AA222 Final Report: Bayesian Structure Learning and Causal Discovery Using Generalized Disjunctive Programming

Romeo Valentin (*romeov@stanford.edu*)

Spring 2023

Abstract

In this project we provide a new formulation for *Bayesian Network Discovery*, leveraging a novel formulation using *Generalized Disjunctive Programming*, and evaluating the formulation's efficacy. We present the assumptions, formulation, and implementation, and discuss performance results. We also comment on the relations to the more difficult problem of *Causal Structure Identification* and show that under some assumptions our method recovers the correct causal graph.

1 Introduction

Given a dataset of observations for a set of random variables, we may try to infer the probabilistic dependencies between the variables. For example, (i) we can try to predict the effects of a medicine on a patient given the patient's genetic information or current health condition; or (ii), a customer could try to infer the quality and subjective appeal of a wine bottle, given information about the wine's properties like age, pH value, and own preferences.

Inferring such a graph, however, is generally non-trivial, and even shown to be NP-hard in the general case. Nonetheless, several optimization-based approaches exist; commonly either independence-based (e.g. Spirtes, Glymour, and Scheines 1993; Pearl 2009) or score-based (e.g. Heckerman, Geiger, and Chickering 1995; Heckerman, Meek, and Cooper 2006) algorithms are used. We also point to previous work in exact methods, based on Integer Programming; for instance de Campos and Ji (2011).

In this work, we propose a new formulation by combining ideas from Generalized Disjunctive Programming (GDP) with common problem formulations known from Integer Programming and Bayesian Network Discovery. We will see how the GDP formulation leads to a natural formulation of the required constraints, and further allows for further heuristics, like limiting the number of parents, to be encoded directly into the problem. The formulation can then be solved either directly, or by automatically reformulating it as a Mixed-Integer Non-linear Program (MINLP) and solving it with highly optimized, free or commercially available solvers.

In order to set up the problem, we assume that the variable dependencies follow a sparse linear model with additive, non-Gaussian noise. We will see that this formulation allows us to obtain a relatively simple objective function, the minimization of which leads to the optimal structure. However, the optimization problem is constrained by integer and binary constraints, the number of which scales quadratically with the number of variables, but is constant with regard to the number of observations.

Advantages. Advantages of exact solvers, rather than heuristics based ones, include that they are guaranteed to converge to the global optimum, and further continuously provide an estimate of the "optimality gap"; i.e. the gap between the best-found-solution (or "incumbent solution") and a lower bound that has been established. This allows practitioners to estimate at any time how much improvement could still be gained by running the program longer.

The formulation as a GDP also allows for further logical constraints to be added easily. For example, if additional knowledge is available, or can be inferred through statistical (conditional) independence tests, it is straightforward to put additional constraints in the form of logical statements onto the disjuncts. Further, even if not discussed in this work, the GDP (or MINLP) formulation also allows for more complicated non-linear models, rather than simple additive noise models. Indeed, for a formulation $y_i = f(Pa(y_i)) + \delta$ where each variable is given by a *convex* function $f(\cdot)$ of it's parents $Pa(y_i)$, with additive noise, the formulation can still provide relatively fast results, with little change to the formulation or optimization setup necessary.

Finally, since it is possible to automatically convert any GDP formulation to a MINLP formulation, a wide variety of solvers can be applied directly, including highly-optimized multi-threaded commercial solvers.[1] Alternatively, custom solvers can be used and easily extended with heuristics suitable for the concrete problem setup.

1.1 Contributions

In this project, we make the following contributions:

I. Explore how topological graph constraints can be rewritten using the GDP formulation;

II. Introduce a new formulation for finding Bayesian Network Structures given data, and discuss connections to finding Causal Graph Structures; and

III. Showcase the efficacy of the method on two simple datasets.

[1]We are using the Gurobi solver (`https://gurobi.com`), which provides free licenses for academic use.

Accompanying code, including an implementation of the model, is provided at `https://github.com/RomeoV/AA222_FinalProject.jl`.

2 Mathematical Background

Bayesian Networks and Bayesian Network Discovery. *Bayesian Networks* try to model the statistical relations of observational variables. In particular, given observational data with multiple variables, we can try to infer new predictions by *factorizing* the structure and finding sparse probabilitic relations between different variables. *Bayesian Network Discovery* then specifically deals with the problem of finding such a sparse, factorized representation given data. In this setting, an algorithm considers different possible (directed) dependencies between variables, and ranks them according to a *score*, e.g. the Bayesian Information Criterion (Schwarz 1978), or by considering constraint violations (Spirtes, Glymour, and Scheines 1993 , see e.g. PC-Algorithm).

Many specific formulations of this problem exist, but we will consider the formulation given by (Zheng et al. 2018), which consists of a continuous convex term representing the data likelihood, a convex regularization term, and a topological constraint

$$\min_{X_{ij}} \frac{1}{2} tr\left\{(I-X)(I-X)^{\intercal} S\right\} + \lambda \cdot \phi(X) \tag{1}$$

where X represents the edge weights of the implied Bayesian Network graph, which must be directed and acyclic, and S is given by the data. $\phi(X)$ is a regularization term which may be used to induce sparsity. The Directed-Acyclic-Graph (DAG) constraint is generally non-trivial to deal with; Manzour, Küçükyavuz, and Shojaie (2019) propose different techniques from Integer Programming, specifically cutting planes, linear ordering, and topological ordering. Zhang et al. (2022) instead propose reformulating the DAG-constraint as the trace of a matrix-exponential, and approximating it using a truncated power iteration, which can then be numerically optimized. In the following, we will propose a new way to deal with this constraint, by using Generalized Disjunctive Programming.

Mixed-Integer and Generalized Disjunctive Programming. Mixed-Integer Non-Linear Programming (MINLP) is a widely-applicable problem formulation, in which a known objective function is to be minimized using a set of continuous and discrete variables, which are subject to a series of known constraints. If the objective function and constraints are convex, there exist efficient algorithms to solve such problems, which have successfully been applied to problems with thousands of variables (Grossmann and Trespalacios 2013; Duran and Grossmann 1986).

However, writing a problem in a MINLP formulation can be a non-trivial task, and often may loose some structure which could otherwise be used to solve the problem more efficiently. To this end, the related paradigm of *(Generalized) Disjunctive Programming*

(GDP) has been introduced (Bergamini, Aguirre, and Grossmann 2005; Trespalacios and Grossmann 2014; Grossmann and Ruiz 2012; Balas 1979), which allows to encode the notion of *choices* as well as logical constraints directly, directly using so called Disjunctive Constraints. Disjunctive Constraints can be thought of as "either-or" constraints, where we must choose to either abide by constraint-set A or constraint-set B. In Section 3 we will see how the GDP formulation can be used as a powerful yet simple framework to encode complicated relationships and constraints. Finally, we note that any GDP can be automatically reformulated as a MINLP problem. However sometimes it is also possible to exploit the GDP structure directly to improve the efficiency of the optimization algorithms.

3 Mathematical Formulation

In this section, we first discuss the underlying assumptions of our model, as well as the implications on the identifiability of a causal graph structure. Then, we show how we can use Generalized Disjunctive Programming (GDP) to encoder the assumptions and constraints that we are making into a standardized formulation. Finally, we put together the whole model and briefly discuss the steps necessary to solve the problem.

3.1 Mathematical assumptions and implications

In order to find an underlying structure to our data, we first state and justify some assumptions. We model our data naively as a linear model with additive noise

$$\mathcal{D} = \mathcal{X}\mathcal{D} + \Delta.$$

Although this model may often not be accurate, we argue that it provides a reasonable first approximation for many relations in the real world. It also significantly reduces the complexity of the optimization problem, and provides certain identifiability results discussed in the next section. However, we also note that more complicated models are compatible with our algorithm, however are not discussed here.

We further assume that there exists an acyclic relationship between the variables. Cyclic probabilistic models can, for example, be used to model states which are in equilibrium; however we do not consider those here. Finally, we assume that our graphical structure is sparse, and further introduce a heuristic to directly limit the number of parents for any variable.

Implications on causality and causal identifiability We briefly comment on some "stronger" properties of our problem setup beyond Bayesian Network Structures, namely Causal Network Structures. Causal Network Structures are similar to Bayesian Network Structures, but have edges oriented in the "correct" way in the sense that intervening on a variable by drawing it from a new distribution still allows probabilistically correct inference. Wrong model assumptions, and hidden confounding variables, may however still be present.

We first note that, in general, linear model with Gaussian noise are non-identifiable (Peters, Janzing, and Schölkopf 2017 , Thm 4.2) in the sense that the correct causal directions can not be concluded from observational data, even in the limit of infinite data. However, the linear structure makes it identifiable in "almost every other case", e.g. with non-Gaussian noise, or noise with equal variance. We argue that for many problems the noise is indeed not Gaussian.

We further note that minimization of a consistent loss function[2], like the Bayesian Information Criterion, together with the identifyability assumption discussed above, guarantees that the true causal model will be found upon convergence, given that the model assumptions hold, and in the limit of infinite data. We consider this an additional strong motivation for our approach, since we can guarantee convergence to a globally optimal solution, given enough time. However, these insights also leads us to the following intuition: Fundamentally, convergence to the true result is limited by the "degree of identifiability", which loosely relates to how large the class of graph structures is that could have generated the data, and which "collapses" fully in the limit of perfectly Gaussian noise. Therefore, this also limits how fast our search algorithm (or any algorithm) can converge, which in general means that we can not expect particularly fast convergence, even for a small number of variables.

3.2 Using Disjunctive Programming to encode assumptions and constraints

We will now describe how the different components described in the previous section can be naturally formulated using the tools of Generalized Disjunctive Programming.

Encoding the DAG constraint. We use a disjunctive formulation together with an integer-valued topological-ordering constraint to encode the directed-acyclic-graph (DAG) assumption (Section 3.1). For that, notice that for any pair of variables (i, j), one of the following statements is always true:

1. There is a path from variable i to j;
2. there is a path from variable j to i; or
3. there is no path in either direction.

For the first case, we can not necessarily imply that $x_{ij} \neq 0$, however we can imply that $x_{ji} = 0$ and that $o_i < o_j$, where $o \in \{1, \ldots, N\}$ denotes the topological ordering of the variables. The inverse conclusion holds for the second case. For the third case, we can only

[2]Consistency here is used in the sense that minimizing a loss function yields the optimal result for an underlying search problem.

conclude $x_{ij} = x_{ji} = 0$, however put no restrictions on o_i and o_j. We can therefore encode the DAG constraint by adding the disjuncts

$$\begin{bmatrix} Y_{ij}^{(1)} \\ x_{ji} = 0 \\ o_i < o_j \end{bmatrix} \veebar \begin{bmatrix} Y_{ij}^{(2)} \\ x_{ij} = 0 \\ o_j < o_i \end{bmatrix} \veebar \begin{bmatrix} Y_{ij}^{(3)} \\ x_{ij} = 0 \\ x_{ji} = 0 \end{bmatrix} \tag{2}$$

for each unique pair $i \neq j$, i.e. $\forall i \in \{1, \dots, N\}, j \in \{i+1, \dots, N\}$, where $\veebar$ denotes an exclusive-or relationship between the binary variables Y_{ij}^1,Y_{ij}^2 and Y_{ij}^3. For notational convenience, we extend the definition of $Y_{ij}^{(k)}$ to $\mathcal{I}^2$ by further defining $Y_{ji}^{(1)} = Y_{ij}^{(2)}$ and $Y_{ji}^{(3)} = Y_{ij}^{(3)}$.

Constraining the number of parent nodes. Reducing the maximum number of parent nodes for any variable is a common way to reduce the size of the search space. Using the disjunctive formulation from Eq. (2), we can add this constraint in a natural way: Let $P_{\max}$ be the maximum number of parent nodes. Then, we can simply add N constraints

$$\sum_{i \neq j} Y_{ij}^{(1)} \leq P_{\max} \tag{3}$$

with fixed $j \in \{1, \dots, N\}$.

Adding further external knowledge. Using the disjuncts from Eq. (2), we can also add further knowledge, for example from external insights, or derived from statistical properties about the data. For instance, we can fix a variable i as a "sink" node (i.e. the result of a causal process) by allowing no out edges ($Y_{ij}^{(1)} = 0 \quad \forall j$), or similarly as a "root" node ($Y_{ji}^{(1)} = 0 \quad \forall j$), and can also force the graph to be fully connected ($\sum_j Y_{ij}^{(1)} + Y_{ji}^{(1)} \geq 1 \quad \forall i \in \mathcal{I}$).

Encoding the penalty function. We can rewrite both the L_1- and the L_∞-penalties as linear objectives. For the L_∞-penalty, i.e. $\phi(x) = \|x\|_\infty = \max_{ij} x_{ij}$, we can introduce a single auxiliary variable $\xi_\infty > 0$ and constraints

$$-\xi \leq x_{ij} \leq \xi \tag{4}$$

for all (i, j) in $\mathcal{I}^2$, and add $\lambda \cdot \xi$ to the objective function. Similarly, for the L_1-penalty, i.e. $\phi(x) = \sum_{ij} |x_{ij}|$, we introduce a new auxiliary variable $\xi_{ij} > 0$, introduce constraints

$$-\xi_{ij} \leq x_{ij} \leq \xi_{ij} \tag{5}$$

and add $\lambda \cdot \sum_{ij} \xi_{ij}$ to the objective function. Finally, a L_2-penalty may be added without reformulation, i.e. by simply adding $\lambda \cdot \sum_{ij} x_{ij}^2$ to the objective function.

3.3 Resulting problem formulation as Generalized Disjunctive Program

Using the results from the previous sections, we propose the following mathematical formulation for the Bayesian Network Discover Task. Let $D \in \mathbb{R}^{M \times N}$ be the dataset of M observations and dimensionality N, and let $S = \frac{1}{M} D^\intercal D$. Further, let $\mathcal{I} = \{1, \dots, N\}$ be the index set of variables, and let $\tilde{\mathcal{I}}^2$ be the set of unique index pairs, i.e. $\{(i,j) : i \in \mathcal{I}, j \in \{i+1, \dots, N\}\}$. Then, formulate the problem as

$$
\begin{aligned}
&\min_X \frac{1}{2} \{(I - X)(I - X)^\intercal S\} + \lambda \cdot \sum_{ij} \xi_{ij} && \\
&\begin{bmatrix} Y_{ij}^{(1)} \\ x_{ji} = 0 \\ o_i < o_j \end{bmatrix} \veebar \begin{bmatrix} Y_{ij}^{(2)} \\ x_{ij} = 0 \\ o_i > o_j \end{bmatrix} \veebar \begin{bmatrix} Y_{ij}^{(3)} \\ x_{ij} = 0 \\ x_{j,i} = 0 \end{bmatrix} && \forall (i,j) \in \tilde{\mathcal{I}}^2 \\
&Y_{i,i}^{(3)} = 1 && \forall i \in \mathcal{I} \\
&\sum_{j \in \mathcal{I}} Y_{ij}^{(1)} + \sum_{j \in \mathcal{I}} Y_{j,i}^{(2)} \geq 1 && \forall i \in \mathcal{I} \\
&\sum_{i \neq j} Y_{ij}^{(1)} \leq P_{\max} && \forall j \in \mathcal{I} \\
&-\xi_{ij} \leq x_{ij} \leq \xi_{ij} && \forall (i,j) \in \mathcal{I}^2 \\
&o_i \in \mathcal{I} && \forall i \in \mathcal{I} \\
&\xi_{i,j} \in [0, \max(|\underline{x}|, |\bar{x}|)] && \forall (i,j) \in \mathcal{I}^2 \\
&x_{i,j} \in [\underline{x}, \bar{x}] && \forall (i,j) \in \mathcal{I}^2.
\end{aligned}
\tag{6}
$$

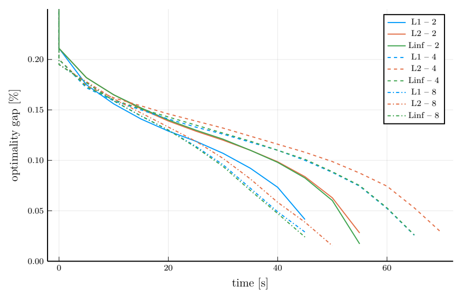
with a slight abuse of notation for $Y_{j,i}$ as introduced in Section 3.2, where the parameters x_{ij} are bounded by $\underline{x}$ and $\bar{x}$ and the number of parents per node is bound by $P_{\max}$.

4 Experimental Results

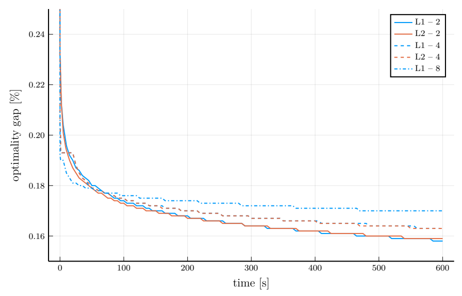
We apply the formulation introduced in the previous section to two datasets; one with 8 and one with 12 variables, and show convergence plots over time for each. Then, we apply the formulation to a large dataset with 50 variables and report time until a "reasonable" solution has been found.

The model is implemented using the *JuMP.jl* library[3], additionally leveraging the *DisjunctiveProgramming.jl* library[4] to implement and reformulate the disjuncts. We use the BigM method with $M = 10$ for reformulation to the MINLP, which slightly outperformed the Hull reformulation method. Variable upper and lower bounds are chosen as $(-10, 10)$,

[3]`https://jump.dev`
[4]`https://github.com/hdavid16/DisjunctiveProgramming.jl`



(a) Small dataset with 8 variables.



(b) Small dataset with 12 variables.

Figure 1: Convergence results on the two datasets, with a timelimit of 10 minutes.

we regularize the parameters with the L_1-penalty and a factor of $\lambda = 0.1$. We use the Gurobi solver[5] to solve the resulting MINLP problem, and have run the experiments on a 11th Gen Intel(R) Core(TM) i7-11800H @ 2.30GHz with 16 threads and 32GiB of RAM.

Fig. 1 shows the convergence results of the formulation on the two datasets, with different choices of regularization and maximum number of parents. We can see that the small problem is solved to optimality after approximately 60 seconds, regardless of the parameter choices. Interestingly, we can observe that the setting with a large number of parents converges fastest; however this is not reproduced for the medium dataset. Further, we can observe on both datasets that L_1-regularization outperforms the others. For the medium dataset, we can see that a reasonable initial result is achieved quickly, however further convergence is slow. Finally, for the large dataset, an initial feasible solution is found after 30 seconds, and the convergence curve flattens out after about 60 seconds, with an optimality gap of 14%.

5 Outlook and Discussion

We have seen that Bayesian Structure Learning can be formulated using tools from Generalized Disjuncive Programming and solved using established MINLP methods. Under some assumptions, we have also argued that not only a Bayesian Network can be found, but indeed the correct Causal Structure will be recovered; however the problem is computationally ill-posed without further restrictions. Nonetheless, reasonable solutions can be found in relatively short time, even for problems with a larger amount of variables (≈ 50).

In the future, we propose investigating extending the formulation beyond linear models, and evaluating both theoretical and performance results. Further, statistical tests can provide (conditional) independence tests before any structure fitting, which may be included as constraints to the formulation.

[5]`https://gurobi.com`

Bibliography

Balas, Egon. 1979. "Disjunctive Programming." In *Annals of Discrete Mathematics.* Discrete Optimization II. Elsevier.

Bergamini, Maria Lorena, Pio Aguirre, and Ignacio Grossmann. 2005. "Logic-Based Outer Approximation for Globally Optimal Synthesis of Process Networks." *Computers & Chemical Engineering*, no. 9 (August).

Campos, Cassio P. de, and Qiang Ji. 2011. "Efficient Structure Learning of Bayesian Networks Using Constraints." *Journal of Machine Learning Research*, no. 20.

Duran, Marco A., and Ignacio E. Grossmann. 1986. "An Outer-Approximation Algorithm for a Class of Mixed-Integer Nonlinear Programs." *Mathematical Programming*, no. 3 (October).

Grossmann, Ignacio E., and Juan P. Ruiz. 2012. "Generalized Disjunctive Programming: A Framework for Formulation and Alternative Algorithms for MINLP Optimization." In *Mixed Integer Nonlinear Programming.* The IMA Volumes in Mathematics and Its Applications. Springer.

Grossmann, Ignacio E., and Francisco Trespalacios. 2013. "Systematic Modeling of Discrete-Continuous Optimization Models through Generalized Disjunctive Programming." *Aiche Journal*, no. 9.

Heckerman, David, Dan Geiger, and David M. Chickering. 1995. "Learning Bayesian Networks: The Combination of Knowledge and Statistical Data." *Machine Learning*, no. 3 (September).

Heckerman, David, Christopher Meek, and Gregory Cooper. 2006. "A Bayesian Approach to Causal Discovery." In *Innovations in Machine Learning: Theory and Applications.* Studies in Fuzziness and Soft Computing. Springer.

Manzour, Hasan, Simge Küçükyavuz, and Ali Shojaie. 2019. "Integer Programming for Learning Directed Acyclic Graphs from Continuous Data." April 23, 2019. `https://arxiv.org/abs/1904.10574`.

Pearl, Judea. 2009. *Causality.* 2nd ed. Cambridge University Press.

Peters, Jonas, Dominik Janzing, and Bernhard Schölkopf. 2017. *Elements of Causal Inference: Foundations and Learning Algorithms.* Adaptive Computation and Machine Learning Series. MIT Press.

Schwarz, Gideon. 1978. "Estimating the Dimension of a Model." *The Annals of Statistics*, no. 2. `https://www.jstor.org/stable/2958889`.

Spirtes, Peter, Clark Glymour, and Richard Scheines. 1993. *Causation, Prediction, and Search.* Lecture Notes in Statistics. Springer.

Trespalacios, Francisco, and Ignacio E. Grossmann. 2014. "Review of Mixed-Integer Nonlinear and Generalized Disjunctive Programming Methods." *Chemie Ingenieur Technik*, no. 7.

Zhang, Zhen, Ignavier Ng, Dong Gong, Yuhang Liu, Ehsan M. Abbasnejad, Mingming Gong, Kun Zhang, and Javen Qinfeng Shi. 2022. "Truncated Matrix Power Iteration for Differentiable DAG Learning." December 20, 2022. https://arxiv.org/abs/2208.14571.

Zheng, Xun, Bryon Aragam, Pradeep K Ravikumar, and Eric P Xing. 2018. "DAGs with NO TEARS: Continuous Optimization for Structure Learning." In *Advances in Neural Information Processing Systems*. Curran Associates, Inc.