2) by coefficients λ_i and the lower-bound version of the inequalities (3) by λ_C where $C \in \mathcal{C}, \lambda_C \geq 0$, then the sum of resulting inequalities is a valid inequality

$$\sum_{\substack{i \in P \\ J \subseteq P \setminus \{i\}}} d_{i \leftarrow J} x_{i \leftarrow J} \geq \sum_{i \in P} \lambda_i - \sum_{C \in \mathcal{C}} \lambda_C (|C| - 1)$$

where $d_{i \leftarrow J} = \lambda_i - \sum_{C \in \mathcal{C}, i \in C, C \cap J \neq \emptyset} \lambda_C$. (Each equation (2) can be viewed as the pair of inequalities $\sum_{J \subseteq P \setminus \{i\}} x_{i \leftarrow J} \geq 1$ and $\sum_{J \subseteq P \setminus \{i\}} -x_{i \leftarrow J} \geq -1$, with associated non-negative multipliers λ_i^+ and λ_i^-, and where $\lambda_i = \lambda_i^+ - \lambda_i^-$.) If $\forall i, J : c_{i \leftarrow J} \geq d_{i \leftarrow J}$ then $\sum_{i \in P} \lambda_i - \sum_{C \in \mathcal{C}} \lambda_C (|C| - 1)$ is a lower bound on the objective. The dual of the linear program is just the problem of finding values for the λ_i and the λ_C which lead to the greatest such lower bound by maximising $\sum_{i \in P} \lambda_i - \sum_{C \in \mathcal{C}} \lambda_C (|C| - 1)$ subject to $\forall i, J : c_{i \leftarrow J} \geq d_{i \leftarrow J}, \forall C \in \mathcal{C}, \lambda_C \geq 0$. By the strong duality of linear programming the objective values of the solutions to the dual and the original (*primal*) linear program are equal (assuming both have a solution).

Let λ^* be the solution to the dual linear program. Note that any reasonable LP solver will compute λ^* in the course of solving the primal LP, a separate optimisation process is not required. Suppose that instead of including all family variables, only a subset $\mathcal{V}$ are in the LP, thus most probably not allowing an optimal solution to the (primal) LP. Let $x_{i \leftarrow J}$ be a family variable not in $\mathcal{V}$ whose local cost $c_{i \leftarrow J}$ is low enough so that $c_{i \leftarrow J} < d^*_{i \leftarrow J} = \lambda^*_i - \sum_{C \in \mathcal{C}, i \in C, C \cap J \neq \emptyset} \lambda^*_C$. Then if one were to add $x_{i \leftarrow J}$ to $\mathcal{V}$, the new dual linear program thus produced would have an additional constraint $c_{i \leftarrow J} \geq d_{i \leftarrow J} = \lambda_i - \sum_{C \in \mathcal{C}, i \in C, C \cap J \neq \emptyset} \lambda_C$ which the earlier solution λ^* does not satisfy. The addition of the variable $x_{i \leftarrow J}$ to the primal LP thus adds a cutting plane to the dual LP which renders the dual solution λ^* no longer feasible.

Pricing is the process of finding and adding new variables $x_{i \leftarrow J}$ which render the current dual solution infeasible. Typically, adding such variables to the primal LP will result in an LP solution with a lower objective value. If no such variables can be found then it follows that the LP has been solved to optimality *even though only a subset of IP variables have been used*. All potential variables which are not included in the LP are implicitly set to 0 in this optimal solution.

The condition $c_{i \leftarrow J} < \lambda^*_i - \sum_{C \in \mathcal{C}, i \in C, C \cap J \neq \emptyset} \lambda^*_C$ is equivalent to $c_{i \leftarrow J} - \lambda^*_i + \sum_{C \in \mathcal{C}, i \in C, C \cap J \neq \emptyset} \lambda^*_C < 0$. $c_{i \leftarrow J} - \lambda^*_i + \sum_{C \in \mathcal{C}, i \in C, C \cap J \neq \emptyset} \lambda^*_C$ is called the *reduced cost* for $x_{i \leftarrow J}$. The goal of pricing is to find variables which, given the current dual values λ^*_i, λ^*_C, have negative reduced cost. If such variables are found they are added to $\mathcal{V}$ producing a new LP whose dual is solved leading to new dual values and a new pricing problem. Note that a new variable must be added to all constraints where it has non-zero coefficient. So $x_{i \leftarrow J}$ must be added to the constraint $\sum_{J \subseteq P \setminus \{i\}} x_{i \leftarrow J} = 1$ as well as to any cluster constraint whose cluster C has $i \in C, J \cap C \neq \emptyset$. In this way new variables are added to the problem in *pricing rounds*.

The least any pricing method must deliver is, on each pricing round, to find at least one variable with negative reduced cost or establish that no such variable exists. One simple pricing strategy is,

$$\begin{aligned} &\text{MINIMISE}_J && z = c_{i\leftarrow J} - \lambda_i^* + \sum_{\substack{C\in\mathcal{C}, i\in C \\ C\cap J\neq\emptyset}} \lambda_C^* \\ &\text{SUBJECT TO} && z < 0 \end{aligned}$$

Figure 2: Pricing problem for BN variable i where $\mathcal{C}$ is the current set of clusters and λ_i^* and λ_C^* are the current dual values for the constraints.

$$\begin{aligned} &\text{MINIMISE}_J && z = n\log\sigma^2_{i\leftarrow J} + \Lambda^2|J| + \sum_{\substack{C\in\mathcal{C}, i\in C \\ C\cap J\neq\emptyset}} \lambda_C^* \\ &\text{SUBJECT TO} && z < \lambda_i^* \end{aligned}$$

Figure 3: Gaussian ℓ_0 pricing problem for BN variable i where $\mathcal{C}$ is the current set of clusters and λ_i^* and λ_C^* are the current dual values for the constraints.

for each $i \in P$, to find a parent set J, if any, whose reduced cost is most negative (so that at most p variables are added in any pricing round). The pricing problem using this approach is show in Fig 2.

3.2. Pricing for BNSL: a worked example

We will illustrate pricing when the BN score is the Gaussian ℓ_0-penalised maximum likelihood score for scoring Gaussian DAGs (van de Geer and Bühlmann, 2013). The ℓ_0-penalised maximum likelihood score is a decomposable score and the local cost for variable i having parents J is:

$$c_{i\leftarrow J} = L(i, J) + \Lambda^2|J| \tag{5}$$

where

$$L(i, J) = (n/2)[\log(2\pi) + 1 + \log\sigma^2_{i\leftarrow J}] \tag{6}$$

and where n is the size of the data, $n\sigma^2_{i\leftarrow J}$ is the sum of squared residuals resulting from OLS regression using variables J to predict variable i and Λ^2 is the tuning parameter for the ℓ_0 penalty $\Lambda^2|J|$.

After removing the constant terms $\log(2\pi) + 1$ and rescaling Λ^2 we end up with a simplified cost which when plugged into the pricing problem shown in Fig 2 and then rearranging leads to the pricing problem specific to Gaussian ℓ_0-penalised maximum likelihood shown in Fig 3. Note that λ_i^* is independent of J and so has been removed from the objective in Fig 3.

We now illustrate the price-and-cut method to solve a linear relaxation using the `gaussian.test` dataset from the BNLEARN R package (Scutari, 2010). This dataset consists of $n = 5000$ samples generated from a Gaussian distribution defined by choosing parameters for the 7-node DAG shown in Fig 4. The parameters can be found in the script `gaussian.test.R` (available from BNLEARN) which was used to generate the sample. The ℓ_0 tuning parameter Λ^2 was set to 1. Full de-

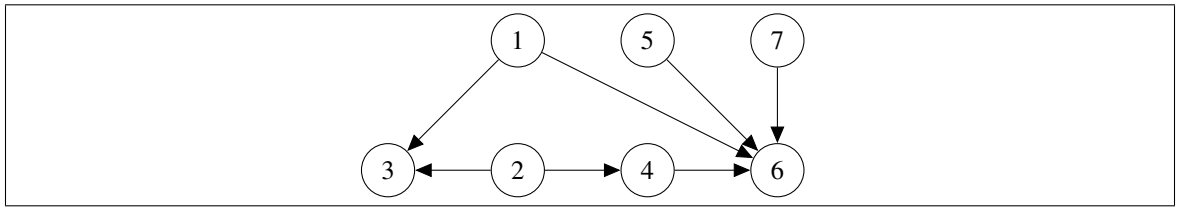

Figure 4: DAG used to generate `gaussian.test` dataset

tails can be found in Appendix A. The results can be reproduced by cloning commit b04ff3f from the `pricing` branch of the `pygobnilp` repo `https://bitbucket.org/jamescussens/pygobnilp`. Just type `python testpricing.py` at the command line.

We start with an initial LP (LP 1) where we only have family variables for empty parent sets: $\mathcal{V} = \{x_{i\leftarrow\emptyset} : i = 1,\ldots,7\}$ and $\mathcal{C} = \emptyset$, so the only constraints are the $p = 7$ linear constraints (2). There is only one feasible solution to this LP, which is thus also optimal: all 7 of the variables are set to 1.

Solving the 7 Gaussian ℓ_0 pricing problems (Fig 3) for $i = 1,\ldots,7$ using the dual values associated with this optimal solution finds the following family variables with minimal reduced cost: $x_{1\leftarrow\{2,3,4,5,6,7\}}$, $x_{2\leftarrow\{1,3,4\}}$, $x_{3\leftarrow\{1,2\}}$, $x_{4\leftarrow\{1,2,5,6,7\}}$, $x_{5\leftarrow\{1,4,6,7\}}$, $x_{6\leftarrow\{1,3,4,5,7\}}$ and $x_{7\leftarrow\{1,3,4,5,6\}}$. These variables have costs -466.13, -1921.01, 3718.43, 1238.48, 6507.84, 7076.24 and 4800.51 respectively and *reduced costs* of -7581.44, -14564.78, -12624.79, -13440.29, -4026.65, -9132.33 and -5718.30, respectively. Since at this point there is no penalty for cycles these are, inevitably, just the parent sets with minimal cost for each $i \in P$, with the reduced cost quantifying how much better these parent sets are than the empty parent set. As expected, each minimal cost parent set approximates the Markov blanket of the child variable in the true data-generating BN (Fig 4). For $i = 1,2,3,4,5$ exactly the Markov blanket is returned, for $i = 6,7$ the parent 3 is added to the Markov blanket, which is probably a result of having a low value (i.e. 1) for Λ^2. Adding these 7 variables (so that now there is a total of 14 variables), resolving the dual LP and pricing again produces no new variables. So we have an optimal solution to the LP with no acyclicity constraints.

At this point a cutting plane algorithm is run. The particular cutting plane method chosen for this illustrative example limits the number of cutting planes to 20 and 20 are indeed found. Once these 20 cuts are added we have a new LP (LP 2) which must be solved to optimality using pricing. It turns out that 7 pricing rounds are required to solve this LP introducing 4,4,5,4,3,1 and 0 family variables respectively. This interleaved process of pricing and cutting is continued until no new cuts can be found. The process is summarised in Table 1. Eventually we obtain a value of 45301 for the solution to the linear relaxation of IP shown in Fig 1. This is achieved using only 52 family variables and 60 cluster constraints, even though there are $7 \times 2^6 = 448$ family variables and $2^7 - 7 = 121$ cluster constraints available. Running the cutting plane algorithm again but this time with all 448 family variables included and thus no need for pricing produces exactly the same objective value for the linear relaxation.

4. Branch-price-and-cut for BNSL

Pricing allows us to solve linear relaxations without explicitly introducing all possible IP variables. But linear relaxations merely provide bounds on IP solutions. Almost always the solution to the linear relaxation of the BNSL IP (Fig 1) will contain non-integer values and thus violate the inte-

LP	$\|C\|$	$\|\mathcal{V}\|$	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

Table 1: Cut-and-price for the `gaussian.test` dataset. Each of the 4 progressively tighter linear relaxations solved to optimality using pricing.

grality constraints (4). To make further progress it is necessary to *branch*: pick a variable (typically one that is non-integer in the linear relaxation solution) and create two subproblems: one with that variable set to 0 and one with that variable set to 1. (All IP variables will be binary.)

Each subproblem thus created is an IP and it is useful to solve its linear relaxation—and we do this using pricing. So pricing is used *at every node of the branch-and-bound tree* used to solve the IP. Cuts may or may not be added: doing so tightens the linear relaxation thus providing better (local) bounds but also takes time.

Instead of branching on family variables $x_{i\leftarrow J}$, we create new binary *arrow variables* $x_{i\leftarrow j}$ to branch on since this produces a more balanced search tree. $x_{i\leftarrow j} = 1$ iff there is an arrow from parent j to child i. Arrow and family variables satisfy the constraints $x_{i\leftarrow j} = \sum_{J:j\in J} x_{i\leftarrow J}$. In any node of the branch-and-bound tree there will be forbidden ($x_{i\leftarrow j} = 0$) and obligatory ($x_{i\leftarrow j} = 1$) arrows. The pricing algorithm for each node is thus (easily) altered to only look for new family variables which are consistent with the set of forbidden and obligatory arrows at that node.

Suppose we are at a node where $x_{i\leftarrow j} = 1$ and $x_{j\leftarrow k} = 1$ then, due to acyclicity we cannot have $x_{k\leftarrow i} = 1$ so $x_{k\leftarrow i}$ should immediately be fixed to 0. To facilitate such propagations a third kind of binary variable is introduced: *partial order variables* $x_{i\rightsquigarrow j}$, where there is such a variable for each ordered pair $i, j \in P$, where $i \neq j$. These variables satisfy the following constraints: $\forall i, j \in P : x_{i\leftarrow j} \leq x_{i\rightsquigarrow j}$, $\forall i, j \in P : x_{i\rightsquigarrow j} + x_{j\rightsquigarrow i} \leq 1$ and $\forall i, j, k \in P : x_{i\rightsquigarrow j} + x_{j\rightsquigarrow k} - x_{i\rightsquigarrow k} \leq 1$. In this way the partial order variables represent a partial order consistent with the ordering induced by the arrow variables and thus permit the desired propagation.

We now have the required ingredients for solving the BNSL IP: *pricing* to solve large linear relaxations, *cutting* to produce linear relaxations that provide good bounds and *branching* since linear relaxations alone are not enough to solve an IP. However to effectively solve large BNSL IP problems some additional steps are required.

Firstly, we need to ensure that unnecessary lower and upper bounds on variables do not appear as constraints in any linear relaxation, since if they do they will have associated dual values which will complicate the pricing algorithm. We will be using the SCIP system (Bestuzheva et al., 2021) since it provides support for pricing, cutting and branching. SCIP allows one to declare lower and upper variable bounds to be *lazy* which prevents the bound appearing in the LP. Here we set the upper bound (i.e. 1) for family variables to be lazy (since they are implied by the constraints (2) in Fig 1), and both the lower and upper bounds of arrow variables to be lazy.

Secondly, we need to represent the constraints between family, arrow and partial order variables appropriately. In any node of the branch-and-bound tree where $x_{i\leftarrow j} = 1$ we clearly want all family variables where child i lacks j as a parent to be ruled out and similarly when $x_{i\leftarrow j} = 0$ all parent sets for i containing j should go. If we represent these constraints as general linear constraints

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

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

Figure 5: Pricing problem for BN variable i where $\mathcal{C}$ is the current set of clusters and λ^*_i, $\lambda^*_{i \leftarrow j}$ and λ^*_C are the current dual values for the constraints.

we will not get these propagations. Since new variables can be created by pricing and then added to existing constraints only propagations that remain valid no matter how the constraint is thus altered can be permitted. In our actual implementation the constraint $x_{i \leftarrow j} = \sum_{J: j \in J} x_{i \leftarrow J}$ is replaced by two *set partitioning constraints* $\sum_{J: j \in J} x_{i \leftarrow J} + \neg x_{i \leftarrow j} = 1$ and $\sum_{J: j \notin J} x_{i \leftarrow J} + x_{i \leftarrow j} = 1$. A set partitioning constraint states that exactly one of a set of binary variables has value 1. Since SCIP is explicitly informed that these are set partitioning constraints it knows that only binary variables with coefficient 1 will be added. From this it follows that SCIP can legitimately (and does!) rule out all family variables for child i lacking j as soon as $x_{i \leftarrow j} = 1$ and similarly those containing j as soon as $\neg x_{i \leftarrow j} = 1$ (i.e. $x_{i \leftarrow j} = 0$). Such propagations are valid irrespective of whether pricing adds new IP variables to the set partitioning constraints. For similar reasons the constraint $x_{i \leftarrow j} \leq x_{i \leftsquigarrow j}$ is actually posted as a *set packing constraint* $x_{i \leftarrow j} + \neg x_{i \leftsquigarrow j} \leq 1$.

Thirdly we do not want constraints that only exist to allow propagations to be included in any linear relaxation. Doing so creates extra dual values for the pricer to deal with and moreover makes the linear relaxation LP unnecessarily bigger (and thus slower to solve). For this reason all constraints involving partial order variables and the constraint $\sum_{J: j \notin J} x_{i \leftarrow J} + x_{i \leftarrow j} = 1$ are not included in linear relaxations. This is done in SCIP by setting the constraints' *initial* and *separate* flag appropriately when the constraint is added to the problem. Note that this means partial order variables are not in any LP.

Fourthly, even though the constraints $x_{i \leftsquigarrow j} + x_{j \leftsquigarrow i} \leq 1$ and $x_{i \leftsquigarrow j} + x_{j \leftsquigarrow k} - x_{i \leftsquigarrow k} \leq 1$ are not in the LP there are many of them. So instead of adding these constraints we create a single *partial order constraint* containing all partial order variables. This is done by using a slightly altered version of the 'LOP' linear ordering constraint handler (written by March Pfetsch) that is supplied as an example with the SCIP system. The end result is that we get the propagations we want.

Fifthly, for each arrow $i \leftarrow j$, the constraints $\sum_{J: j \in J} x_{i \leftarrow J} + \neg x_{i \leftarrow j} = 1$ are present in the LP and thus have associated dual values which must be taken in account when pricing. Let $\lambda^*_{i \leftarrow j}$ be the dual value for $\sum_{J: j \in J} x_{i \leftarrow J} + \neg x_{i \leftarrow j} = 1$. The pricing problems in Figs 2 and 3 can accommodate arrow dual values by adding $-\sum_{j \in J} \lambda^*_{i \leftarrow j}$ to the objective. Fig 5 is the pricing problem with dual values for the arrow constraints included.

5. Optimisations

The preceding sections have outlined the basic branch-price-and-cut approach. We now move on to consider optimisations, beginning with a look at how to restrict the set of candidate family variables for pricing-in. Suppose we have some candidate parent set J for some BN child i which we know is

not required for an optimal DAG, so the constraint $x_{i\leftarrow J} = 0$ is valid. It follows that such a family variable $x_{i\leftarrow J}$ should not be added by pricing, *even if it has negative reduced cost in some linear relaxation*.

Fortunately, there are typically many such family variables that can be excluded. Let J be called a *potentially optimal parent set (POP)* for i if: $\forall J' \subsetneq J : c_{i\leftarrow J} < c_{i\leftarrow J'}$, i.e. all of J's proper subsets have strictly higher cost. If J is *not* a POP for i then it follows that there is an optimal DAG with $x_{i\leftarrow J} = 0$ since if we had a DAG where J were the parent set for i then we could replace J with a proper subset J' with $c_{i\leftarrow J} \geq c_{i\leftarrow J'}$ and create a DAG whose cost is no higher. So we could safely add the following constraints to the IP for BNSL (Fig 1):

$$\forall i \in P, J \subseteq P \setminus \{i\} : \exists J' \subsetneq J : c_{i\leftarrow J} \geq c_{i\leftarrow J'} \rightarrow x_{i\leftarrow J} = 0 \tag{7}$$

However, directly adding the constraints (7) by considering each $i \in P, J \subseteq P \setminus \{i\}, J' \subsetneq J$ is clearly impracticable for all but the smallest problems, so a more sophisticated approach is required. In the standard (without-pricing) version of GOBNILP only POP family variables (up to some pre-defined cardinality limit) are included in the problem. When we allow pricing then we want (i) to avoid pricing-in known non-POPs and (ii) to exploit the restriction to POPs to speed up the pricing algorithm.

Consider now the example given in Section 3.2. In the first round of pricing we found minimal cost parent sets for each child variable. For example, it was found that a minimal cost parent set for child 3 was $\{1, 2\}$ with a cost of 3718 (rounded to nearest integer). We will assume throughout that if J and J' are both minimal costs parent sets and $J' \subsetneq J$ then J is not the returned minimal cost parent set. With this assumption, it follows that $\{1, 2\}$ is a POP for child 3. More importantly, since $\{1, 2\}$ has minimal cost, there are no POPs for child 3 in the set interval $\{J : \{1, 2\} \subsetneq J \subseteq P \setminus \{3\}\}$ which has cardinality $2^6 - 4 = 60$. So for child 3 we have 'got lucky': not only have we identified a POP, we have also identified a large number of non-POPs and thirdly we know that if 1 and 2 can be made parents of 3 without creating a cycle then $\{1, 2\}$ should be chosen as the parent set for 3. To formalise this last point, let $x_{i\nleftarrow j}$ denote that j is an ancestor of i. It follows that whenever 3 is a non-ancestor of both 1 and 2 then, since we are only concerned with finding an optimal DAG, we can just fix the parent set of 3 to be $\{1, 2\}$:

$$\neg x_{1\nleftarrow 3} \wedge \neg x_{2\nleftarrow 3} \rightarrow x_{3\leftarrow\{1,2\}} = 1 \tag{8}$$

Call constraints such as (8) *best-parent-set* constraints.

Since identifying low cardinality minimal cost parent sets can be so useful it also worth finding minimal cost parent sets where the set of allowed parents is reduced. Returning again to the example in Section 3.2 we see that the minimal cost parent set for 7 is $\{1, 3, 4, 5, 6\}$ with a cost of 4800. However, if we disallow 6 as a parent it turns out that the minimal cost parent set is the empty set with a cost of 10518. (This is no surprise since in the data-generating DAG (Fig 4) 7 is d-separated from all other BN variables conditional on any set not containing 6.) This means there are no POPs for child 7 in the set interval $\{J : \{\} \subsetneq J \subseteq P \setminus \{6, 7\}\}$. The relevant best-parent-set constraint is $x_{6\nleftarrow 7} \rightarrow x_{7\leftarrow\{\}} = 1$.

A procedure for finding set intervals of non-POPs is given in Appendix B. The basic idea is to find minimal cost parent sets when given subsets of parents are disallowed and to record the associated set intervals of non-POPs. The pricer can then exclude parent sets in these non-POP intervals from its search for negative reduced cost family variables.

Another simple optimisation is to simply do less pricing by precomputing the local costs for all potentially optimal parent sets (POPs) up to some predetermined cardinality limit k. This can speed up later pricing since we can add a constraint to indicate that all necessary parent sets of size up to k have already been found. Another optimisation adds cutting planes additional to cluster constraints in the form of k-cluster constraints (Cussens, 2011). These tighten the linear relaxations and only slightly complicate the pricing algorithm.

A last optimisation concerns how pricing and cutting are interleaved. In the standard approach to price-and-cut, pricing is used to solve each of a series of linear programs to optimality. Once each LP is solved a cutting plane algorithm is run to produce a new LP which is a tighter linear relaxation. This is the method used in the example outlined in Section 3.2 and described in detail in Appendix A. It is also how the SCIP solver expects price-and-cut to be implemented. But why spend time solving an LP to optimality if it will soon be replaced by a tighter relaxation due to the addition of cutting planes? Based on this argument GOBNILP allows the user to *delay pricing*. In this approach pricing is only run once the cutting plane algorithm can find no new cutting planes.

Above we have listed a number of optimisations specific to pricing. When using a Gaussian ℓ_0 objective there is another useful optimisation that can be used whether or not we are pricing. If Λ^2, the ℓ_0 penalty, were set to 0 then clearly every complete DAG would be optimal with a common cost, which we will denote L^*. So we have the following inequality which is valid no matter what value of Λ^2 is actually being used:

$$\sum_{\substack{i \in P \\ J \subseteq P \setminus \{i\}}} L(i, J) x_{i \leftarrow J} \geq L^* \tag{9}$$

This has proved to be a useful inequality for getting a lower bound in the normal case where $\Lambda^2 > 0$ since its coefficients only differ from those of the objective by $\Lambda^2 |J|$. Since this inequality is in each linear relaxation, pricing has to take its dual value into account, which is easily done.

6. Does it work?

The preceding sections have outlined an integer programming approach to learning DAGs under causal sufficiency where potential parent sets for individual random variables are introduced in the course of the learning (=optimisation). The issue is whether this an effective approach for this sort of causal discovery.

The short answer is that it is possible to learn optimal DAGs using pricing (together with cutting and branching), but the time it takes to do this depends crucially on how quickly the pricing problem (Fig 5) can be solved and how many IP variables have to be 'priced-in' to solve each of the many linear relaxations that arise. Note that (typically) several pricing rounds will be run *for every node in the IP search tree* to ensure the linear relaxation for that node is solved to optimality. Consider, for example, the Gaussian ℓ_0 pricing problem (shown, with dual values for 'arrow constraints' omitted, in Fig 3). One strategy for solving this problem is to encode it in such a way that an existing solver can solve it. This is the approach we take: the encoding as a non-convex constrained optimisation problem is given in Fig 6, the solver we use is SCIP. In Fig 6 each binary variable y_j indicates that j is in the new parent set for i. The β_j are linear regression coefficients, $\boldsymbol{X}_i$ is the vector of observed values for random variable i in the data, $\boldsymbol{X}_{-i}$ is the data matrix of all variables other than i and S is the matrix $\boldsymbol{X}_{-i}^\top \boldsymbol{X}_{-i}$. The variable x_{σ^2} represents squared error (when predicting random variable

$$\text{MINIMISE} \qquad z = n x_{\log \sigma^2} + \sum_{j \in P \setminus \{i\}} (\Lambda^2 - \lambda^*_{i \leftarrow j}) y_j + \sum_{C \in \mathcal{C}} \lambda^*_C y_C \tag{10}$$

$$\text{SUBJECT TO} \qquad \sum_{j \in P \setminus \{i\}} \gamma_j^2 + c \leq n x_{\sigma^2} \tag{11}$$

$$\log x_{\sigma^2} \leq x_{\log \sigma^2} \tag{12}$$

$$\boldsymbol{\gamma} = S^{1/2} \boldsymbol{\beta} - S^{-1/2} \boldsymbol{X}_{-i}^\top \boldsymbol{X}_i \tag{13}$$

$$(\beta_j, 1 - y_j) : \text{SOS-1} \qquad j \in P \setminus \{i\} \tag{14}$$

$$y_C = \bigvee_{j \in C} y_j \qquad C \in \mathcal{C} \tag{15}$$

$$z < \lambda^*_i$$

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

Figure 6: Gaussian ℓ_0 pricing for BN variable i as nonlinear constrained optimisation.

i in a linear regression model with coefficients β_j) and $x_{\log \sigma^2}$ is the log of that squared error. The 'SOS-1' constraints state that $y_j = 0 \rightarrow \beta_j = 0$. Intermediate variables γ_j have been introduced so that the constraint between linear regression coefficients and squared error can be expressed as the *second-order cone* constraint (11). This has been shown (empirically) to lead to faster solving than expressing this relationship directly. A binary variable $y_C, C \in \mathcal{C}$ indicates that the parent set intersects with the cluster C in which case the associated 'dual penalty' λ^*_C is included in the objective value (10). $y_C = 1$ if and only if a parent is a member of C (15). Constraints for ruling out 'non-POP set intervals' (see Section 5) are easily added to Fig 6. If we wish to exclude all $\{J : \underline{J} \subseteq J \subseteq \overline{J}\}$ then the constraint $\sum_{j \in \underline{J}} \neg y_j + \sum_{j \in P \setminus \overline{J}} y_j \geq 1$ is added.

For the 7 BN variable `gaussian.test` dataset used in Section 3.2 it takes just under 3 minutes to find a guaranteed optimal DAG using pricing (pricing not delayed, all POPs up to size 2 precomputed, without identifying intervals of non-POPs). If pricing is delayed, then the time is reduced to just under 2 minutes, so in this case delaying pricing brought some advantage. If we now identify intervals of non-POPs by considering disallowed parent sets of size at most one, then solving comes down to 48 seconds. However pricing is not required for such a small example and GOBNILP-without-pricing learns an optimal DAG for this dataset in under a second! (All experiments conducted with Gaussian BIC scoring with no parent set size limit using GOBNILP commit 2495e79 (neat branch) on a single core of a 15 Gb RAM, 2.7GHz CPU laptop. SCIP used CPLEX to solve LP relaxations.)

To test the usefulness of pricing on a harder problem, a dataset with 20 random variables and 100 datapoints was generated from a known true Gaussian DAG. (This was done using a routine included with the NOTEARS (Zheng et al., 2018) software.) If pricing is not used, solving takes 327 seconds (31 seconds to generate 6634 local costs, 84 seconds to find an optimal DAG and a further 212 seconds to prove that this DAG is indeed optimal). On this particular dataset using pricing always performs worse than not using pricing - no variant of pricing has been able to solve the problem to optimality, although reasonably good incumbents and lower bounds are produced.

The general point, which these two problems exemplify, is that pricing should only be used when necessary. This is principally because, at least in the case of the non-convex pricing problem shown in Fig 6, pricing is expensive—when pricing is used it takes up almost all of the solving time. If pricing is not used then not only is no time spent pricing-in new variables but the IP solver can take advantage of the knowledge that no new IP variables will be introduced. Moreover, if there is no pricing then the solver is free to add in any cutting planes (additional to the cluster constraints) that it finds useful to add. The SCIP system has many 'separators' which add cutting planes, including e.g. separators for adding zero-half cuts and Gomory cuts (see Bestuzheva et al. (2021) for details). In contrast, when pricing we cannot allow this since we need to keep track of all cuts to account for their dual values and to know whether new priced-in variables should be added to them.

However, there are cases where pricing is necessary to make any progress. For example, a 10,000 datapoint dataset with 25 random variables was generated from a known true Gaussian DAG. When GOBNILP without pricing attempted to learn from this dataset it ran out of memory while attempting to create the IP since the number of initial IP variables was too great. This problem disappears when pricing is used since only a small number of initial family variables need be found. Using pricing on this dataset (no matter which variant of pricing) an incumbent DAG and lower bound are found in a few seconds where the 'gap' between these two is only 0.06%, so, using pricing, we have quickly found a DAG close to optimal, instead of running out of memory and not returning any DAG. However, getting this gap down to 0% (i.e. finding a provably optimal DAG) has so far not proved possible.

7. Discussion

We have a working implementation—an extended version of GOBNILP(commit 2495e79 at `https://bitbucket.org/jamescussens/gobnilp`)—that allows new candidate parent sets to be created during the solving/learning process via *pricing*. This is done in a principled way by considering *reduced costs* for these candidate parent sets which take into account the dual values associated with constraints ensuring acyclicity. We have shown, unsurprisingly, that fast pricing is required for fast learning and shown that, perhaps less obviously, delaying pricing can bring benefits.

Although we view this contribution as a useful first step in extending score-based causal discovery, it is clear that there is some way to go before this approach is a state-of-the-art method. Some of the necessary work concerns the details of implementation and some is more algorithmic: designing fast pricing algorithms for a range of popular scores for causal model learning.

In this paper we have focused exclusively on DAG learning with the assumption of causal sufficiency. However, it is actually score-based learning of more complex causal models, e.g. those with latent variables where pricing can be of most benefit. For example, Chen et al. (2021) showed how to encode BIC score-based learning of ancestral acyclic directed mixed graphs (ADMGs) as an IP problem. But ADMG learning requires even more IP variables than DAG learning: one for every candidate district. It would be interesting to investigate the feasibility of pricing in candidate districts while learning the ADMG.

References

Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 8.0. Technical report, Optimization Online, December 2021. URL http://www.optimization-online.org/DB_HTML/2021/12/8728.html.

Rui Chen, Sanjeeb Dash, and Tian Gao. Integer programming for causal structure learning in the presence of latent variables. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1550–1560. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/chen21c.html.

David M. Chickering, David Heckerman, and Christopher Meek. Large-sample learning of Bayesian networks is NP-hard. *Journal of Machine Learning Research*, 20:1287–1330, October 2004.

James Cussens. Maximum likelihood pedigree reconstruction using integer programming. In *Proceedings of the Workshop on Constraint Based Methods for Bioinformatics (WCB-10)*, Edinburgh, July 2010.

James Cussens. Bayesian network learning with cutting planes. In Fabio G. Cozman and Avi Pfeffer, editors, *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence (UAI 2011)*, pages 153–160, Barcelona, 2011. AUAI Press.

James Cussens, David Haws, and Milan Studený. Polyhedral aspects of score equivalence in Bayesian network structure learning. *Mathematical Programming*, 164:285–324, 2017a. doi:10.1007/s10107-016-1087-2.

James Cussens, Matti Järvisalo, Janne H. Korhonen, and Mark Bartlett. Bayesian network structure learning with integer programming: Polytopes, facets, and complexity. *Journal of Artificial Intelligence Research*, 58:185–229, 2017b.

Mathias Drton and Marloes H. Maathuis. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4(1):365–393, 2017. doi: 10.1146/annurev-statistics-060116-053803. URL https://doi.org/10.1146/annurev-statistics-060116-053803.

Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *Proceedings of 13th International Conference on Artificial Intelligence and Statistics (AISTATS 2010)*, volume 9, pages 358–365, 2010. Journal of Machine Learning Research Workshop and Conference Proceedings.

Marco Scutari. Learning Bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(3):1–22, 2010. doi: 10.18637/jss.v035.i03.

Sara van de Geer and Peter Bühlmann. ℓ_0-penalized maximum likelihood for sparse directed acyclic graphs. *The Annals of Statistics*, 41(2):536–567, 2013.

Laurence A. Wolsey. *Integer Programming*. Wiley, 1998.

Xun Zheng, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. Dags with no tears: Continuous optimization for structure learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/e347c51419ffb23ca3fd5050202f9c3d-Paper.pdf.

Appendix A. Pricing for BNSL: a worked example in detail

This section contains full details of the Gaussian ℓ_0-penalised maximum likelihood pricing example summarised in Section 3.2. The objectives here are given in terms of *scores* rather than *costs* and are not rounded to the nearest integer. So, for example, the objective for the first LP (which has no cluster constraints) is shown here to have an optimal solution with objective value of −20954.37 rather than 20954 as given in Section 3.2. Similarly, the two values given for a priced-in family variable are its local score and its reduced local score. Note that for priced-in variables the reduced local score is always positive (i.e. the reduced cost is always negative).

Now to explain the following table in more detail. $\mathcal{V}_i$ is the set of family variables introduced in pricing round $i-1$ with the initial family variables denoted $\mathcal{V}_1$. C_i are the clusters associated with the cluster constraint cutting planes introduced in separation round i. Separation rounds are thus called since the cutting planes found in a separation round separate the current LP solution from the convex hull of integer solutions. Each section in the table with title **LP** specifies a linear program with variables $\mathcal{V}$ and with cluster constraints given by some set of clusters C. Included is the primal solution to the LP after the text "Solution is", the objective value and the solution to the dual of the LP (after the text "Dual values"). The dual values λ_i^* for $i = 1, \ldots, 7$ are given, in order, after the text "λ_i^* =". These are the dual values associated with the equations (2) in Fig 1. Dual values λ_C^* for cluster constraints are also given, except those which have value 0 are omitted.

After each **LP** section there is a section **Pricing** which shows which new variables, if any, were added to the LP to get a better LP solution. Each priced-in variable is given with its local score and reduced local score. If a pricing round ends with no new variables to add it follows that the LP has been solved to optimality and so a separation round is done to find cuts to add to the LP. The clusters associated with these cuts are in the sections with title **Separation**. The process (known as the *price-and-cut loop*) ends when no cutting planes can be found.

$\mathcal{V}_1 = x_{1\leftarrow\{\}}, x_{2\leftarrow\{\}}, x_{C\leftarrow\{\}}, x_{4\leftarrow\{\}}, x_{5\leftarrow\{\}}, x_{6\leftarrow\{\}}, x_{7\leftarrow\{\}}$

LP. $C = \emptyset$. $\mathcal{V} = \mathcal{V}_1$. Solution is $x_{1\leftarrow\{\}} = 1$, $x_{2\leftarrow\{\}} = 1$, $x_{3\leftarrow\{\}} = 1$, $x_{4\leftarrow\{\}} = 1$, $x_{5\leftarrow\{\}} = 1$, $x_{6\leftarrow\{\}} = 1$, $x_{7\leftarrow\{\}} = 1$. Objective is −88043.06. Dual values: $\lambda_i^* = -7115.31, -12643.78, -16343.24, -14678.79, -10534.51, -16208.60, -10518.84$.

Pricing. New variables: $\mathcal{V}_2 = (x_{1\leftarrow\{2,3,4,5,6,7\}}, 466.13, 7581.44)$, $(x_{2\leftarrow\{1,3,4\}}, 1921.01, 14564.78)$, $(x_{3\leftarrow\{1,2\}}, -3718.43, 12624.79)$, $(x_{4\leftarrow\{1,2,5,6,7\}}, -1238.48, 13440.29)$, $(x_{5\leftarrow\{1,4,6,7\}}, -6507.84, 4026.65)$, $(x_{6\leftarrow\{1,3,4,5,7\}}, -7076.24, 9132.33)$, $(x_{7\leftarrow\{1,3,4,5,6\}}, -4800.51, 5718.30)$.

LP. $C = \emptyset$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{2}\mathcal{V}_i$. Solution is $x_{1\leftarrow\{2,3,5,6,4,7\}} = 1, x_{2\leftarrow\{1,3,4\}} = 1, x_{3\leftarrow\{1,2\}} = 1, x_{4\leftarrow\{1,2,5,6,7\}} = 1, x_{5\leftarrow\{1,4,6,7\}} = 1, x_{6\leftarrow\{1,3,4,5,7\}} = 1, x_{7\leftarrow\{1,3,4,5,6\}} = 1$. Objective is -20954.37. Dual values: $\lambda_i^* = 466.13, 1921.01, -3718.43, -1238.48, -6507.84, -7076.24, -4800.51$.

Pricing. New variables: None

Separation. New clusters: $C_1 = \{1,2\}, \{2,4\}, \{4,5\}, \{1,3\}, \{1,4\}, \{5,6\}, \{1,5\}, \{2,3\}, \{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\}, \{1,2,3,4\}, \{1,3,5\}, \{1,4,5\}, \{1,5,6\}, \{2,3,4,5\}, \{1,2,4,5\}, \{2,4,7\}, \{1,3,4,5\}$.

LP. $C = C_1$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{2}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1, x_{2\leftarrow\emptyset} = 1, x_{3\leftarrow\{1,2\}} = 1, x_{4\leftarrow\{1,2,5,6,7\}} = 1, x_{5\leftarrow\emptyset} = 1, x_{6\leftarrow\{1,3,4,5,7\}} = 1, x_{7\leftarrow\{1,3,4,5,6\}} = 1$. Objective is -47127.29. Dual values: $\lambda_i^* = -7115.31, -12643.78, -12424.37, -14678.78, -10534.51, -11102.91, -4800.51$. $\lambda_{\{2,4\}}^* = 13440.30$, $\lambda_{\{1,3\}}^* = 7581.45$, $\lambda_{\{5,6\}}^* = 4026.66$, $\lambda_{\{2,3\}}^* = 1124.49$.

Pricing. New variables: $\mathcal{V}_3 = (x_{1\leftarrow\{4,5,6,7\}}, -3071.68, 4043.62)$, $(x_{2\leftarrow\{1,3,5,6,7\}}, 97.25, 11616.53)$, $(x_{3\leftarrow\{4,5,6,7\}}, -7713.50, 4710.86)$, $(x_{4\leftarrow\{1,3,5,6,7\}}, -3060.54, 11618.23)$.

LP. $C = C_1$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{3}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{2\leftarrow\emptyset} = 1/2$, $x_{4\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\emptyset} = 1$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{2\leftarrow\{1,3,4\}} = 1/2$ $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1$, $x_{3\leftarrow\{4,5,6,7\}} = 1/2$. Objective is -44771.85. Dual values: $\lambda_i^* = -7115.31, -12643.78, -14105.73, -14678.78, -10534.51, -11102.91, -4800.51$, $\lambda_{\{1,2\}}^* = 2355.43$, $\lambda_{\{2,4\}}^* = 1822.05$, $\lambda_{\{1,4\}}^* = 5226.01$, $\lambda_{\{5,6\}}^* = 4026.66$, $\lambda_{\{2,3\}}^* = 3995.07$, $\lambda_{\{2,3,4\}}^* = 6392.23$.

Pricing. New variables: $\mathcal{V}_4 = (x_{1\leftarrow\{2,3,5,6,7\}}, 418.10, 5177.97)$, $(x_{2\leftarrow\{1,5,6,7\}}, -5192.93, 5095.39)$, $(x_{3\leftarrow\{1,5,6,7\}}, -9010.62, 5095.10)$, $(x_{4\leftarrow\{2,3,5,6,7\}}, -1287.31, 5177.17)$.

LP. $C = C_1$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{4}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{2\leftarrow\emptyset} = 1/2$, $x_{4\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\emptyset} = 1$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{2\leftarrow\{1,3,4\}} = 1/2$ $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1$, $x_{3\leftarrow\{4,5,6,7\}} = 1/2$. Objective is -44771.85. Dual values: $\lambda_i^* = -7115.31, -12643.784, -14188.30, -14678.78, -10534.51, -11102.91, -9895.92$, $\lambda_{\{1,2\}}^* = 2355.43$, $\lambda_{\{2,4\}}^* = 1822.05$ $\lambda_{\{1,4\}}^* = 48.03$, $\lambda_{\{5,6\}}^* = 4026.66$, $\lambda_{\{2,3\}}^* = 3995.07$, $\lambda_{\{1,3,4\}}^* = 5177.98$, $\lambda_{\{2,3,4\}}^* = 1296.82$, $\lambda_{\{2,4,7\}}^* = 5095.40$.

Pricing. New variables: $\mathcal{V}_5 = (x_{1\leftarrow\{2,5,6,7\}}, -3304.52, 1455.33)$, $(x_{2\leftarrow\{4\}}, 503.85, 4933.34)$, $(x_{3\leftarrow\{5,6,7\}}, -9273.27, 4915.02)$, $(x_{4\leftarrow\{2,5,6,7\}}, -1471.23, 4993.26)$, $(x_{7\leftarrow\{1,3,5,6\}}, -5361.48, 4534.43)$.

LP. $C = C_1$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{5}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{2\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\emptyset} = 1$, $x_{2\leftarrow\{1,3,4\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{4\leftarrow\{1,3,5,6,7\}} = 1/2$, $x_{1\leftarrow\{2,3,5,6,7\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{7\leftarrow\{1,3,5,6\}} = 1/2$. Objective value is -40047.12. Dual values: $\lambda_i^* = -7115.31, -12643.78, -9273.27, -14678.78, -10534.51, -10858.29, -5361.48$, $\lambda_{\{1,2\}}^* = 1417.15$, $\lambda_{\{2,4\}}^* = 1822.05$, $\lambda_{\{4,5\}}^* = 244.62$, $\lambda_{\{1,3\}}^* = 453.24$, $\lambda_{\{1,4\}}^* = 48.03$, $\lambda_{\{5,6\}}^* = 3782.04$, $\lambda_{\{1,2,4\}}^* = 5663.02$, $\lambda_{\{2,3,4\}}^* = 5101.60$, $\lambda_{\{2,4,7\}}^* = 560.96$.

Pricing. New variables: $\mathcal{V}_6 = (x_{1\leftarrow\{3,4,5,6,7\}}, -495.90, 455.10)$, $(x_{2\leftarrow\{5,6,7\}}, -8632.31, 3450.49)$, $(x_{4\leftarrow\{5,6,7\}}, -10605.66, 3267.53)$, $(x_{5\leftarrow\{1,2,6,7\}}, -6729.11, 23.33)$.

LP. $C = C_1$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{6}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow 2,3,5,6,7} = 1/2$, $x_{2\leftarrow 5,6,7} = 1/2$, $x_{2\leftarrow 1,3,4} = 1/2$, $x_{3\leftarrow 1,2} = 1/2$, $x_{3\leftarrow 5,6,7} = 1/2$, $x_{4\leftarrow 1,2,5,6,7} = 1/2$, $x_{4\leftarrow 1,3,5,6,7} = 1/2$, $x_{5\leftarrow\emptyset} = 1$, $x_{6\leftarrow 1,3,4,5,7} = 1$, $x_{7\leftarrow 1,3,5,6} = 1$. Objective is -38321.86. Dual values: $\lambda_i^* = -7115.31, -9193.27, -9273.27, -11204.93, -10534.51, -10881.64, -5361.48$, $\lambda_{\{1,2\}}^* = 962.04$, $\lambda_{\{2,4\}}^* = 1822.05$, $\lambda_{\{4,5\}}^* = 221.27$, $\lambda_{\{1,3\}}^* = 2178.49$, $\lambda_{\{1,4\}}^* = 48.03$, $\lambda_{\{5,6\}}^* = 3805.39$, $\lambda_{\{2,3\}}^* = 455.11$, $\lambda_{\{1,2,4\}}^* = 4392.88$, $\lambda_{\{2,3,4\}}^* = 2921.23$, $\lambda_{\{2,4,7\}}^* = 560.96$.

Pricing. New variables: $\mathcal{V}_7 = (x_{1\leftarrow\{5,6,7\}}, -6743.90, 371.39)$, $(x_{2\leftarrow\{1,3,5,6\}}, -202.07, 259.92)$, $(x_{4\leftarrow\{1,3,6\}}, -3634.97, 207.79)$.

LP. $C = C_1$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{7}\mathcal{V}_i$. Solution is $x_{1\leftarrow\{2,3,5,6,7\}} = 1/2$, $x_{1\leftarrow\{5,6,7\}} = 1/2$, $x_{2\leftarrow\{5,6,7\}} = 1/2$, $x_{2\leftarrow\{1,3,5,6\}} = 1/2$. $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1$, $x_{5\leftarrow\emptyset} = 1$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{7\leftarrow\{1,3,5,6\}} = 1/2$, Objective is -38006.19. Dual values: $\lambda_i^* = -6743.90$, -9193.27, -9273.27, -11204.93, -10534.51, -10881.64, -5361.48, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{2,4\}} = 1614.25$, $\lambda^*_{\{4,5\}} = 221.27$, $\lambda^*_{\{1,3\}} = 1862.82$, $\lambda^*_{\{1,4\}} = 48.03$, $\lambda^*_{\{5,6\}} = 3805.39$, $\lambda^*_{\{2,3\}} = 507.23$, $\lambda^*_{\{1,2,4\}} = 4337.14$, $\lambda^*_{\{2,3,4\}} = 3184.78$, $\lambda^*_{\{2,4,7\}} = 560.96$.

Pricing. New variables: $\mathcal{V}_8 = (x_{4\leftarrow\{2,3,6,7\}}, -1460.31, 47.46)$,

---

LP. $C = C_1$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{8}\mathcal{V}_i$. Solution is $x_{1\leftarrow\{5,6,7\}} = 1/2$, $x_{1\leftarrow\{2,3,5,6,7\}} = 1/2$, $x_{2\leftarrow\{5,6,7\}} = 1/2$, $x_{2\leftarrow\{1,3,5,6\}} = 1/2$. $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1$, $x_{5\leftarrow\emptyset} = 1$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{7\leftarrow\{1,3,5,6\}} = 1/2$, Objective is -38006.19 Dual values $\lambda_i^* = -6743.90$, -9193.27, -9273.27, -11157.45, -10534.51, -10929.12, -5361.48, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{2,4\}} = 1661.73$, $\lambda^*_{\{4,5,\}} = 173.79$, $\lambda^*_{\{1,3\}} = 1862.8$, $\lambda^*_{\{1,4\}} = 48.03$, $\lambda^*_{\{5,6\}} = 3852.87$, $\lambda^*_{\{2,3\}} = 554.71$, $\lambda^*_{\{1,2,4\}} = 4337.14$, $\lambda^*_{\{2,3,4\}} = 3137.30$, $\lambda^*_{\{2,4,7\}} = 560.96$.

Pricing. New variables: None.

Separation. New clusters: $C_2 = \{1,7\}, \{2,3,7\}, \{6,7\}, \{1,2,7\}, \{2,6,7\}, \{1,3,7\}, \{1,2,3,7\}, \{4,6,7\}, \{1,4,7\}, \{2,4,6,7\}, \{1,2,4,7\}, \{1,3,4,7\}, \{2,3,4,7\}, \{1,2,3,4,7\}, \{1,4,6,7\}, \{4,6\}, \{1,3,4,6,7\}, \{1,4,6\}, \{1,2,4,6,7\}, \{2,4,6\}$.

LP. $C = \dot{\bigcup}_{i=1}^{2}C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{8}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{2\leftarrow\{1,3,4\}} = 1/2$, $x_{2\leftarrow\{4\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$ $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\emptyset} = 1/2$, $x_{4\leftarrow\{1,3,5,6,7\}} = 1/2$ $x_{5\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\{1,4,6,7\}} = 1/2$ $x_{6\leftarrow\emptyset} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1/2$ $x_{7\leftarrow\emptyset} = 1/2$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$. Objective value -45300.94 Dual values: $\lambda_i^* = -7115.31$, -10683.44, -11023.37, -14678.78, -10534.51, -16208.59, -10518.83, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{5,4\}} = 221.27$, $\lambda^*_{\{1,3\}} = 2016.53$, $\lambda^*_{\{4,1\}} = 0.55$, $\lambda^*_{\{5,6\}} = 3805.39$, $\lambda^*_{\{2,3\}} = 455.11$, $\lambda^*_{\{4,1,2\}} = 4230.91$, $\lambda^*_{\{4,2,3\}} = 3083.20$, $\lambda^*_{\{4,7,2\}} = 301.03$, $\lambda^*_{\{1,7\}} = 371.40$, $\lambda^*_{\{6,7\}} = 3295.78$, $\lambda^*_{\{4,7,2,3\}} = 1750.09$, $\lambda^*_{\{4,6\}} = 2031.167$.

Pricing. New variables: $\mathcal{V}_9 = (x_{1\leftarrow\{5,6\}}, -6845.99, 269.30)$, $(x_{2\leftarrow\{5,6\}}, -10383.55, 299.87)$, $(x_{4\leftarrow\{2\}}, -1531.14, 1960.32)$, $(x_{6\leftarrow\{1,2\}}, -13663.57, 2545.00)$.

LP. $C = \dot{\bigcup}_{i=1}^{2}C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{9}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{2\leftarrow\emptyset} = 1/2$, $x_{2\leftarrow\{1,3,4\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{1,3,5,6,7\}} = 1/2$, $x_{4\leftarrow\{2\}} = 1/2$, $x_{5\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\{1,4,6,7\}} = 1/2$, $x_{6\leftarrow\emptyset} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1/2$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{7\leftarrow\emptyset} = 1/2$. Objective value is -45300.94. Dual values: $\lambda_i^* = -7115.31$, -12643.78, -10438.69, -14678.78, -10534.51, -16208.59, -10518.83, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{4,5,\}} = 221.27$, $\lambda^*_{\{1,3\}} = 1431.85$, $\lambda^*_{\{1,4\}} = 0.55$, $\lambda^*_{\{5,6\}} = 3536.07$, $\lambda^*_{\{1,5\}} = 269.31$, $\lambda^*_{\{2,3\}} = 455.11$, $\lambda^*_{\{1,2,4\}} = 3646.24$, $\lambda^*_{\{2,3,4\}} = 3667.87$, $\lambda^*_{\{2,4,7\}} = 301.03$, $\lambda^*_{\{1,7\}} = 1271.44$, $\lambda^*_{\{6,7\}} = 2980.42$, $\lambda^*_{\{2,3,4,7\}} = 1165.42$, $\lambda^*_{\{4,6\}} = 70.82$, $\lambda^*_{\{2,4,6\}} = 2545.01$.

Pricing. New variables: $\mathcal{V}_{10} = (x_{1\leftarrow\{6,2,3\}}, -61.03, 1014.12)$, $(x_{2\leftarrow\{1,3\}}, -237.38, 2509.69)$, $(x_{4\leftarrow\{1,3\}}, -3690.99, 2507.68)$, $(x_{6\leftarrow\{1,3\}}, -13691.81, 2516.76)$, $x_{7\leftarrow\{2,3,5,6,4\}}, -5316.08, 755.86)$.

LP. $C = \dot{\bigcup}_{i=1}^{2}C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{10}\mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/4$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{1\leftarrow\{5,6,7\}} = 1/4$, $x_{2\leftarrow\emptyset} = 1/4$, $x_{2\leftarrow\{1,3\}} = 1/2$, $x_{2\leftarrow\{5,6,7\}} = 1/4$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1/4$, $x_{4\leftarrow\{2\}} = 3/4$, $x_{5\leftarrow\emptyset} = 3/4$, $x_{5\leftarrow\{1,4,6,7\}} = 1/4$, $x_{6\leftarrow\{1,3,4,5,7\}} = 3/4$, $x_{6\leftarrow\{1,3\}} = 1/4$ $x_{7\leftarrow\emptyset} = 3/4$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/4$, Objective is -43970.52. Dual values: $\lambda_i^* = -7115.31$, -12643.78, -12804.34, -14678.78, -10534.51, -13961.12, -10518.83, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{1,3\}} = 1944.77$, $\lambda^*_{\{5,6\}} = 2246.83$, $\lambda^*_{\{2,3\}} = 599.17$, $\lambda^*_{\{4,1,2\}} = 4303.22$, $\lambda^*_{\{4,2,3\}} = 3010.89$, $\lambda^*_{\{5,4,2,3\}} = 1779.82$, $\lambda^*_{\{1,7\}} = 102.09$, $\lambda^*_{\{6,7\}} = 3864.98$, $\lambda^*_{\{4,7,2,3\}} = 1751.24$, $\lambda^*_{\{4,6\}} = 23.34$, $\lambda^*_{\{4,1,6\}} = 269.31$, $\lambda^*_{\{4,6,2\}} = 480.39$.

Pricing. New variables: $\mathcal{V}_{11} = (x_{2\leftarrow\{6\}}, -10741.98, 1421.39)$, $(x_{4\leftarrow\{6\}}, -12755.05, 1150.66)$, $(x_{6\leftarrow\{3\}}, -13734.11, 227.00)$.

LP. $C = \dot{\bigcup}_{i=1}^{2} C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{11} \mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\emptyset} = 1/2$, $x_{7\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1/2$, $x_{5\leftarrow\{1,4,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1/2$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{2\}} = 1/2$, $x_{2\leftarrow\{1,3\}} = 1/2$, $x_{6\leftarrow\{1,3\}} = 1/2$, $x_{2\leftarrow\{6\}} = 1/2$, Objective is -43259.82. $\lambda_i^* = -7115.31, -11078.32, -12030.71, -13113.32, -10534.51, -13734.11, -10518.83$, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{5,4\}} = 221.27$, $\lambda^*_{\{1,3\}} = 2020.17$, $\lambda^*_{\{4,1\}} = 0.55$, $\lambda^*_{\{5,6\}} = 3219.95$, $\lambda^*_{\{2,3\}} = 455.11$, $\lambda^*_{\{4,1,2\}} = 3813.80$, $\lambda^*_{\{4,2,3\}} = 3079.55$, $\lambda^*_{\{5,1,3\}} = 227.01$, $\lambda^*_{\{5,4,2,3\}} = 358.42$, $\lambda^*_{\{1,7\}} = 515.56$, $\lambda^*_{\{6,7\}} = 3030.75$, $\lambda^*_{\{4,7,2,3\}} = 2172.00$, $\lambda^*_{\{4,6\}} = 28.52$, $\lambda^*_{\{4,1,6\}} = 42.30$, $\lambda^*_{\{4,6,2\}} = 336.33$.

Pricing. New variables: $\mathcal{V}_{12} = (x_{1\leftarrow\{6\}}, -6869.54, 203.45)$, $(x_{4\leftarrow\{1,2,6,7\}}, -1459.12, 0.62)$.

LP. $C = \dot{\bigcup}_{i=1}^{2} C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{12} \mathcal{V}_i$. Solution is $x_{5\leftarrow\emptyset} = 1/2$, $x_{7\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1/2$, $x_{5\leftarrow\{1,4,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1/2$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{2\}} = 1/2$, $x_{2\leftarrow\{1,3\}} = 1/2$, $x_{2\leftarrow\{6\}} = 1/2$, $x_{6\leftarrow\{3\}} = 1/2$, $x_{1\leftarrow\{6\}} = 1/2$, Objective is -43158.09. Dual values $\lambda_i^* = -6940.37, -11078.32, -12002.83, -13113.32, -10534.51, -13762.64, -10518.83$, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{5,4\}} = 220.63$, $\lambda^*_{\{1,3\}} = 2093.38$, $\lambda^*_{\{4,1\}} = 1.19$, $\lambda^*_{\{5,6\}} = 3424.05$, $\lambda^*_{\{2,3\}} = 455.11$, $\lambda^*_{\{4,1,2\}} = 3739.95$, $\lambda^*_{\{4,2,3\}} = 3006.34$, $\lambda^*_{\{5,1,3\}} = 23.54$, $\lambda^*_{\{5,4,2,3\}} = 358.42$, $\lambda^*_{\{1,7\}} = 515.56$, $\lambda^*_{\{6,7\}} = 2855.17$, $\lambda^*_{\{4,7,2,3\}} = 2319.05$, $\lambda^*_{\{3,6,4,1,7\}} = 28.52$, $\lambda^*_{\{4,1,6\}} = 42.30$, $\lambda^*_{\{4,6,2\}} = 336.33$.

Pricing. New variables: $\mathcal{V}_{13} = (x_{1\leftarrow\{2,3,5,6\}}, -47.52, 3.08)$.

LP. $C = \dot{\bigcup}_{i=1}^{2} C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{13} \mathcal{V}_i$. Solution is $x_{5\leftarrow\emptyset} = 1/2$, $x_{7\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{4\leftarrow\{1,2,5,6,7\}} = 1/2$, $x_{5\leftarrow\{1,4,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1/2$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{2\}} = 1/2$, $x_{2\leftarrow\{1,3\}} = 1/2$, $x_{2\leftarrow\{6\}} = 1/2$, $x_{6\leftarrow\{3\}} = 1/2$, $x_{1\leftarrow\{6\}} = 1/2$, Objective is -43158.09. Dual values: $-6940.37, -11078.32, -12005.92, -13113.32, -10534.51, -13762.64, -10518.83$, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{5,4\}} = 220.63$, $\lambda^*_{\{1,3\}} = 2096.47$, $\lambda^*_{\{4,1\}} = 1.19$, $\lambda^*_{\{5,6\}} = 3424.05$, $\lambda^*_{\{2,3\}} = 455.11$, $\lambda^*_{\{4,1,2\}} = 3739.95$, $\lambda^*_{\{4,2,3\}} = 3003.25$, $\lambda^*_{\{5,1,3\}} = 23.54$, $\lambda^*_{\{5,4,2,3\}} = 358.42$, $\lambda^*_{\{1,7\}} = 512.47$, $\lambda^*_{\{6,7\}} = 2855.17$, $\lambda^*_{\{4,7,2,3\}} = 2322.14$, $\lambda^*_{\{3,6,4,1,7\}} = 28.52$, $\lambda^*_{\{4,1,6\}} = 42.30$, $\lambda^*_{\{4,6,2\}} = 336.33$.

Pricing. New variables: None

Separation. New clusters: $C_3 = \{2, 3, 6\}, \{1, 3, 6\}, \{2, 3, 4, 6\}, \{1, 2, 3, 6\}, \{1, 2, 3, 4, 6\}, \{1, 6\}, \{1, 2, 6\}, \{3, 6\}, \{1, 2, 4, 6\}, \{2, 3, 4, 5, 6\}, \{2, 3, 4, 6, 7\}, \{1, 3, 4, 6\}, \{1, 3, 5, 6\}, \{1, 3, 6, 7\}, \{1, 2, 3, 5, 6\}, \{2, 3, 6, 7\}, \{2, 3, 5, 6\}, \{1, 2, 3, 4, 5, 6\}, \{1, 2, 3, 4, 6, 7\}, \{1, 2, 3, 6, 7\}$.

LP. $C = \dot{\bigcup}_{i=1}^{3} C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{13} \mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{4\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\emptyset} = 1/2$, $x_{6\leftarrow\emptyset} = 1/2$, $x_{7\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{2\leftarrow\{1,3,4\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{5\leftarrow\{1,4,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1/2$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{4\leftarrow\{1,3,5,6,7\}} = 1/2$, $x_{2\leftarrow\{4\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$. Objective is -45300.94. Dual values: $-7115.31, -12643.78, -13942.09, -14678.78, -10534.51, -16208.59, -10518.83$, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{5,4\}} = 220.63$, $\lambda^*_{\{1,3\}} = 2190.82$, $\lambda^*_{\{4,1\}} = 1.19$, $\lambda^*_{\{5,6\}} = 3806.03$, $\lambda^*_{\{2,3\}} = 455.11$, $\lambda^*_{\{4,1,2\}} = 3645.60$, $\lambda^*_{\{4,2,3\}} = 2908.90$, $\lambda^*_{\{4,7,2\}} = 301.03$, $\lambda^*_{\{1,7\}} = 512.47$, $\lambda^*_{\{6,7\}} = 2710.47$, $\lambda^*_{\{4,7,2,3\}} = 2194.34$, $\lambda^*_{\{4,6\}} = 70.82$, $\lambda^*_{\{6,1,3\}} = 198.77$, $\lambda^*_{\{4,6,2,3\}} = 2275.69$, $\lambda^*_{\{1,6\}} = 70.54$.

Pricing. New variables: $\mathcal{V}_{14} = (x_{1\leftarrow\{2,3\}}, -106.68, 11.36)$. $(x_{6\leftarrow\{2,3\}}, -13710.19, 23.91)$.

LP. $C = \dot{\bigcup}_{i=1}^{3} C_i$. $\mathcal{V} = \dot{\bigcup}_{i=1}^{14} \mathcal{V}_i$. Solution is $x_{1\leftarrow\emptyset} = 1/2$, $x_{5\leftarrow\emptyset} = 1/2$, $x_{6\leftarrow\emptyset} = 1/2$, $x_{7\leftarrow\emptyset} = 1/2$, $x_{1\leftarrow\{2,3,4,5,6,7\}} = 1/2$, $x_{2\leftarrow\{1,3,4\}} = 1/2$, $x_{3\leftarrow\{1,2\}} = 1/2$, $x_{5\leftarrow\{1,4,6,7\}} = 1/2$, $x_{6\leftarrow\{1,3,4,5,7\}} = 1/2$, $x_{7\leftarrow\{1,3,4,5,6\}} = 1/2$, $x_{4\leftarrow\{1,3,5,6,7\}} = 1/2$, $x_{3\leftarrow\{5,6,7\}} = 1/2$, $x_{4\leftarrow\{2\}} = 1/2$, $x_{2\leftarrow\emptyset} = 1/2$, Objective value is -45300.93. $-7115.31, -12643.76, -13942.09, -14678.76, -10534.51, -16208.59, -10518.83$, $\lambda^*_{\{1,2\}} = 962.04$, $\lambda^*_{\{4,2\}} = 1822.05$, $\lambda^*_{\{5,4\}} = 220.63$, $\lambda^*_{\{1,3\}} = 2190.83$, $\lambda^*_{\{4,1\}} = 1.19$, $\lambda^*_{\{5,6\}} = 3806.03$, $\lambda^*_{\{2,3\}} = 455.11$, $\lambda^*_{\{4,1,2\}} = 3645.59$, $\lambda^*_{\{4,2,3\}} = 2908.89$, $\lambda^*_{\{4,7,2\}} = 301.03$, $\lambda^*_{\{1,7\}} = 512.47$, $\lambda^*_{\{6,7\}} =$

2710.47, $\lambda^*_{\{4,7,2,3\}} = 2194.34$, $\lambda^*_{\{4,6\}} = 70.82$, $\lambda^*_{\{4,6,2\}} = 23.92$, $\lambda^*_{\{6,1,3\}} = 222.69$, $\lambda^*_{\{4,6,2,3\}} = 2251.77$, $\lambda^*_{\{1,6\}} = 46.61$.

Pricing. New variables: None

Separation. New clusters: None

Appendix B. Identifying intervals of non-POPs

In the example given in Section 3.2 we find that the minimal cost parent set for child 7 is $\{1, 3, 4, 5, 6\}$ with a cost of 4800, if 6 is disallowed as a parent then the empty set has minimal cost (cost is 10518), and if 4 and 5 (but not 6) are disallowed then $\{1, 2, 6\}$ has minimal cost (cost is 8036). The relevant best-parent-set constraints are as follows:

$$\neg x_{1\leftsquigarrow 7} \wedge \neg x_{3\leftsquigarrow 7} \wedge \neg x_{4\leftsquigarrow 7} \wedge \neg x_{5\leftsquigarrow 7} \wedge \neg x_{6\leftsquigarrow 7} \rightarrow x_{7\leftarrow\{1,3,4,5,6\}} = 1 \quad (16)$$

$$x_{6\leftsquigarrow 7} \rightarrow x_{7\leftarrow\{\}} = 1 \quad (17)$$

$$x_{4\leftsquigarrow 7} \wedge x_{5\leftsquigarrow 7} \wedge \neg x_{1\leftsquigarrow 7} \wedge \neg x_{2\leftsquigarrow 7} \wedge \neg x_{6\leftsquigarrow 7} \rightarrow x_{7\leftarrow\{1,2,6\}} = 1 \quad (18)$$

We now consider how to generate best-parent-set constraints in a systematic way. Let us represent each best-parent-set constraint as a triple (disallowed parents, minimal cost parent set, cost of minimal cost parent set), so the three constraints (16)–(18) are represented as:

$$(\emptyset, \{1, 3, 4, 5, 6\}, 4800) \quad (19)$$

$$(\{6\}, \emptyset, 10518) \quad (20)$$

$$(\{4, 5\}, \{1, 2, 6\}, 8036) \quad (21)$$

We assume we have access to a function `mincost` which takes a set of disallowed parents D and returns the triple (D, B, c) where B is the minimal cost (i.e. 'best') parent set and c is the minimal cost. So for example from (18) we have $\texttt{mincost}(\{4, 5\}) = (\{4, 5\}, \{1, 2, 6\}, 8036)$. Note that if D has a subset D' with $\texttt{mincost}(D') = (D', B', c')$ and it so happens that $D \cap B' = \emptyset$ then $\texttt{mincost}(D) = (D, B', c')$, since any extra disallowed parents in $D \setminus D'$ are not present in the best parent set B', so if they are disallowed we still get B' as best parent set. Based upon this simple observation, Algorithm 1 generates a set of triples T such that for any $D \subseteq P \setminus \{i\}$ there is exactly one $(D', B', c') \in T$ such that $D' \subseteq D$ and $D \cap B' = \emptyset$ and so $\texttt{mincost}(D)$ is available as (D, B', c'). Fig 7 shows the set of triples output by Algorithm 1 when run for child 7 using the data described in Section 3.2.

$(\emptyset,$	$\{1,3,4,5,6\},$	$4800)$
$(\{1\},$	$\{2,3,4,5,6\},$	$5316)$
$(\{3\},$	$\{1,2,4,5,6\},$	$4801)$
$(\{4\},$	$\{1,2,5,6\},$	$5062)$
$(\{5\},$	$\{1,3,4,6\},$	$7991)$
$(\{6\},$	$\emptyset$	$10518)$
$(\{1,2\},$	$\{3,4,5,6\},$	$5612)$
$(\{1,3\},$	$\{4,5,6\},$	$7990)$
$(\{1,4\},$	$\{2,3,5,6\},$	$5527)$
$(\{1,5\},$	$\{2,3,4,6\},$	$8078)$
$(\{2,3\},$	$\{1,4,5,6\},$	$4801)$
$(\{2,4\},$	$\{1,3,5,6\},$	$5361)$
$(\{3,5\},$	$\{1,4,6\},$	$7991)$
$(\{4,5\},$	$\{1,2,6\},$	$8036)$
$(\{1,2,4\},$	$\{3,5,6\},$	$5722)$
$(\{1,2,5\},$	$\{3,4,6\},$	$8139)$
$(\{1,3,4\},$	$\{2,5,6\},$	$8027)$
$(\{1,3,5\},$	$\{4,6\},$	$8839)$
$(\{1,4,5\},$	$\{2,3,6\},$	$8122)$
$(\{2,3,4\},$	$\{1,5,6\},$	$9677)$
$(\{2,4,5\},$	$\{1,3,6\},$	$8089)$
$(\{1,2,3,4\},$	$\{5,6\},$	$9779)$
$(\{1,2,4,5\},$	$\{3,6\},$	$8161)$
$(\{1,3,4,5\},$	$\{2,6\},$	$8858)$
$(\{2,3,4,5\},$	$\{1,6\},$	$9793)$
$(\{1,2,3,4,5\},$	$\{6\},$	$9872)$

Figure 7: Example output of Algorithm 1

```
1  T ← {mincost(∅)}
2  F ← T
3  while F ≠ ∅ do
4      F′ ← ∅
5      foreach (D̃, B̃, c̃) ∈ F do
6          foreach d ∈ B̃ do
7              D ← D̃ ∪ {d}
8              if ∃(D, B, c) ∈ F′ then
9                  continue                          // mincost(D) already computed
               end
10             if ∃(D′, B′, c′) ∈ T : D′ ⊊ D ∧ D ∩ B′ = ∅ then
11                 continue                  // mincost(D) = (D, B′, c′) already available
               end
12             F′ ← F′ ∪ {mincost(D)}
           end
       end
13     T ← T ∪ F′
14     F ← F′
   end
15 return T
```

Algorithm 1: Finding triples representing minimal cost parent sets subject to some parents being disallowed.