A Hybrid Approach to Bayesian Network Structure Learning From Data With GOBNILP

Zhi Peen Su

March 2023

Abstract

A Bayesian network is a probabilistic graphical model which uses a directed acyclic graph (DAG) to represent a set of variables and their conditional dependencies. The nodes in the DAG represent variables and the edges between the nodes indicates some sort of conditional dependency with each other, and the arrow which links one node to another represents the direction of causality [21]. Bayesian networks have been widely used in the field artificial intelligence, medical diagnosis [27], etc. to help make predictions by performing causal inference. This paper presents a hybrid algorithm for learning the structure of Bayesian networks from data by combining the PC algorithm with GOBNILP [8]. The resulting hybrid algorithm aims to reduce the computation time while being able to maintain the accuracy of the learned graph.

Contents

1 Introduction

Prior to the introduction of Bayesian networks, many textbooks on statistics and probability theory suggest that the most optimal way to build a strong foundation of probabilistic knowledge is by defining a joint probability distribution, denoted by $P(x_0, x_1, ..., x_{n-1})$, that covers all propositions and their combinations. However, creating a joint probability distribution for n number of variables requires a table with 2^n entries, which can be both inefficient to store and compute the marginal probabilities [21]. This is where Bayesian networks come in as they aim to efficiently capture the relationships between variables in a joint probability distribution [19].

However, reality tends to be disappointing as obtaining the factorized joint probability distribution for the network at hand is often not possible. Furthermore, identifying structural properties such as node interconnections, parent sets, edge directions, and other relevant parameters can be a challenging and tedious task. Which leads to the task of learning Bayesian network structures from data.

The task of learning the most optimal structure of a Bayesian network through data is known as Bayesian network structure learning. Given a directed acyclic graph (DAG) G, parameters Θ from a data set of D with n number of variables, the task is performed in 2 steps in a Bayesian method [31].

$$P(G, \Theta \mid D) = P(G \mid D)P(\Theta \mid G, D)$$

1. $P(G, \Theta \mid D)$ is the learning of the network from data
2. $P(G \mid D)$ is the learning of the structure of the network from data
3. $P(\Theta \mid G, D)$ is the learning of the parameters of the DAG

However, the learning of $P(G \mid D)$ is known to be extremely challenging [31] and the algorithms which are tasked to solve this problem falls under 3 main categories: the score-based approach, the constraint-based approach, and the hybrid approach.

Score-based Approach The score-based approach (sometimes known as search-and-score), works in a way that it searches over the space of available DAGs to obtain the DAG with the highest score which best fits the data. To begin the search process, the algorithm decides a scoring function used to evaluate the network [6].

Constraint-based Approach The constraint-based approach works by identifying conditional independences (CI) between nodes in the graph and removes edges the moment some CI is found.

Hybrid Approach The hybrid approach combines both aspects of the score-based approach and constraint-based approach to learn the graph.

1.1 The Structure of Bayesian Networks

A Bayesian network is a directed acyclic graph (DAG), where the Bayesian network given by $B = \{(V, E), \Theta\}$ and the graph $G = (V, E)$. In the graph, each vertex, V, represents some random variable in the data. The edge, E, which connects the vertices, represents some dependent relationship. Attached to each vertex, is a **conditional probability distribution**, Θ.

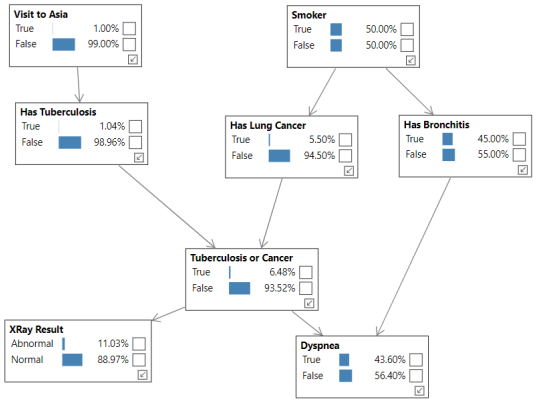
Definition 1 ***Conditional Probability Distribution*** *By Bayes Theorem, given 2 joint distributions of random variables X and Y, The conditional probability distribution of Y given X is the probability distribution of Y when X is observed to be some value [20].*

Given a joint probability distribution $P(x_1, x_2, ..., x_n)$, "the structure of the Bayesian network starts by choosing x_1 as a root node and assign to it the marginal probability of $P(x_1)$, which is dictated by $P(x_1, x_2, ..., x_n)$. We then do the same for x_2 and form a node for x_2. If there exists some form of dependent relationship between x_1 and x_2, where x_2 is dependent on x_1, then an edge which connects x_1 and x_2 is formed with an arrow in the direction of x_1 to x_2. The following probability is then represented by the expression $P(x_2|x_1)$. However, if the relationship between x_1 andx_2 is not dependent, then an edge between x_1 and x_2 will cease to exist, and x_2 will have the prior probability $P(x_2)$ attached to it [21]." Upon reaching the variable x_n, each node in the DAG is associated with a conditional probability distribution given by its parents in the DAG. For each Bayesian network, the joint probability distribution of the network given by $P(x_1, x_2, ..., x_n)$ is factorized by the **chain rule** to

$$\prod_{i=1} P(x_i|\Pi_{X_i})$$

where Π_{X_i} is the parent set of x_i.

Definition 2 ***(Chain Rule)*** *The chain rule states that if we have a set of n events, $E = \{x_0, x_1, ..., x_{n-1}\}$ then the joint probability of E can be written as a product of n conditional probabilities such that $P(x_0, x_1, ..., x_{n-1}) = P(x_{n-1} \mid x_{n-2}, ..., x_1, x_0)\ P(x_{n-2} \mid x_{n-3}, ..., x_1, x_0)\ ...\ P(x_1 \mid x_0)\ P(x_0)$ [21].*



(a) Asia Network

Figure 1: Example of the Asia Bayesian Network Taken From [23]

For example, the distribution corresponding to the DAG in Figure 1 can be written by observing their causalities such that:
$P(x_i \mid \Pi_{x_i}) =$
P(Visit to Asia) P(Smoker)
P(Has Tuberculosis | Visit to Asia)
P(Has Lung Cancer | Is a Smoker)
P(Has Bronchitis | Is a Smoker)
P(Has Tuberculosis or Cancer | Has Lung Cancer, Has Tuberculosis)
P(Dyspnea | Has Tuberculosis or Cancer, Has Bronchitis)
P(XRay Result | Has Tuberculosis or Cancer)

2 Related Works

The learning of Bayesian networks from large amount of data is shown by Chickering to be an NP-hard problem, even if the parent limit of the network is 3 [4]. As the nature of the problem is NP-hard, it is not exactly possible to perform an exhaustive search over the space of available DAGs and obtain the best DAG given the data, unless the network is extremely small, $\mid V \mid \leq 10$. This is true even for medium-sized networks where the lower bound for $\mid V \mid$ is 20. In which case, research around this area have been towards developing algorithms to learn the structure of the graph to find the best approximation towards the true DAG. These methods fall under 3 main categories.

2.1 The Score-Based Approach

When it comes to score-based approach, the algorithm first chooses a scoring function before performing a search over the network. There are 2 categories of scoring function, The Bayesian scoring family or the information theory scoring family. The idea behind the Bayesian scoring function is that it takes the approach of computing the posterior probability distribution by starting from a prior probability distribution on all the possible networks, which is $P(B \mid D)$, the best network is the one which maximizes the posterior probability [2]. The Bayesian scoring family consists score functions such as the Bayesian Dirichlet (BD) and their variants Bayesian Dirichlet estimation (BDE) and Bayesian Dirichlet estimated uniform (BDeu).

On the other hand, many of the information theory score functions takes the form of a penalized log-likelihood function. These scoring functions work by measuring the amount of information gained from observing the data and then adding a penalty term to the log-likelihood. Popular penalized scoring functions include the Minimum Description Length (MDL), Akaike Information Criterion (AIC), and Bayesian Information Criterion (BIC), among others [2]. By adding a penalty term to the score, these functions discourage overly complex models and favor simpler ones that explain the data well [33]. An example of a score function is AIC. It takes into account the number of estimated parameters, k, and the maximum value of the likelihood function, L. AIC is calculated using the formula:

$$AIC = 2k - 2L$$

The preferred model among a set of candidate models is the one with the minimum AIC value. AIC rewards a good fit, as measured by the likelihood function, but also includes a penalty that increases as the number of estimated parameters increases. This penalty discourages overfitting, which is when the model fits the data too closely and does not generalize well to new data [24].

Once a scoring function is decided, a search algorithm is then used to search over all possible DAGs and assign a score to each of them. The most commonly used search algorithm used can either be a local search algorithm. The local search algorithm looks at one possible network structure at a time, and explores the neighborhood of the current network by making small changes to it, such as adding or removing a single edge. The algorithm then evaluates the score of each network in the neighborhood and selects the best one as the new current network. This process is repeated until no further improvements can be made [25]. Examples of local search algorithms are the tabu search algorithm [12] or hill-climbing algorithm [17]. Greedy search algorithms on the other hand start with an empty network structure and iteratively add or remove edges based on the current score of the network, an example of a greedy search algorithm is the K2 algorithm [6].

Additionally, recent advances in the score-based algorithms for learning Bayesian networks uses the Integer Linear Programming (ILP) approach. The ILP approach encodes the structure of the graph, the scoring, and ensuring

that the graph does not form cycles into a linear programming problem [8], more about this will be talked upon in section 3.6.

2.2 The Constraint-Based Approach

Based on the definition of the **d-separation** criterion [21], the approach work by performing some CI test to identify CI between two random variables X and Y given a third set of random variables Z,

$$(X \perp Y|Z)$$

where $X, Y \notin Z$ for every CI test within the data. Upon performing the CI tests, a binary output is then obtained of the following form; "An edge could exist between X and Y given Z" or "An edge does not exist between X and Y given Z". If the former answer is obtained, the edge is not removed. However if the latter answer is obtained, the edge between X and Y is ruled out. To construct a DAG, an arrow is drawn from one variable to another based on the direction of causality.

Definition 3 ***d-separation*** *X and Y are d-separated if X is independent of Y given a set of variables, Z,* $(X \perp Y \mid Z)$ *where* $X, Y \notin Z$.

Pearl and Verma introduced one of the earliest constraint-based approaches for learning causal networks with the Inductive Causation (IC) algorithm [11]. The IC algorithm starts by learning the skeletal structure of the DAG by finding any form of CI between a pair of variables, X and Y. If no form of CI is found, then an edge is formed connecting X and Y. Do this for every possible pairs of variables to form a partial undirected graph. The latter part of the algorithm orients the edges by checking that whether the set of variables Z which d-separates X and Y is adjacent to X and Y.

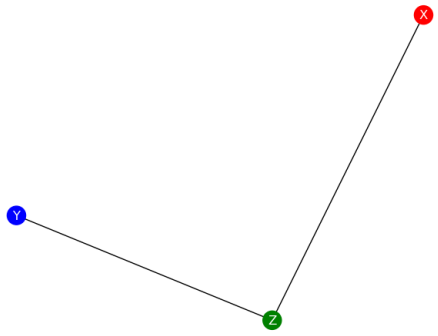


Figure 2: This is a Subgraph of the Full Graph

Figure 2 shows a subgraph of vertices X, Y, Z. If it is discovered that Z is an element in the set which d-seperates X and Y, the edges are then oriented such that $(X- > Z < -Y)$

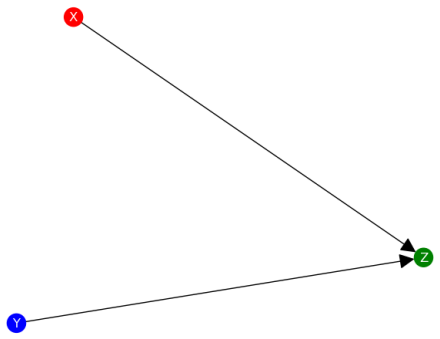


Figure 3: Directed **v-structure** of the Subgraph

Definition 4 *A v-structure in the graph takes the following form* $(X- > Z < -Y)$.

However, if Z is not an element in the d-seperating set of X and Y, the algorithm then orients as many edges as possible while making sure no new V-structures and directed cycles are being created. While the algorithm itself takes a fairly straightforward approach, the IC algorithm is not very practical to use in larger problems. This is because the complexity of the algorithm is heavily bounded in the skeletal learning phase, IC_{SL}, which by brute force requires an exponential search for the seperating set [11]. Later on, Spirtes and Clark realized that the complexity of the IC_{SL} algorithm could be reduced to a polynomial time algorithm by reducing the search space of CIs and subsequently developed the predictive causation (PC) algorithm [28].

The PC algorithm is considered as a cutting-edge causal discovery algorithm, especially for data involving a large number of variables. Unlike the IC algorithm, which is subject to significant constraints imposed by IC_{SL}. The PC algorithm starts by forming a complete undirected graph and removes edges adjacent to variables X and Y, initiating the process from lower orders of conditional independence. However, just like the IC algorithm, the complexity of the PC algorithm is still bound to the skeletal-learning phase. The worst-case scenario of the algorithm happens when no CI relationships exist between the variables in the data and is bounded by

$$2\binom{n}{2}\sum_{i=0}^{k}\binom{n-1}{i}$$

which is then bounded by

$$\frac{n^2(n-1)^{k-1}}{(k-1)!}$$

where k is the maximum degree of any vertex and n is $\mid V \mid$, as shown in [28]. In such a case, the algorithm has to test all possible CI relationships between pairs of variables which would be no different than using IC_{SL}. However, an important benefit of the PC algorithm is that it does not need to search for higher orders of CI and that the algorithm can be terminated as soon as it is judged to be inefficient, while the edge orientation phase of the algorithm remains the same as that of the IC algorithm. As such, the skeletal learning part of the PC algorithm, PC_{SL} will be used in the hybrid algorithm to optimize GOBNILP.

2.3 The Hybrid Approach

As the name implies, the hybrid approach works by combining both aspects of score-based algorithm and constraint-based algorithm to form a hybrid algorithm. Typically the hybrid learning approach has two phases, The first approach called the **restrict phase** where it rules out certain edges with the constraint-based algorithm to reduce the search space of candidate DAGs. The second phase, the **maximize phase**, implements the score-based algorithm to find the optimal DAG based on the score function used.

One such example would be the Max-Min Hill-Climbing (MMHC) algorithm. The MMHC algorithm first learns the skeleton of the graph by using the constraint-based algorithm, the Max-Min Parents and Child (MMPC) algorithm then performs edge orientation using the greedy Bayesian-scoring hill climbing search. On the other hand, the hybrid HPC (H2PC) algorithm uses the PC_{SL} algorithm, while also using the Bayesian-scoring hill climbing search for the score-based algorithm [18]. Further examples include the Inter-IAMB algorithm which uses the Incremental Association Markov Blanket as the constraint-based component and a greedy hill climbing search for the score-based component [3].

2.4 Which Algorithm Performs Better?

Scutari performed an experiment to find out which of the 3 algorithms yielded better results. He used 2 score-based algorithms (tabu search [12], simulated annealing with BIC [9], greedy equivalent search with log BDeu [5]), 3 constraint-based algorithms (PC [28], Grow Shrink [26], Inter-IAMB [3]), and 2 hybrid algorithms (MMHC [18], RSMAX2 [32]) According to Scutari's research, constraint-based algorithms were found to be more accurate when working with smaller datasets compared to score-based algorithms, while also being as fast as hybrid algorithms. However, Scutari found that the performance of the tabu search algorithm demonstrates higher accuracy and faster learning compared to both constraint-based algorithms and hybrid algorithms [31].

2.5 Bayesian Model Averaging

Lastly, there is one more approach worth bringing up which takes a different approach to the 3 main methods. When it comes to model uncertainty, there are 2 main methods of approaching this, the first being Bayesian model selection for which the score-based algorithm, the constraint-based algorithm and the hybrid algorithm all fall under. The Bayesian model selection chooses the model the algorithm deems best, then conduct causal inference with the assumption that the model chosen is the true model. The Bayesian Model Averaging (BMA) approach on the other hand involves averaging over multiple DAGs in the DAG space and takes into account the uncertainty in the model selection process. The BMA approach allows for the incorporation of multiple models and their associated uncertainties, resulting in a more robust and reliable causal inference. However, the BMA approach can be computationally expensive and requires careful consideration of the prior distribution used for the model parameters [14].

3 Methods

3.1 Hypothesis Test

Constraint-based algorithms performs hypothesis tests to identify CI between variables, allowing the skeleton of the graph to be learned. Given 2 variables, X and Y, and a set of variables which might d-separate X and Y, called Z. The null hypothesis (h_0) and alternative hypothesis (h_1) are first stated:

h_0 : X and Y are independent given Z

h_1 : X and Y are not independent given Z

The null hypothesis assumes there is no relationship between the variables while the alternative hypothesis assumes there might be some form of dependency between the variables.

However, in the case that the hypothesis test rejects h_0, proceed to perform the hypothesis test on higher **orders** of CI.

Definition 5 ***(CI Order)*** *The order of a CI test is equal to the cardinality of the set of variables Z in a hypothesis test of* $(X \perp Y \mid Z)$.

3.2 The chi-squared χ^2 test

When it comes to hypothesis testing, commonly used hypothesis test includes Pearson's chi-squared χ^2 test [10], the log-likelihood (G^2) test to test for independence between 2 discrete variables, or the mutual information (MI) test. Below is an example of the chi-squared test on a dataset:

Given table 1 as above and we wanted to test whether variables 'Three' and 'Seven' were independent of each other given the data.

1. Start by defining h_0 and h_1,

One	Two	Three	Four	Five	Six	Seven	Eight
1	1	0	0	0	1	0	1
1	0	0	0	0	0	1	0
0	0	1	0	0	0	1	0
1	0	1	0	0	0	1	0
1	0	1	0	0	0	0	1
1	0	1	0	0	0	1	1
1	0	1	0	0	0	1	1
1	0	1	0	0	0	0	1
1	0	1	0	0	0	0	1
1	0	1	0	0	0	1	0

Table 1: A Table which Represents 8 Variables and their Data

h_0 : Variables 'Three' and 'Seven' are independent

h_1 : Variables 'Three' and 'Seven' are not independent

Once defined the hypotheses, Compute the contingency table of the variables.

2. Create a contingency table with the counts of the categories for variables 'Three' and 'Seven'.

To create this table, count the number of occurrences of 1's and 0's per variable.

	Three	Seven	R_{sum}
Number of 0's	2	4	6
Number of 1's	8	6	14
C_{sum}	10	10	Grand Total = 20

3. Compute the expected frequencies for each cell in the contingency table under the assumption of independence. This can be done by multiplying the row total and column total for each cell and then dividing by the **Grand Total**, which in this case is 20. expected frequency = (row total x column total) / grand total

In the case of the first cell where the observed number of 0's for 'Three' = 2, compute the expected frequency, f_i, by doing $f_{three=1} = \frac{6 \cdot 10}{20} = 3$. Do this for each cell and obtain the expected frequency table:

4. Finally, given the contingency tables for the observed data and the expected data, proceed to compute the test statistic $\chi^2_{test-statistic}$ for the data.

	Three	Seven
Number of 0's	3	3
Number of 1's	7	7

The equation to compute $\chi^2_{test-statistic}$

$$\chi^2_{test-statistic} = \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

In which case, once that is done every variable, the follow equation is obtained:

$$\chi^2_{test-statistic} = \frac{(2-3)^2}{3} + ... + \frac{(6-7)^2}{7} \tag{2}$$

$$\chi^2_{test-statistic} = 2/3 + 2/7 = 0.953 \tag{3}$$

Lastly, the dof of for a χ^2 test with a contingency table of size (m x n) is $dof = (m-1)\cdot(n-1) = (2-1)\cdot(2-1) = 1$

Assuming this test is performed at a 5% significance level,check the critical value for a χ^2 test with 5% significance level and degree of freedom of 1, which in this case $\chi^2 = 3.841$. Since $\chi^2_{test-statistic}$ for the χ^2 test $< \chi^2_{dof=1,SL=0.05}$, the null hypothesis test is not accepted but not completely rejected.

3.3 Significance Threshold of the χ^2 Test

The significance threshold of the a hypothesis test, often denoted by the symbol alpha (α), plays an important role in determining which CI test are statistically significant in a way that determines whether or not an edge in the graph should be removed. In most cases, the value of α is set to be equal to 0.05. Increasing the value of $\alpha > 0.05$ (which also reduces the value of the p-value) would cause acceptance of less null hypotheses, which in turn removes fewer edges from the graph. Similarly, reducing the value of $\alpha < 0.05$ would result in more edges being removed from the graph.

However, there are trade-offs for different values of α with the accuracy of the hybrid graph and the runtime. When the value of α is increased, the probability of committing a **Type I error** increases together with it. In other words, by increasing the significance threshold, there is the off chance that there might be some edge in graph which should not be there, but due to this error from the test, the edge is kept in the graph. Subsequently, reducing the value of α increases the probability of commiting a **Type II error**, which just removes an edge which should have been there.

Although an α value of 0.05 is considered 'reasonable' in most scenarios, it may not be suitable for certain networks. Therefore, it is necessary to evaluate the score of the DAG with different levels of significance thresholds.

Definition 6 ***(Type I error)*** *A false-positive error, which means that the test rejects the null hypothesis given that the null hypothesis is true.*

Definition 7 ***(Type II error)*** *A true-negative error, which means that the test accepts the null hypothesis given that the null hypothesis is false.*

3.4 The PC algorithm

The first step is to identify the confidence intervals (CIs) of the graph, which can be achieved by using the PC_{SL} algorithm. This algorithm is combined with GOBNILP, to form the hybrid algorithm.

The PC Algorithm [28]

1. **Input** : A dataset with a set of variables V
2. Create a complete undirected graph G wth the set of vertices, V.
3. $n = 0$.
4. **repeat**
   (a) **repeat**
      i. Set X and Y to be variables which are adjacent in the graph
      ii. Initialize S_{XY} to be the set of neighbors of X and Y
      iii. Initialize Z as the combinations of variables of S_{XY} where the cardinality of S_{XY} is n.
      iv. Perform a hypothesis test of independence for X, Y, given Z:
         If $(X \perp Y \mid Z)$, exit the loop and move onto step 5, else continue.
   (b) Do this for all pairs of adjacent variables in the graph until an empty graph is formed or the cardinality of $S_{XY} < n$
   (c) $n = n + 1$.
5. Once all the orders of CI has been searched through, a graph with some disconnected edges is then obtained.
   (a) For a triple of vertices in the graph X, Y, Z_0, If and only if $Z_0 \in S_{XY}$, orient the edges such that $X \rightarrow Z_0 \leftarrow Y$. Do this for all triple of vertices in the graph.
6. **repeat**
   (a) If $X \rightarrow Y$, Y and Z are adjacent, X and Z are not adjacent, and there is no arrowhead at Y, orient $Y - Z$ as $Y \rightarrow Z$
   (b) If there is a directed path going from X to Y, and there is an edge connecting X and Y, orient edges $X - Y$ such that $X \rightarrow Y$.
7. The last step is just to prevent cycles from being formed while being able to orient as many edges as possible.

It should be noted that for the implementation of the hybrid algorithm, only steps 1 to 5 will be utilized. As an illustration, consider the input of a dataset consisting of 8 variables such that V = {One, Two, ... , Eight} into the algorithm. The initial step involves constructing a complete undirected graph as follows.

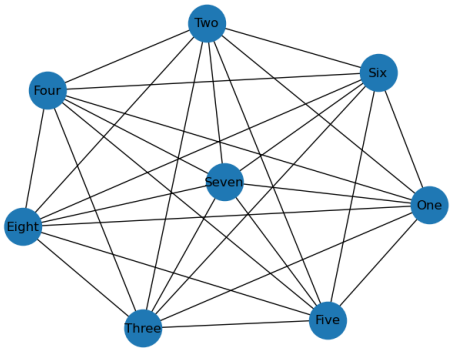


Figure 4: Complete Undirected Graph, Each Vertex Representing a Variable

The independent relationships are then obtained for the 0^{th} order CI and is as follows:

- One $\perp$ Four
- One $\perp$ Five
- Two $\perp$ Four
- Two $\perp$ Five
- Three $\perp$ Four
- Four $\perp$ Six
- Four $\perp$ Seven
- Four $\perp$ Eight

A total of 8 independent relationships, proceed to d-separate those pairs of variables, resulting in the following graph:

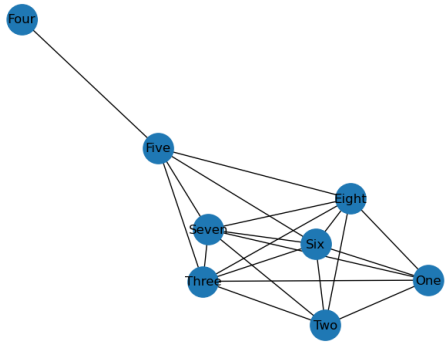


Figure 5: Graph After Removing 0^{th} Order CI

Now test for 1^{st} order and once again obtain the follow set of variables with an independent relationship:

- One $\perp$ Six
- One $\perp$ Seven
- Two $\perp$ Three
- Two $\perp$ Seven
- Two $\perp$ Eight
- Three $\perp$ Five
- Three $\perp$ Six
- Three $\perp$ Seven
- Five $\perp$ Seven
- Five $\perp$ Eight
- Seven $\perp$ Eight

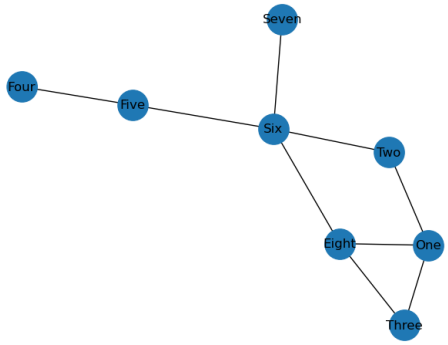


Figure 6: Graph After Removing 1st Order CI

'Repeating the process for $n = 2, 3, ..., S_{max}$, a total of 19 independent relationships were obtained in addition to the 11 obtained in the 0th order CI. The resulting graph is shown below:'

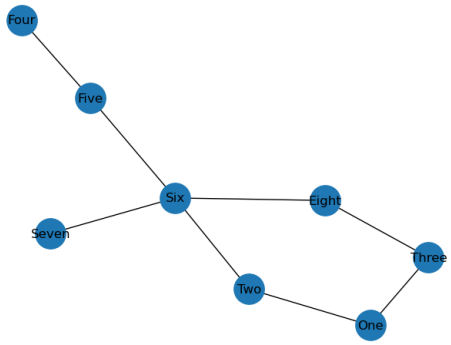


Figure 7: Final Undirected Graph

At this point, 20 edges from the graph has been removed, resulting in the graph having a remaining of 8 edges. While it might be possible to brute force smaller datasets such as this one, it will not be computationally feasible to perform the test on higher orders of CI, even on medium-sized networks. This will be shown why in section 4.3.

3.5 Networks Used

In this project, the performance of the hybrid algorithm (PC_{SL} + GOBNILP) was evaluated with networks of different sizes. The data used were synthetic data obtained from [30] which uses a sampling method known as **forward sampling**. 12 networks were used to evaluate the performance of the algorithm, the

networks evaluated consists of small-sized networks such as the Asia network which was used for clinical diagnosis [22] | V_{asia} | = 8. Medium-sized networks such as the Alarm network which was used to implement an alarm message system for patient monitoring [16] | V_{Alarm} | = 37, large-sized networks such as the HailFinder network which was used to forecast severe weather conditions [1] | $V_{HailFinder}$ |= 56, etc. To sum it up, the networks employed to evaluate the hybrid algorithm are presented in the table below.

Network	Number of Variables
Asia	8
Child	20
Insurance	27
Water	32
Alarm	37
Barley	48
HailFinder	56
Hepar2	70
Andes	223
Link	724
PathFinder	109
Munin2	1003

Table 2: Summary of Networks Used, Obtained From [30]

It is worth noting that although the networks have been arranged in ascending order according to the number of variables, it is important to note that the computational time required to learn the structure for each network may not necessarily increase linearly. This is because each network contains a different number of data points, and the computation time required to learn the structure may vary based on the average number of data points per variable in the network.

Definition 8 ***(Forward Sampling)*** *A method of generating samples for every variable in a Bayesian network such that each sample is generated in proportion to their own probability.*

3.6 GOBNILP

Globally Optimal Bayesian Network learning using Integer Linear Programming (GOBNILP) is a program developed by Cussens which acts as a score-based approach to learning the network structure which will also be the score-based part of the hybrid algorithm. As opposed to other score-based algorithms, GOBNILP approaches the learning of Bayesian network structures by encoding DAGs into a integer program. This is done by creating binary 'family' variables for each node v and for parents set W, where W is the parents set of v, such that $I(W->v)$. If the cardinality of the parent set is 0, then $W = \emptyset$ [8].

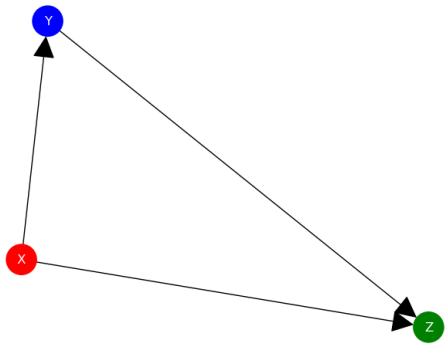


Figure 8: DAG with 3 vertices

GOBNILP will encode the DAG as follows in an integer program in the following form:

$T_X \leftarrow \{\}$	$T_X \leftarrow \{Y\}$	$T_X \leftarrow \{Z\}$	$\mathrm{T}_X \leftarrow \{Y, Z\}$
1	0	0	0
$T_Y \leftarrow \{\}$	$T_Y \leftarrow \{X\}$	$T_Y \leftarrow \{Z\}$	$\mathrm{T}_Y \leftarrow \{X, Z\}$
0	1	0	0
$T_Z \leftarrow \{\}$	$T_Z \leftarrow \{X\}$	$T_Z \leftarrow \{Y\}$	$\mathrm{T}_Z \leftarrow \{X, Y\}$
0	0	0	1

Table 3: Figure 8 Encoded in an ILP [7]

However, suppose the PC_{SL} algorithm was used to find that $X \perp Y$ and produces the graph:

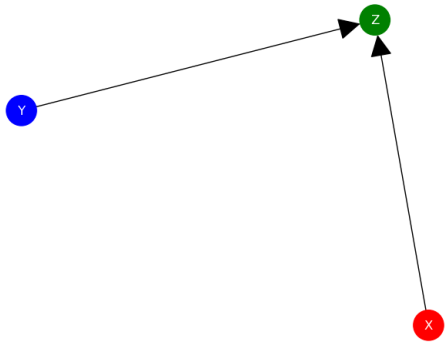


Figure 9: Graph After Discovering $X \perp Y$

GOBNILP then encodes the graph in a compact manner that reduces computational requirements, as the resulting table is smaller and easier to perform computations on.

$T_X \leftarrow \{\}$	$T_X \leftarrow \{Z\}$	$T_Y \leftarrow \{\}$	$\mathrm{T}_Y \leftarrow \{Z\}$
1	0	1	0
$T_Z \leftarrow \{\}$	$T_Z \leftarrow \{X\}$	$T_Z \leftarrow \{Y\}$	$\mathrm{T}_Z \leftarrow \{X, Y\}$
0	0	0	1

Table 4: Encoding After Removing the Edge Between X and Y

Additionally, the score metric used by GOBNILP for Discrete Bayesian Networks is the BDeu score, which is a **decomposable score** and is formally defined by the equation below:

$$PenalizedBDeu(X_i, B, D) = \sum_{j}^{q_i} \sum_{k}^{r_i} log \frac{P(D_{ijk} \mid D_{ij})}{P(D_{ijk} \mid D_{ij}, \alpha_{ij})} \tag{4}$$

- q_i = number of possible values of the parents Π_{x_i}
- r_i = number of possible values of X_i
- D_{ijk} = number of times when $X_i = k$ and $\Pi_{x_i} = j$
- α_{ij} is the hyperparameter that controls the strength of the prior distribution used in Bayesian inference α [24].

Definition 9 ***Decomposable*** *The measure on a structure if decomposable if it can be written as a product of measures or sum when taking the log of the function, each of which only involves a function of one vertex and their parents which allows for more efficient computing [13].*

3.7 Evaluating Structural Accuracy

The primary objective of this project is to minimize the runtime of GOBNILP. However, it is equally important to ensure that the network's structural accuracy is not compromised. Otherwise, all edges from the graph could simply be removed, resulting in an empty graph that would satisfy the objective with maximum efficiency but would be of no practical use.

The F1-score evalutes the structural accuracy by taking type I (false positive) and type II (false negative) errors from the skeleton of the baseline graph and compare it to the skeleton of the hybrid graph [29]. In this case, the baseline graph is the graph produced by GOBNILP alone, while the hybrid graph will be the graph produced after adding constraints. Additionally, the type I error will be the number of mistakes that an edge from the baseline graph given that it should not be there, being in the learned graph will be denoted as FP. The type II error will be the number of mistakes that an edge from the baseline graph

which should be there, but is not present in the learned graph will be denoted as FN. Formally, the F1-score is defined to be the harmonic mean of **precision** and **recall**. But it can also be written as

$$F1_{score} = \frac{2TP}{2TP + FP + FN} \tag{5}$$

Definition 10 ***Precision*** *The number of true positive results divided by the total number of positive results [34].*

Definition 11 ***Recall*** *The number of true positive results divided by the total number of false negative results [34].*

4 Results

4.1 Runtime of GOBNILP

Before proceeding with the experiments, it is important to take into account a few key considerations, such as the overall runtime of the algorithm. The overall runtime (T_{hybrid}) will be defined as the sum of the time taken to perform edge restriction and the time taken to find a good scoring network .

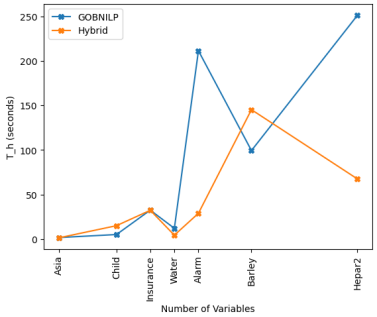
It is important to find a balance between these two factors, cause if the restriction phase (PC_{SL}) of the algorithm takes such a long time to run that GOBNILP is better off without it, then it defeats the purpose of using a hybrid algorithm in the first place. Furthermore, the time required to generate a graph using GOBNILP will be referred to as T_{score}, the graph obtained by GOBNILP without applying any constraints will be denoted by G_{score}, and the graph produced by the hybrid algorithm (the integration of GOBNILP and PC_{SL}) will be denoted as G_{hybrid}.

In the table below, the comparison of the differences in time (measured in seconds) and the accuracy (F1-score) between GOBNILP and the hybrid algorithm is shown. These tests were carried out at the 1st order CI, a 5% significance threshold, and a parent size limit of 2. The F1-score is calculated by comparing the undirected skeleton of graphs G_{score} and G_{hybrid}. The reduction column is represented by a percentage to show the reduction from T_{score} to T_{hybrid}, if reduction is positive then it represents a reduction in time, if positive then it shows an increase in time.

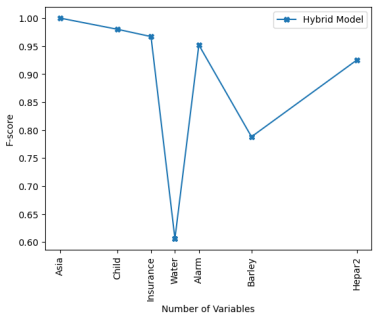
If a network requires an unfeasible amount of time, it implies that GOBNILP was unable to render a graph within 3 hours or even crashed during the process. However, if GOBNILP was able to render a graph given the network, a comparison between the runtime of GOBNILP and the hybrid algorithm was visualized through a graph.

Network	T_{score}	T_{hybrid}	Reduction	F1-score
Asia	1.79	1.45	+19.00	1.000
Child	5.21	15.14	-190.60	0.980
Insurance	32.55	32.39	+0.49	0.967
Water	12.17	4.47	+63.27	0.606
Alarm	211.31	29.01	+86.27	0.952
Barley	99.35	145.23	-46.18	0.860
HailFinder	197.91	156.48	+20.93	0.788
Hepar2	251.09	67.50	+73.20	0.925
Andes	Not Feasible	1969.47	Null	Null
Link	Not Feasible	Not Feasible	Null	Null
Pathfinder	Not Feasible	Not Feasible	Null	Null
Munin2	Not Feasible	Not Feasible	Null	Null

Table 5: This Table Shows the Time Taken **Measured in Seconds** to Learn the Graph by Gobnilp and the Hybrid Algorithm, Along With the F1-score Which Compares G_{score} and G_{hybrid}



(a) Comparing Runtime of GOBNILP and the Hybrid Algorithm



(b) Comparing G_{hybrid} to G_{score} with F1-score

Figure 10: Runtime Comparisons and Model Accuracy

The results indicates that while hybrid algorithm has a significant difference in T_{hybrid}, the accuracy of G_{hybrid} compared to G_{score} seems to have the opposite effect. In other words, there is a trade-off between runtime and accuracy.

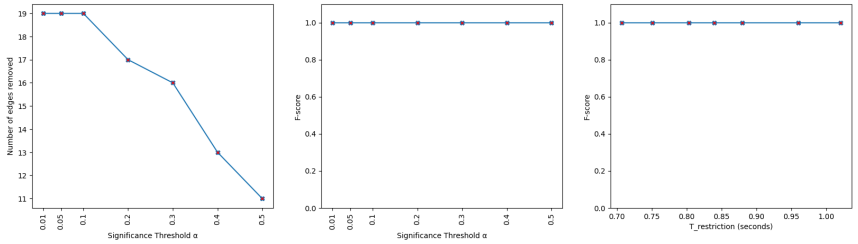
In addition, the study revealed that factors such as the significance threshold and the order of CI used in the hybrid algorithm significantly affect the trade-off between runtime and accuracy. The investigation aimed to address outliers in the data, such as the sharp decline in the F1-score value for the Water network.

4.2 The Effects of the Significance Threshold

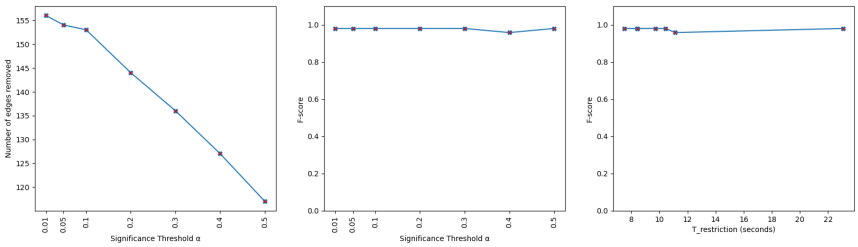
To examine the influence of the significance threshold on the model, an analysis was carried out to explore the balance between runtime and accuracy. The hybrid algorithm was applied with varying thresholds, and the effects on these metrics were observed. Furthermore, the number of edges removed from the graph was measured at different values of α.

$$\alpha = \{0.01, 0.05, 0.10, 0.20, 0.30, 0.40, 0.50\}$$

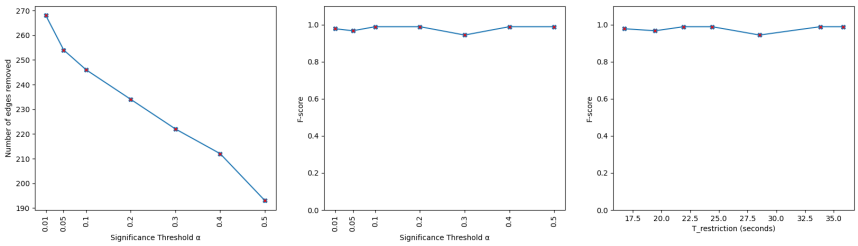
To provide a visual representation between the significance threshold, runtime, and structural accuracy, several figures were produced.



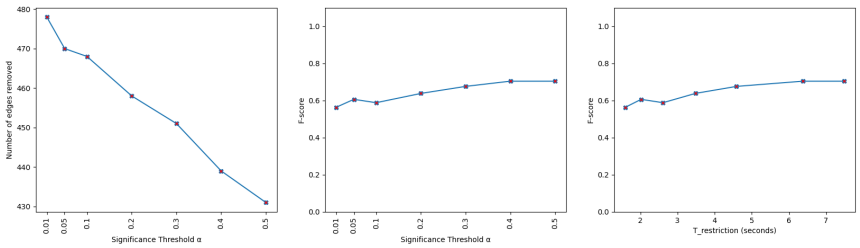
(a) Asia Network



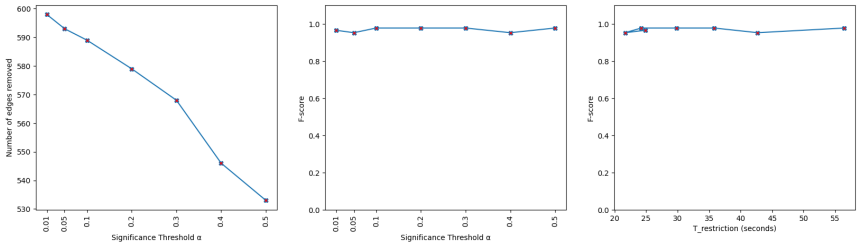
(b) Child Network



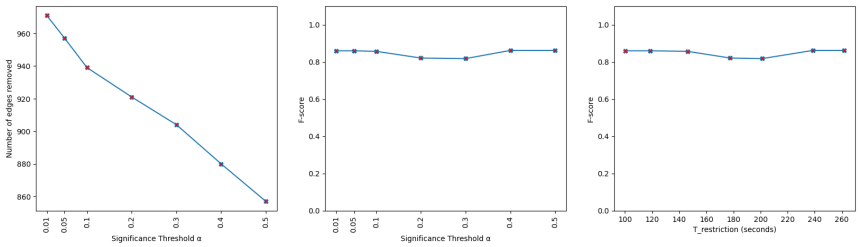
(c) Insurance Network



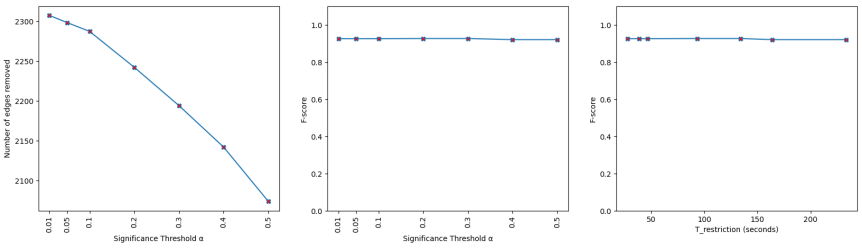
(d) Water Network



(e) Alarm Network



(f) Barley Network



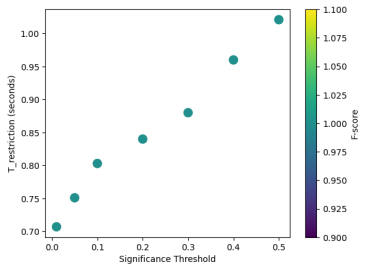
(g) Hepar2 Network

Figure 11: Runtime Comparisons and Model Accuracy

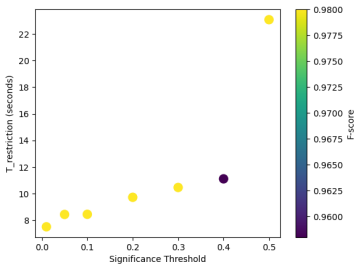
In eacn subfigure, there are 3 plots, each of which respresents some form of correlation. From the left, the first plot shows the correlation between the significance threshold and the number of edges removed, the second plot shows the correlation between significance threshold and F1-score of the graph, while

the third one shows the correlation between significance threshold and runtime. The general trend for the significance threshold against the number of edges removed in the graph seems to be that of a somewhat-linear relationship and remains consistent throughout the results. On cases with smaller-sized networks such as the Asia network, there does not seem to be any form of impact on their accuracy while having a slight improvement in runtime. However, as GOBNILP is able to run quite fast on smaller-sized networks, smaller-sized networks will not be the most critical factor in terms of runtime. In the case of larger networks, such as Barley and Hepar2, the influence of the significance threshold on the algorithm's performance becomes more noticeable. While the impact of the significance threshold on the F1-score and runtime of the algorithm can vary across different networks, the Water network in particular stands out as an outlier with a significantly lower F1-score compared to other networks, even at higher significance thresholds, and a relatively short runtime compared to some other networks. At first glance, the reason for this outlier is not obvious and unfortunately the cause of this outlier is unknown. But one thing for sure, this indicates that the hybrid algorithm used to learn the network structure is not suitable for the Water network. Whereas for most other networks, there is consistency between the F1-score and the runtime in the sense that the deviation of the F1-score with increasing runtime is minor.

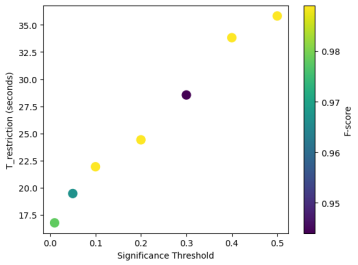
However, the question that arises is how to optimize the performance of the algorithm to achieve maximum F1-score and minimum runtime. To investigate this, a scatter plot was created for each network.



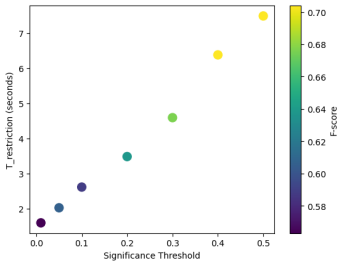
(a) Asia Network



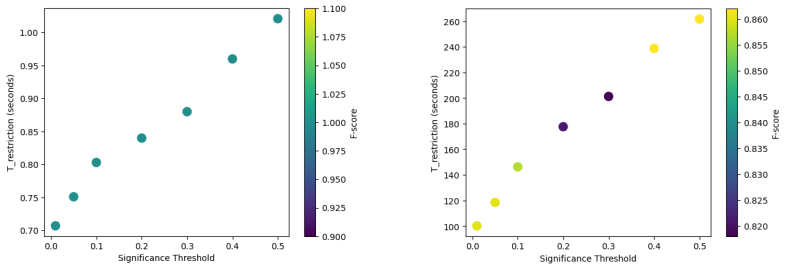
(b) Child Network



(c) Insurance Network

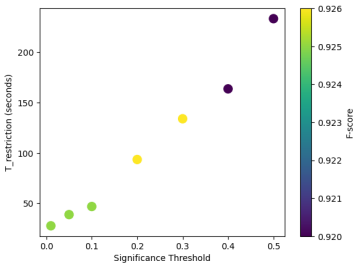


(d) Water Network



(e) Alarm Network

(f) Barley Network



(g) Hepar2 Network

Figure 12: Scatter Plots Comparing Runtime and Significance Threshold

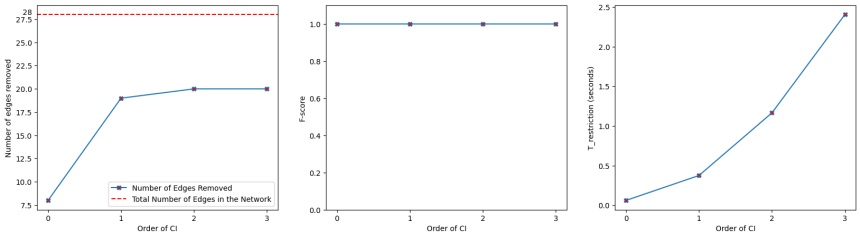
In these plots, the colours represents the value of the F1-score, the y-axis represents the runtime and the x-axis represents the significance threshold. For example, in the case of the Alarm network, the optimal significance threshold value lies between 0.1 and 0.2. Analyzing the scatter plots for each network can provide valuable insights into achieving optimal trade-offs between runtime and F1-score, based on the significance threshold.

4.3 The Effects of the Order of CI

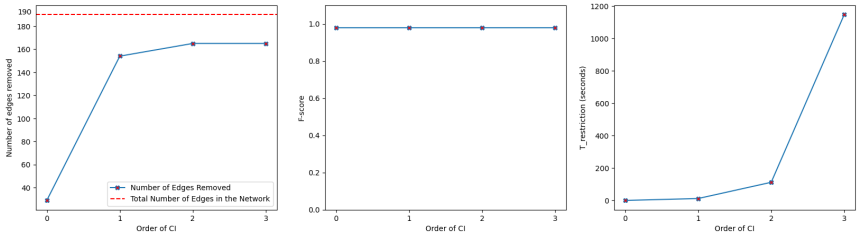
To determine the conditional independence between X and Y given a set of variables Z, denoted as

$$(X \perp Y \mid Z)$$

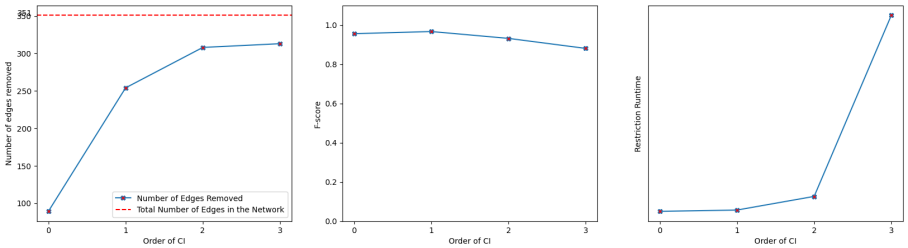
The order of CI, is defined to be the cardinality of Z $n =\mid Z \mid$ when performing the independence test. To put it simply, the order of CI is the order in which the PC_{SL} algorithm identifies d-separation between two variables, starting from 0 and increasing up to the maximum cardinality of Z, denoted as Z_{max}, such that $Z = \{0, 1, ..., Z_{max}\}$. Similar to section 4.2, some basic plots is shown to determine how the order of CI plays a role in the algorithm.



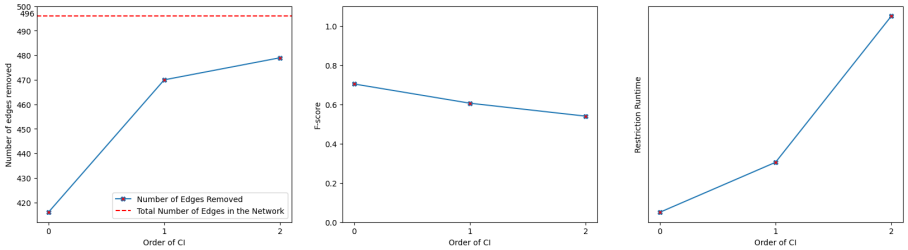
(a) Asia Network



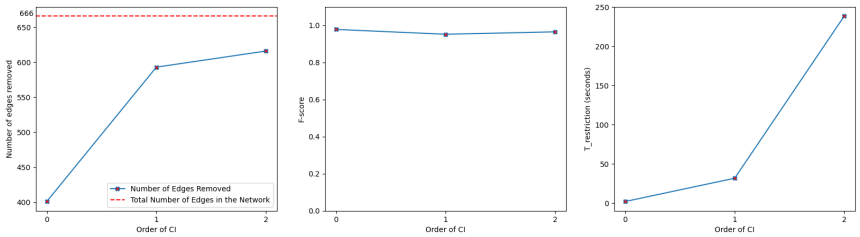
(b) Child Network



(c) Insurance Network



(d) Water Network



(e) Alarm Network

Figure 13: The Effects of the Order of CI on Runtime and Accuracy

Once more, figure 13 shows 3 different plots, in the first one it shows how the number of edges restricted shows a diminishing return with the order of CI, with the red dotted line being the original number of edges in the graph. the second one showing the effects of the CI order on the F1-score and the third showing the effects of the CI order on the runtime of the **restriction** phase of the algorithm. However, due to the limitation that the runtime on networks larger than the alarm network would not run in a feasible amount of time for increasing orders of CI, those networks have been excluded from the results. Although the amount of data in each plot is limited, there are noticeable trends that could justify the order of CI having a diminishing return to the number of edges restricted, while the runtime increases exponentially. The F1-score does not vary significantly except for the Water network.

While it is clear that orders of CI higher than 1 is most often not worth considering, the following question remains, is it better to use 0^{th} order CI or 1^{st} order CI. To answer this question,A simple plot is shown to compare the **overall** runtime on different networks.

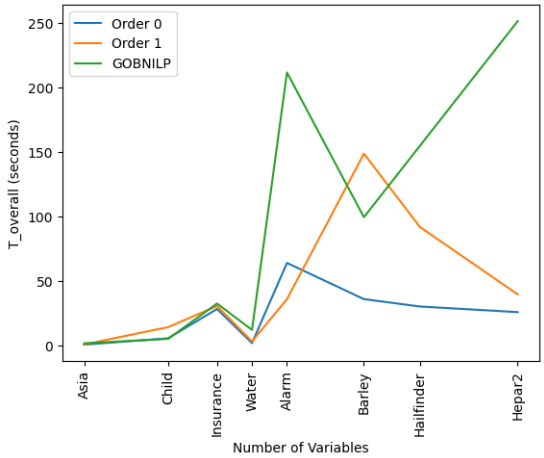


Figure 14: Comparing GOBNILP to Different CI Orders

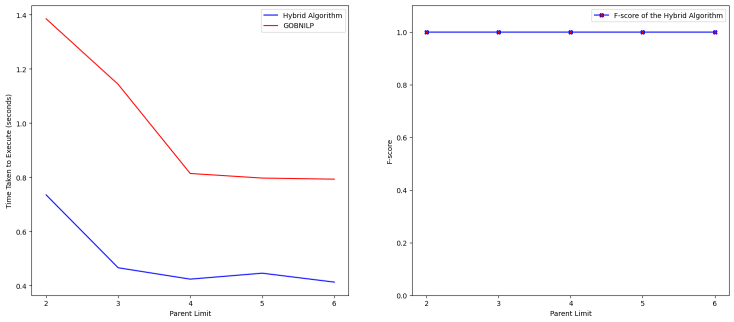
Surprisingly, the plot shows that running the algorithm at the 0^{th} order CI produces the fastest overall runtime. To summarise this section, while the PC_{SL} algorithm allows to search for higher orders of CIs in a 'reasonable' amount of time, it may not always be necessary to do so. Depending on the network used and the context, using lower order CIs might just be sufficient for producing accurate and meaningful results, while also reducing the computational burden.

4.4 The Hybrid Algorithm on Higher Parent Limit

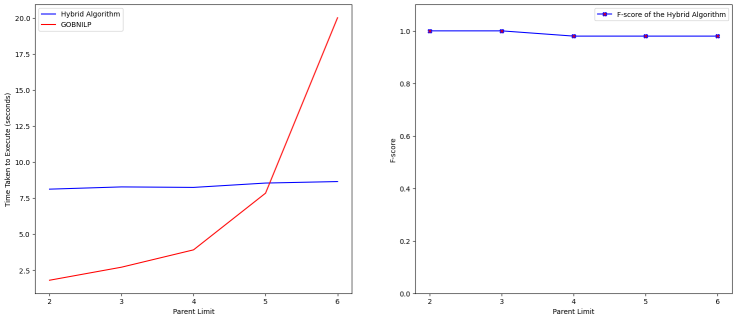
Lastly, The performance of the hybrid algorithm was evaluated on networks with higher parent limit. Previously, all of the experiments were performed on a parent limit on size 2. This was necessary because increasing the parent limit for GOBNILP would increase the search space of available DAGs massively. However, in this section, the performance of the hybrid algorithm is compared to GOBNILP on different parent limits.

$$palim = \{2, 3, 4, 5, 6\}$$

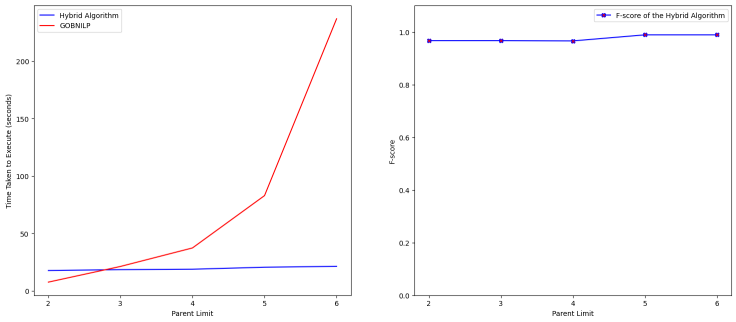
Once again, some figures were plotted to compare the performance of the hybrid algorithm to GOBNILP on higher parent limits. This experiment was carried out with the 1^{st} order CI and the value of α was set to equal 0.05.



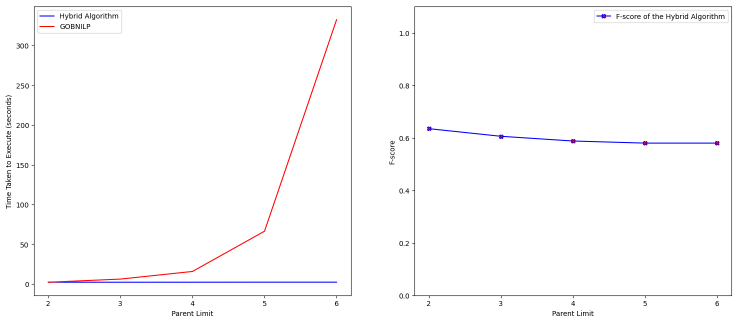
(a) Asia Network



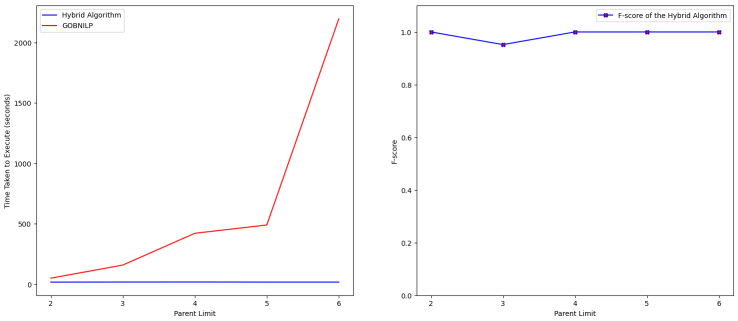
(b) Child Network



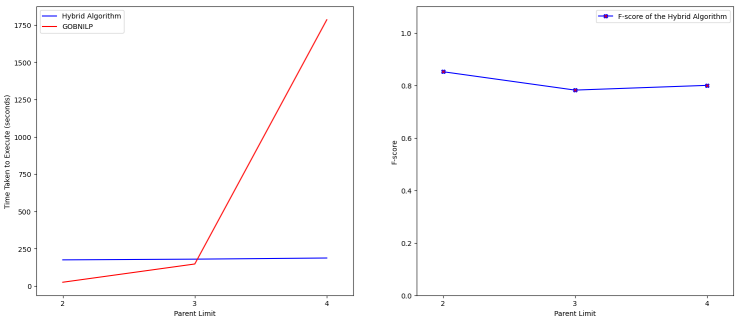
(c) Insurance Network



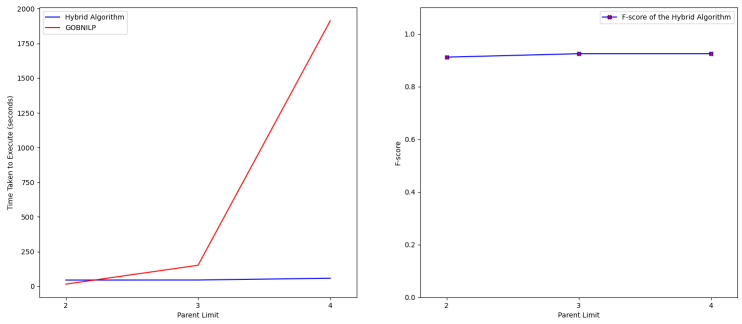
(d) Water Network



(e) Alarm Network



(f) Barley Network



(g) Hepar2 Network

Figure 15: The Difference in Runtime Between T_{hybrid} And T_{score}, as Well as the Accuracy of G_{hybrid} Compared to G_{score}.

Figure 15 consists of two plots for each network. The first plot displays the difference in the runtime of GOBNILP and the hybrid algorithm and the second plot shows the F1-score of G_{hybrid} compared to G_{score} for the same parent limits. As the parent limit increases, GOBNILP shows an exponential trend in the runtime, while the runtime of the hybrid algorithm remains relatively consistent. Additionally, there seems to be very minor fluctuation in terms of G_{hybrid} and G_{score}. In conclusion, while it is not completely safe to say that the hybrid algorithm shows better overall performance on higher parent limits, the consistent performance of the hybrid algorithm in terms of runtime and F1-score suggests that it may be a more reliable choice than GOBNILP, particularly when dealing with larger datasets and higher parent limits.

5 Discussion

Overall, the implementation of the hybrid algorithm was a success with a significant improvement in overall runtime while the graph accuracy of the hybrid algorithm shows quite good of a performance compared to the graph produced by GOBNILP. However, there are some limitations of the hybrid algorithm and the data used which still needs to be acknowledged.

First off, the lack of computational power to run the algorithm on larger networks. While the overall runtime of GOBNILP has shown considerable amounts of improvement, it still takes a significant amount of time to output results on networks such that $\mid V \mid \geq 100$, which was why results on networks larger than Hepar2 have been omitted from the results. This implies that for larger sized networks, it is uncertain if the graph produced by the hybrid algorithm would produce similar structures or if something completely unexpected occurs similarly with the Water network. Although this issue could be alleviated by using better performance computers (supercomputers), it may not be feasible often due to the high cost and limited availability of such resources. In addition,

even with access to supercomputers, there is still a limit to the size of networks that can be analyzed in a reasonable amount of time. Unfortunately, this is a common problem when it comes to combinatorial optimization, so the best solution would be to develop more efficient algorithms which can handle larger data while still providing accurate results within a reasonable amount of time.

Secondly, the method used to evaluate the accuracy of G_{hybrid} compared to G_{score}. Relying solely on the F1-score as a scoring metric is insufficient to evaluate the accuracy, because the F1-score only evaluates the undirected skeleton of the graphs, which does not take edge orientation into account. While there already exists a different comparison metric called the Structural Hamming Distance (SHM) which does this, the main idea behind SHM is that it takes into account every operations needed to transform one graph onto another. Such operations include edge orientation reversal, removing edges, adding edges, etc. However, since GOBNILP is a score-based algorithm, it returns a DAG which maximizes the score.

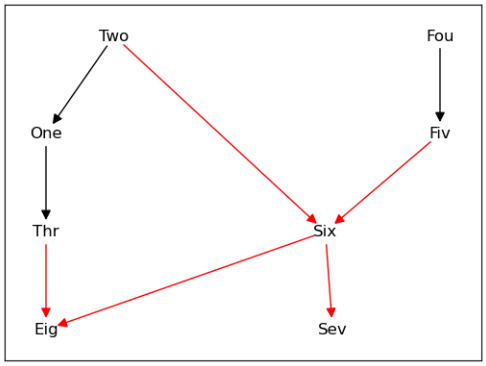


Figure 16: GOBNILP on the Asia Network

It is worth noting that the DAGs produced by GOBNILP consist of two types of arrows. The red arrows indicate a unidirectional relationship between variables, while the black arrows represent a bidirectional relationship, in other words the black arrows could point in either direction and it would still maximize the score given by GOBNILP. With that in mind, it is not exactly possible to use the SHM. A feasible solution to this issue would be to use more score-methods to compare 2 graphs which takes into account and is able to work with the graph given by GOBNILP.

Lastly, note that throughout this experiment, only discrete Bayesian networks were used. Therefore It is also worth exploring other types of Bayesian networks such as Gaussian Bayesian networks or hybrid Bayesian networks, which can contain both discrete and continuous variables. Gaussian Bayesian networks would also require a different scoring function (BGe) compared to the ones used in discrete Bayesian networks. Additionally, hybrid Bayesian networks can be more challenging to learn due to their mixed variables, which can

require more advanced algorithms to produce accurate models.

6 Conclusion and Future Works

Overall, the hybrid algorithm developed in this study demonstrates the potential to optimize GOBNILP for learning Bayesian networks. The results suggest that the PC_{SL} algorithm and GOBNILP can complement each other to overcome their respective limitations and achieve better performance. Nevertheless, the hybrid approach is not a jack-of-all-trades kind of algorithm. Further investigations are needed to address the challenges of learning Bayesian networks for large-scale datasets.

One possible direction for future work is to explore the use of other constraint-based algorithms, such as the PC-stable algorithm, which has been shown to enhance the reliability of causal discovery by further testing for CI after the skeleton of the partially oriented graph is produced [15], and combine it with GOBNILP to see how it fares against this algorithm. Another approach is to compare the performance of different score-based methods in the hybrid approach to GOBNILP, such as MMHC, K2, or tabu search, to determine their relative advantages and trade-offs.

Finally, the hybrid algorithm developed in this study provides a promising framework for learning the structure of Bayesian networks from data, and future research can build on this work to advance the field of causal inference and their applications in various domains.

References

[1] W. Edwards A. Murphy B. Abramson, J. Brown and R. L. Winkler. Hailfinder: A bayesian system for forecasting severe weather. pages 57–71, 1996.

[2] Alexandra M Carvalho. Scoring functions for learning bayesian networks. pages 1–48, 2009.

[3] Kyeol Chang, Junghye Lee, Chi-Hyuck Jun, and Hoeil Chung. Interleaved incremental association markov blanket as a potential feature selection method for improving accuracy in near-infrared spectroscopic analysis. *Talanta*, 178:348–354, 2018.

[4] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of bayesian networks is np-hard. *J. Mach. Learn. Res.*, 5:1287–1330, dec 2004.

[5] Max Chickering. Statistically efficient greedy equivalence search. 124:241–249, 03–06 Aug 2020.

[6] G. Cooper and E. Herskovitz. A bayesian method for the induction of probabilistic networks from data, machine learning. 1992.

[7] J. Cussens. Branch-price-and-cut for causal discovery. 2023.

[8] James Cussens. Bayesian network learning with cutting planes. *CoRR*, abs/1202.3713, 2012.

[9] Yavuz Eren, İbrahim B. Küçükdemiral, and İlker Üstoğlu. Chapter 2 - introduction to optimization. pages 27–74, 2017.

[10] Karl Pearson F.R.S. On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling. pages 157–175, 1900.

[11] Hector Geffner, Rina Dechter, and Joseph Y. Halpern, editors. *Probabilistic and Causal Inference: The Works of Judea Pearl*, volume 36. Association for Computing Machinery, New York, NY, USA, 1 edition, 2022.

[12] Fred Glover. Tabu search - part 1. pages 190–206, 1989.

[13] Geiger D. Chickering D.M. Learning Bayesian networks Heckerman, D. The combination of knowledge and statistical data. pages 197–243, 1995.

[14] van den Bergh D Wagenmakers E-J Hinne M, Gronau QF. A conceptual introduction to bayesian model averaging. pages 200–215, 2020.

[15] Nadir Sella Hervé Isambert Honghao Li, Vincent Cabeli. Pc algorithm with consistent separating sets. 2019.

[16] R. M. Chavez I. A. Beinlich, H. J. Suermondt and G. F. Cooper. The alarm monitoring system: A case study with two probabilistic inference techniques for belief networks. pages 247–256, 1989.

[17] C. F. Aliferis I. Tsamardinos, L. E. Brown. The max-min hill-climbing bayesian network structure learning algorithm. pages 31–78, 2006.

[18] Laura E. Brown Constantin F. Aliferis Ioannis Tsamardinos. The max-min hill-climbing bayesian network structure learning algorithm. pages 31–78, 2006.

[19] Changhe Yuan James Cussens, Brandon Malone. Tutorial on optimal algorithms for learning bayesian networks. 2013.

[20] James Joyce. Bayes' theorem. 2019.

[21] Pearl Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1988.

[22] S. Lauritzen and D. Spiegelhalter. Local computation with probabilities on graphical structures and their application to expert systems (with discussion). 50:157–224, 1988.

[23] Bayes Server Limited. Bayesian network software.

[24] Malone B. Yuan C. Liu, Z. Empirical evaluation of scoring functions for bayesian network model selection. 2012.

[25] Jose Puerta Luis Campos, Juan Fernández-Luna. Local search methods for learning bayesian networks using a modified neighborhood in the space of dags. 2527, 2002.

[26] Dimitris Margaritis. Learning bayesian network model structure from data. 2003.

[27] Scott McLachlan, Kudakwashe Dube, Graham A Hitman, Norman E Fenton, and Evangelia Kyrimi. Bayesian networks in healthcare: Distribution by medical condition. *Artificial Intelligence in Medicine*, 107:101912, 2020.

[28] Richard Scheines Peter Spirtes, Clark Glymour. *Causation, Prediction, and Search*. Springer New York, NY, 1993.

[29] David M. W. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2020.

[30] Marco Scutari. The bayesian network repository. 2023.

[31] Marco Scutari, Catharina Elisabeth Graafland, and José Manuel Gutiérrez. Who learns better bayesian network structures: Constraint-based, score-based or hybrid algorithms? In Václav Kratochvíl and Milan Studený, editors, *Proceedings of the Ninth International Conference on Probabilistic Graphical Models*, volume 72 of *Proceedings of Machine Learning Research*, pages 416–427. PMLR, 11–14 Sep 2018.

[32] Marco Scutari, Phil Howell, David J Balding, and Ian Mackay. Multiple Quantitative Trait Analysis Using Bayesian Networks. *Genetics*, 198(1):129–137, 09 2014.

[33] Marco Taboga. Model selection criteria. 2021.

[34] Kai Ming Ting. Precision and recall. pages 781–781, 2010.