Constraint Based Approach to Learning Bayesian Networks

Zhi Peen Su

March 2023

1 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 how each nodes are dependent on the other [1]. Bayesian networks have been widely used in artificial intelligence, medical diagnosis, decision-making processes, allowing us to model complex systems, make predictions and infer causal relationships between variables. When learning Bayesian networks from data, there are 2 main approaches. One of which is called a score-based approach and the other being a constraint-based approach. The score-based approach works by defining a criterion to evaluate how well a Bayesian network fits the data [1], then searches over the space of networks to find one which achieves the optimal score. This approach has been implemented by Cussens in GOBNILP. On the other hand, the constraint-based approach works by performing hypothesis tests to identify edges within the graph which should not belong there. The aim of this project is to combine both score-based approach and constraint-based approach to obtain a hybrid approach which could lead to the reduction in runtime, and the increase in accuracy for GOBNILP.

[1]
[2]

2 Introduction

The task of learning the most optimal DAG through data is known as Bayesian network structure learning. We learn the network structure through data such as the one shown in figure 1.2.

The score-based algorithm approaches this by searching for network structure hypotheses with high relative posterior probability [2]. Typically, the criterion for selecting the best DAG uses penalized likelihood score such as BIC, BDEU, etc [3]. On the other hand, a constraint-based approach algorithm

approaches this by identifying constraints (GOBNILPs) in the graph then removing every edge.

3 Background

3.1 Why Bayesian networks?

Prior to the introduction on Bayesian networks, many textbooks on 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 its marginal probabilities [3]. This is where Bayesian networks come in as they aim to efficiently capture the relationships between variables in a joint probability distribution [2].

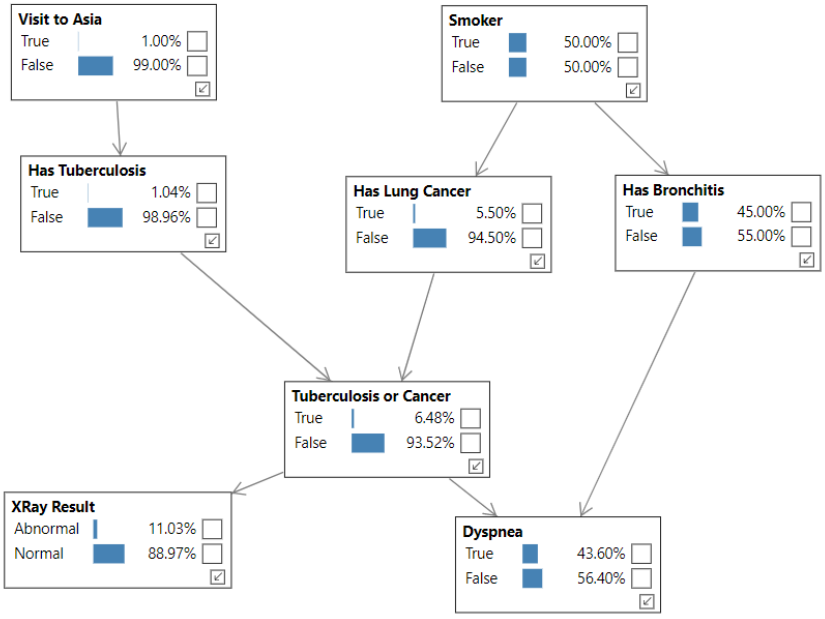
3.2 Structure of a Bayesian network

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, Suppose we are given 2 joint distributions of 2 random variables X and Y, we say that the conditional probability distribution of Y given X is the probability distribution of Y when X is observed to be some value.*

Suppose we are given a joint probability distribution $P(x_1, x_2, ..., x_n)$ with variables $(x_1, x_2, ..., x_n)$. By Pearl [3], "the structure of the Bayesian network is that suppose we start 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." 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** [1] 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* $P(x_0, x_1, ..., x_{n-1}) = P(x_{n-1} \mid x_{n-2}, ..., x_1, x_0)\ P(x_{n-2} \mid x_{n-3}, ..., x_1, x_0)\ ...\ P(x_1 \mid x_0)\ P(x_0)$ *[3].*



(a) Asia Network

Figure 1: The Asia Bayesian Network

The distribution corresponding to the DAG in Figure 1 can be written by observing its causalities such that:

$P(x_i \mid \Pi_{x_i})$ = P(Visit to Asia) P(Smoker)

$P(HasTuberculosis \mid VisittoAsia)$

$P(HasLungCancer \mid /IsaSmoker)$

$P(HasBronchitis \mid IsaSmoker)$

$P(HasTuberculosisorCancer \mid HasLungCancer, HasTuberculosis)$

$P(Dyspnea \mid HasTuberculosisorCancer, HasBronchitis)$

$P(XRayResult \mid HasTuberculosisorCancer)$

However, reality tends to be disappointing and that we often do not have the factorized joint probability distribution for the network we are working with. Additionally, we would have to identify structural properties such as the linking of the nodes, the parent sets of each nodes, the direction of the edges, etc., all by

ourselves, which leads us to the task of learning Bayesian networks structures from data.

4 Related Works

The learning of Bayesian networks from large amount of data is shown by Chickering to be an NP-hard problem, even if we limit the parent limit of the Bayesian network to be equal to 3 [1]. As the nature of the problem is NP-hard, it is not exactly possible to perform some exhaustive search over the space of available DAGs and obtain the best DAG given the data, unless the network is extremely small. 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.

4.1 The 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.

4.1.1 The Scoring Functions

When it comes to scoring functions, there are 2 categories of scoring function, The Bayesian scoring family or the information theory scoring family. The Bayesian scoring function takes the approach of computing the posterior probability distribution probability, starting from a prior probability distribution on the possible networks, conditioned to the data D, which is $P(B \mid D)$, the best network is the one which maximizes the posterior probability []. The Bayesian scoring family consists score functions such as K2, Bayesian Dirichlet (BD) and its variants Bayesian Dirichlet estimation (BDE) and Bayesian Dirichlet estimated uniform (BDeu).

On the other hand, the information theory approach
(http://www.lx.it.pt/ asmc/pub/talks/09-TA/ta$_p res.pdf$)

4.2 The Constraint-Based Approach

Based on the definition of the **d-separation** criterion [3], we approach this by performing some conditional independence (CI) tests to identify CI between two random variables X and Y given a third set of random variables we Z, $(X \perp Y|Z)$ where $X, Y \notin Z$ for every CI test within the data. Upon performing the CI tests, we would obtain a binary output, which is one of the following; "An edge could exist between X and Y given Z" or "An edge does not exist between X and Y given Z". If we obtain the former answer, we do not remove the edge. However if we obtain the latter answer, we rule out the edge and

subsequently any DAGs within the network which contains an edge between X and Y. We would then construct a Directed Acyclic Graph (DAG) by drawing an arrow from one variable to another if and only if the former variable is a direct cause of the latter variable, and there is no other unobserved variable that directly influences both variables.

Definition 3 *(**d-separation**) By Pearl [3], we say that 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 their Inductive Causation (IC) algorithm [11]. The IC algorithm first assumes three fundamental properties to infer causal relationships between variables.

1. **The Causal Markov Condition** (sometimes called the Local Markov Property) [VP90] states that if we are given a DAG D and a probability distribution P, D is a Bayesian network of P if and only if each variable **X** is conditionally independent of all its non-descendents, given its parents Π_X, and no proper subset of Π_X fulfils this condition.

The idea behind the Causal Markov Condition is such that if we know the values of the parents of a variable, X, then we can determine the probability distribution of that variable X without considering any other variables in the network, allowing us to efficiently perform inference in a Bayesian network by reducing the number of probability distribution tables which needs to be considered. [wiki]

2. **Faithfulness** [Clark and Glymour] Sometimes known as the Stability but more frequently known as the Faithfulness Condition. If G is a causal graph and P being the probability distribution generated by G, the Faithfulness Condition is satisfied if and only if every conditional independence relation is true in P are entailed by the Markov condition applied to G.

In other words, a Bayesian network should only contain an edge between variables which have direct impacts in the causal relationships. This criterion also helps us prevent over-complicating the DAG.

3. **Causal Sufficiency** A set V of variables is causally sufficient for a population if and only if in the population every common cause of any two or more variables in V is in V, or has the same value for all units in the population. [PC book]

Based on the pricinple of Occam's Razor, which states that the simplest explaination that fits the data is usually the best. In terms of a Bayesian network, it just means that the network should only include variables which are necessary to avoid over-complicating things.

The algorithm itself then takes a fairly straightforward approach towards discovering causal relationships. The first phase starts by obtaining the skeletal

structure of the graph through performing CI tests. With the skeleton of the graph, it identifies v-structures within the graph and lastly prevents new ones or cycles from being formed.

Skeletal Identification of the IC algorithm.

1. Given a dataset, find the combinations of every possible pair of random variables A, B. Then find a set of variables S_{ab} which d-separates A and B ($A \perp B \mid S_{ab}$). If and only if every possible sets of combinations for all S_{ab} have been tested, we add an edge between every pair of A and B, $(A - B)$.

Once the skeletal structure of the graph has been obtained, we move on to the second phase, where we identify the **v-structures**.

Definition 4 *(**v-structure** A v-structure is a structure within the graph which takes the following form $A- > B < -C$)*

2. Find every triple of vertices in the graph with variables A, B, C such that $(A - C - B)$, check if $C \in S_{ab}$. If so, move onto the next step. Else orient the edges such that $(A- > C < -B)$

As we have identified in our CI test that C is indeed an element in the set S_{ab} which d-separates A and B, implying A and B are independent when C is observed ($A \perp B \mid C$). Once every v-structure in the graph has been identified, we move onto the final step, that is to prevent cycles being formed or more v-structures from being created.

3. In the partially directed graph, orient as many edges as possible while making sure no new v-structures are created and no directed cycles are formed.

Although this algorithm allows us to discover the structure of the DAG, this algorithm cannot be in practice especially on larger-sized networks as the number of CI tests grows exponentially bigger with its number of vertices. P. Spirtes and C. Glymour provided an improvised version of the IC algorithm which alleviated this issue, the PC algorithm [15].

While the latter stage of the PC algorithm remains the same as that of the IC algorithm, the main motivation of the PC algorithm is to reduce many CI tests as possible by performing the CI tests hierarchically. The skeletal-learning process of the PC algorithm starts by searching through the entire set of variables which d-separates variables X and Y, the PC algorithm first starts by creating a complete undirected graph from the set of variables V. it then starts by identifying CI by searching for sets of variables of the **adjacents** of X, Y with sets of increasing cardinality $0, 1, ...$, starting from 0. Removing an edge as soon as some conditional independence is discovered.

However, just like the IC algorithm, the complexity of the PC algorithm is still bound to its skeletal-learning phase. The worst-case scenario of the algorithm happens when no CI relationships exist between the variables in the data.

In such a case, the algorithm has to test all possible CI relationships between pairs of variables, leading to exponential runtime scaling with the number of variables. This makes it inefficient to run the algorithm on larger networks. However, the PC algorithm is still considered to be a computationally efficient method for learning Bayesian networks.

4.3 Hybrid Approaches

Just like its name suggests, 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 [Laura E. Brown]. 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 H2PC algorithm uses a different approach to skeletal learning by using the PC algorithm, while also using the Bayesian-scoring hill climbing search algorithm for its score-based approach.

5 Methods

5.1 Hypothesis Test

Constraint-based algorithms performs hypothesis tests to identify CI between variables, allowing us to learn the skeleton of the graph. Given 2 variables, X and Y, and a set of variables which might d-separate X and Y, called Z. We first start by stating our null hypothesis (h_0) and alternative hypothesis (h_1)

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 our hypothesis test rejected h_0, we do not completely reject h_0 but simply state : 'Given the observed data, we do not accept the null hypothesis'. We then proceed to perform hypothesis test on higher **orders** of CI.

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

5.2 The chi-squared χ^2 test

When it comes to hypothesis testing, commonly used hypothesis tests includes Pearson's chi-squared χ^2 test [20], 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:

Table 1.1

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

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

1. We start by defining h_0 and h_1,

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

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

Once defined our hypothesis, we compute the contingency table of our variables.

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

To create this table, we 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, we compute the expected frequency, f_i, by doing $f_{three=1} = \frac{6 \cdot 10}{20} = 3$. Do this for each cell and we get the expected frequency table:

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

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

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 we do that for every variable, we get:

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

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

Lastly, we determine the degrees of freedom (dof). 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 we perform this test at a 5% significance level, we can go ahead and 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 our χ^2 test $< \chi^2_{dof=1,SL=0.05}$, we do not reject the null hypothesis.

5.3 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 us to accept less null hypothesis, which consequently removes less edges from the graph, and vice-versa, 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 we increase the value of α, 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 our

test, we keep it 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.

While $\alpha = 0.05$ is a decent threshold for most cases, on some networks, this value might not be a good fit and we will investigate the score of the Bayesian network given different significant thresholds.

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

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

5.4 The PC algorithm

We first identify the CIs of our graph by using the skeletal-learning phase of the PC algorithm, denoted by PC_{SL}. This algorithm combined with the score based algorithm, **GOBNILP**, will form our hybrid approach.

Algorithm 1 PC_{SL}

Require: A dataset containing a set of variables V
Start by creating a complete undirected graph with an edge between every $\binom{n}{2}$ pair of variables.
We denote Adj_{XY} as the set which contains the adjacent variables for variables (X,Y), and Adj_{max} as the maximum cardinality of Adj_{XY}.
$N \leftarrow \mid Adj_{max} \mid$
$n \leftarrow 0$
while $n \leq N$ **do**
  if $\mid Adj_{XY} \mid \geq n$ **then**
    Test for CI for $(X \perp Y \mid ADJ_{XY})$

Input : A dataset with a set of vertices V

1. We start by forming an edge between every pairs of variables in the set V to produce a complete undirected graph, G.

   n = 0

   repeat

2. Create a set for the pairs of variables X and Y, S_{xy}, such that X and Y are adjacent in G. We denote the set of variables adjacent for each pair of (X,Y) as Adj_{xy}. If the cardinality of Adj_{xy} is greater than or equal to n, $\mid Adj_{xy} \mid \geq n$, we proceed to test for CI.

3. The moment we find a set of Adj_{xy} that d-separates variables X and Y, we remove the edge which connects X and Y, and subsequently remove the pair of variables (X,Y) from S_{xy}

n = n + 1

To illustrate this, suppose we input a dataset with 8 vertices into the algorithm, V = {One, Two, ... , Eight}. We start with the following complete undirected graph.

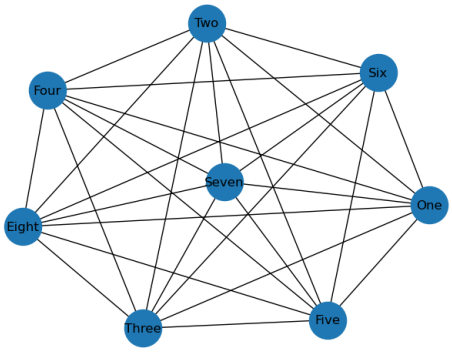


Figure 2: Complete Undirected Graph

We identify 0^{th} order CI and obtain the following indepencies:

- 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, we proceed to d-separate those pairs of variables, resulting in the graph:

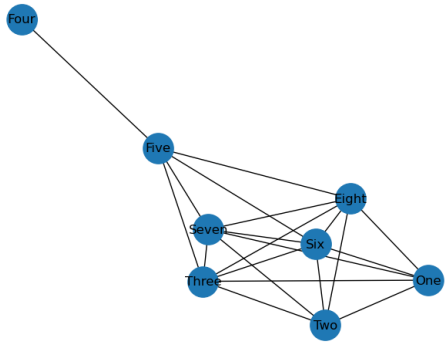


Figure 3: Graph After Removing 0^{th} Order CI

Now test for 1^{st} order , we get the following sets of indepencies:

- 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

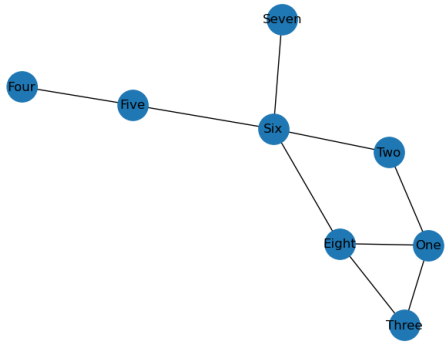


Figure 4: Graph After Removing 1st Order CI

This time we get 11 independent relationships, which totals to 19. We keep going for $n = \{2, 3, ..., S_{max}\}$, We obtain the following graph:

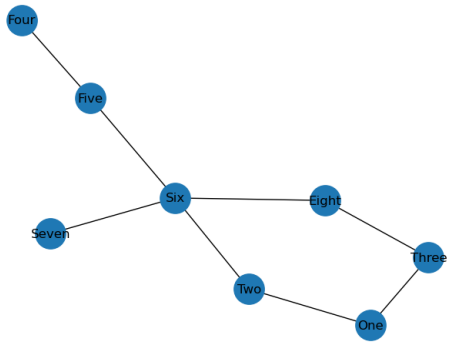


Figure 5: Final Undirected Graph

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 efficient to perform higher orders of CI testing on even medium-sized networks $\mid V \mid \geq 20$, this will be discussed further in section 6.

5.5 Networks Used

In this project, we evaluated the performance of the hybrid algorithm (PC_{SL} + GOBNILP) with networks of different sizes. The data we used are synthetic data obtained from [22] produced from some sampling method such as **forward sampling**. 12 networks were used to evaluate the performance of our algorithm, the networks evaluated consists of small-sized networks such as the Asia network

which was used for clinical diagnosis [23] | V_{asia} | = 8. Medium-sized networks such as the Alarm network which was used to implement an alarm message system for patient monitoring | V_{Alarm} | = 37 [24], large-sized networks such as the HailFinder network which was used to forecast severe weather conditions [26] | $V_{HailFinder}$ |= 56,etc.

To sum it up, the table below shows the networks we will evaluate our algorithm on

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 1: Summary of Networks Used

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 its probability.*

5.6 GOBNILP

Globally Optimal Bayesian Network learning using Integer Linear Programming (GOBNILP) is a program developed by Cussens and Bartlett [9] which acts as a score-based approach to learning the network structure. The way it encodes Bayesian networks as an 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 parents set is 0, then $W = \emptyset$.

GOBNILP measures how good the BN is by applying the one of the score metrics under the BD score family, the BDeu scoring metric. The BDeu scoring metric uses a **decomposable** score and is formally defined with the equation below []:

Definition 9 ***Decomposable*** *[27] We say that the measure on a structure if decomposable if it can be written as a product of measures, each of which only involves a function of one vertex and its parents. This allows for more efficient computing*

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

Where

- 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

After applying the PC_{SL} constraint-based algorithm, we can then obtain a list of tuple pairs of variables (X,Y) that satisfy the condition where the null hypothesis is accepted by the χ^2 test and $X \perp Y$. In this case, we tell GOBNILP to restrict an edge between X and Y, before running the algorithm.

5.7 Evaluating Structural Accuracy

While the aim of this project is to reduce the runtime of GOBNILP as much as possible, we also have to ensure structural accuracy of the network. Otherwise, we could just remove every edge in the graph to produce an empty graph which in a way fulfils the task with maximum efficiency.

The F-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 [28]. In this case, our 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 our 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 our baseline graph which should be there, but is not present in the learned graph will be denoted as FN. Formally, the F-score is defined to be:

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

6 Results

6.1 Runtime of GOBNILP

Before proceeding with our experiments, it is important to take into account a few key considerations, such as the overall runtime of our algorithm. We define the overall runtime T_{hybrid} as the sum for the time taken to restrict edges ($T_{restrict}$) and the time taken to find a good scoring network T_{score}.

$$T_{hybrid} = T_{restrict} + T_{score}$$

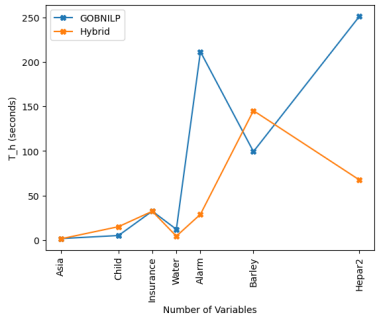
It is important to find a balance between these two factors, cause if the restriction phase (PC_{SL}) of the algorithm takes such a long time to run that GOBNILP is better off without it, then it defeats the purpose of using a hybrid algorithm in the first place. Additionally, I will denote the time taken to produce a graph by GOBNILP as T_{score}, the graph produced by GOBNILP without adding constraint as G_{score}, and the graph produced by the hybrid algorithm (the combination of GOBNILP and PC_{SL}) as G_{hybrid}.

In the table below, I compare the differences in time and accuracy between GOBNILP and the hybrid algorithm. These tests were carried out at the 1st order CI, a 5% significance threshold, and a parent size limit of 2. The F-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 shows a reduction, if positive shows an increase in time.

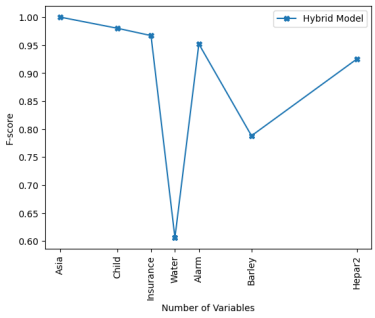
Network	T_{score}	T_{hybrid}	Reduction	F-Score
Asia	1.79	1.45	+19.00	1.000
Child	5.21	15.14	-190.60	0.980
Insurance	32.55	32.39	+0.49	0.967
Water	12.17	4.47	+63.27	0.606
Alarm	211.31	29.01	+86.27	0.952
Barley	99.35	145.23	-46.18	0.860
HailFinder	197.91	156.48	+20.93	0.788
Hepar2	251.09	67.50	+73.20	0.925
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 2: This Table Shows the Time Taken Measured in Seconds to Learn the Graph by Gobnilp and the Hybrid Algorithm, Along With the F-Score Which Compares G_{score} and G_{hybrid}

If a network requires an unfeasible amount of time, it implies that GOBNILP was unable to render a graph within the given 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.



(a) Comparing Runtime of GOBNILP and the Hybrid Algorithm



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

Figure 6: Runtime Comparisons and Model Accuracy

Our 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.

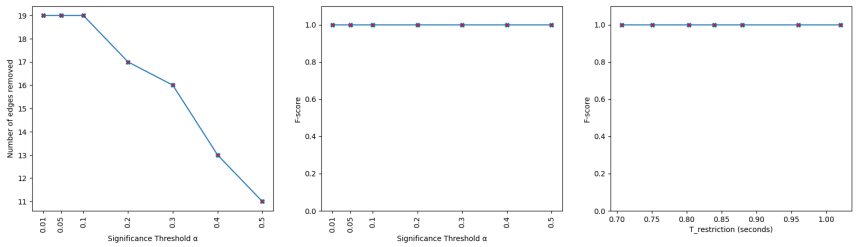
Furthermore, our analysis shows that other factors such as the significance threshold and the order of CI used for the hybrid algorithm has a notable impact on the balance between runtime and accuracy. With those analysis, we will hopefully address the elephant in the room which are the outliers in the data such as the drastic decrease in the F-score value for the Water network.

6.2 The Effects of the Significance Threshold

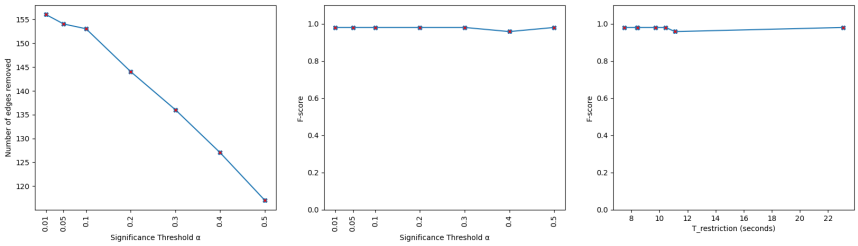
To investigate the impact of the significance threshold on our model, we conducted an analysis to investigate the trade-off between runtime and accuracy. We varied the threshold used for the hybrid algorithm and observed its effect on these two metrics. Additionally, we also measured the number of edges removed from the graph given the significance threshold.

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

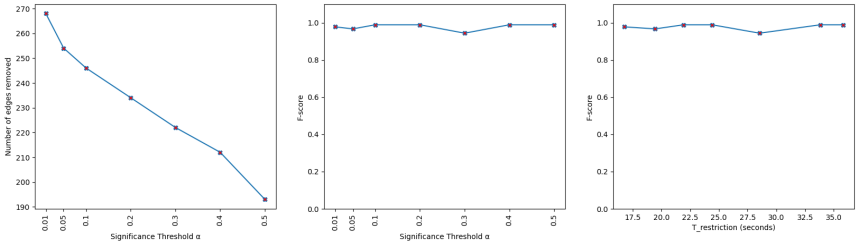
To visualize the relationship between the significance threshold, runtime, and accuracy, I plotted some figures for significance threshold values of:



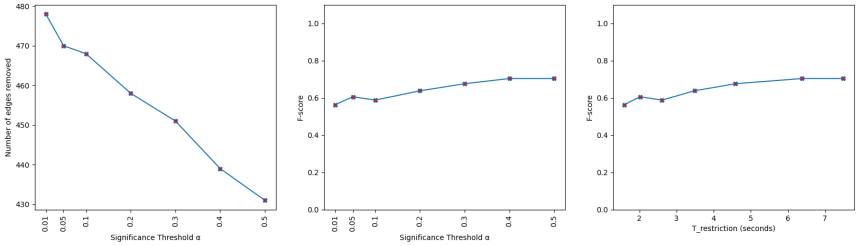
(a) Asia Network



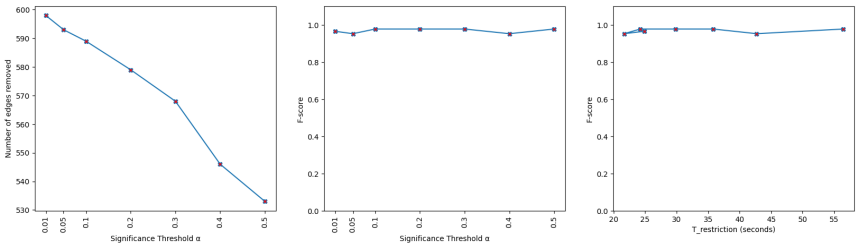
(b) Child Network



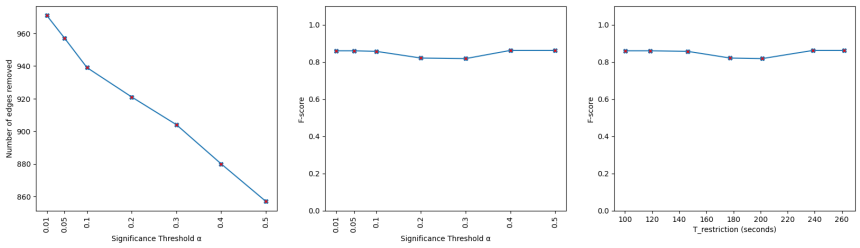
(c) Insurance Network



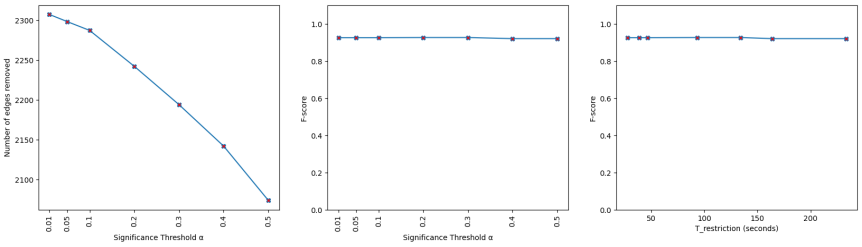
(d) Water Network



(e) Alarm Network



(f) Barley Network



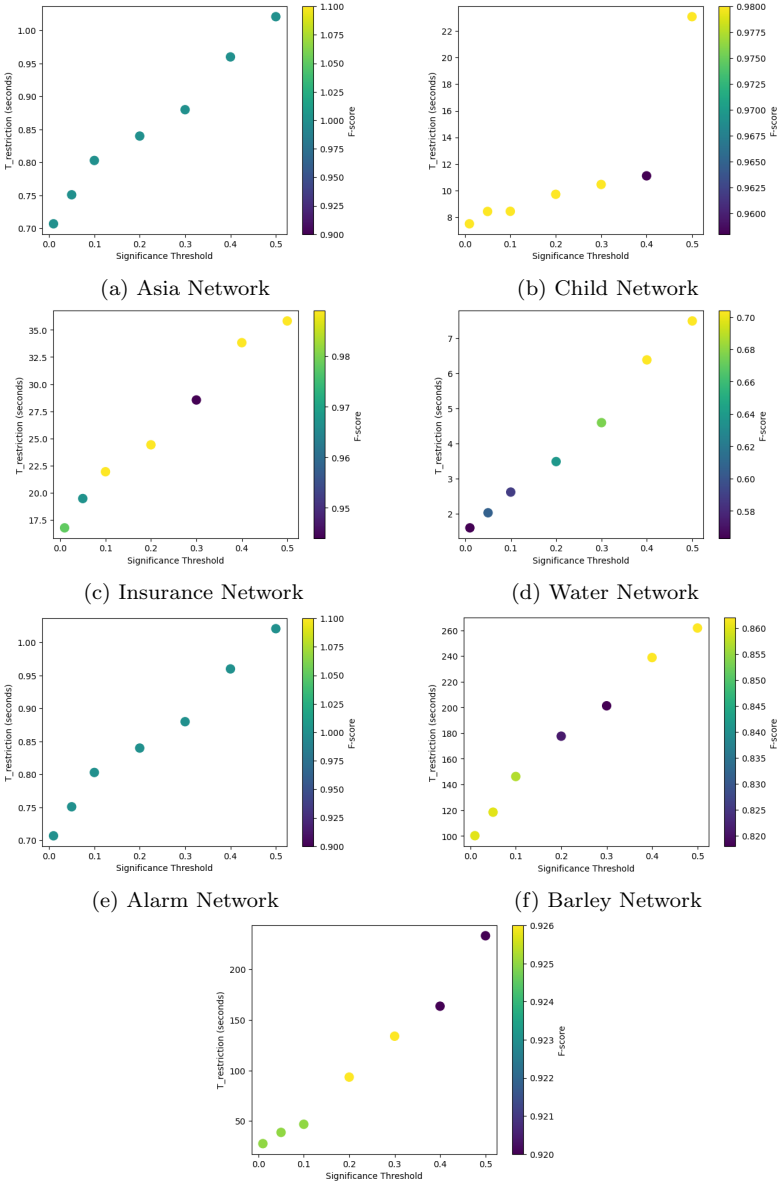
(g) Hepar2 Network

Figure 7: Runtime Comparisons and Model Accuracy

In eacn subfigure, there are 3 plots, each of which respresents some 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 F-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 our results. On cases with smaller-sized networks such as the Asia network, there does not seem to be any form of impact on its 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. As we examine 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 F-score and runtime of the algorithm can vary across different networks, the Water network stands out as an outlier with a significantly lower F-score compared to other networks, even at higher significance thresholds, and a relatively short runtime compared to some other networks. This could be due to the complexity of the network, which may make it more difficult to accurately identify edges which should belong, leading to lower F-scores. This indicates that the hybrid algorithm used to learn the network structure is not suitable for the Water network. Whereas for most of the networks, there is consistency between the F-score and the runtime in the sense that the deviation of the F-score

with increasing runtime is minor.

The question that arises is how we can optimize the performance of the algorithm by achieving maximum F-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 8: Scatter Plots Comparing Runtime and Significance Threshold

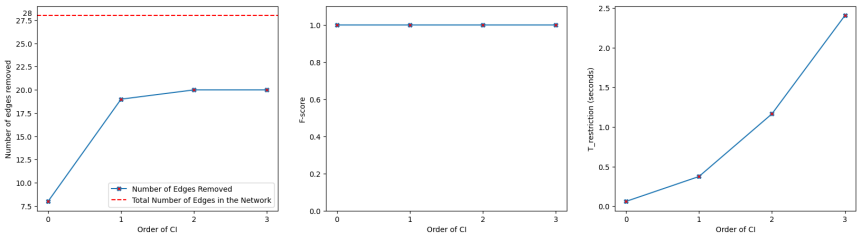
In these plots, the colours represents the value of the F-score, the y-axis represents the runtime and the x-axis represents the significance threshold. For example, in the case of the Alarm network, the optimal significance threshold value lies between 0.1 and 0.2. By analyzing the scatter plots for each of these networks, we can gain better insights on achieving optimal trade-offs between runtime and F-score, given the significance threshold.

6.3 The Effects of the Order of CI

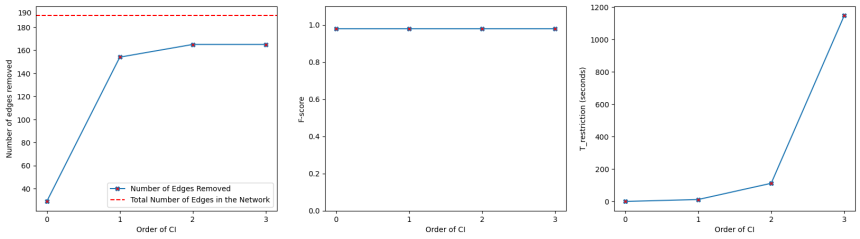
Suppose we wanted to find out if X was independent of Y given a set of variables Z,

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

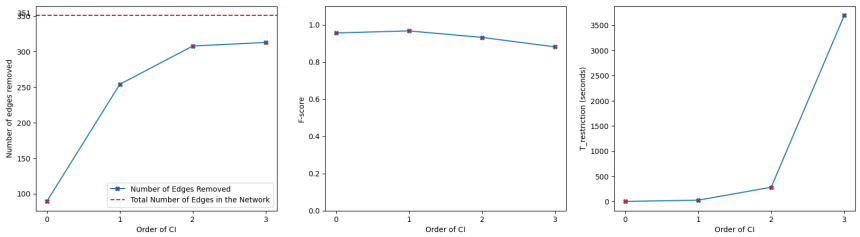
we say that the order of CI, n is equal to the cardinality of Z $n =\mid Z \mid$. To put it simply, the order of CI is the order in which the PC_{SL} algorithm identifies d-separation between two variables, starting from 0 and increasing up to the maximum cardinality of Z, denoted as Z_{max}, such that $Z = \{0, 1, ..., Z_{max}\}$. Similar to our hypothesis test section, we plotted some basic plots for how the order of CI plays a role in the algorithm.



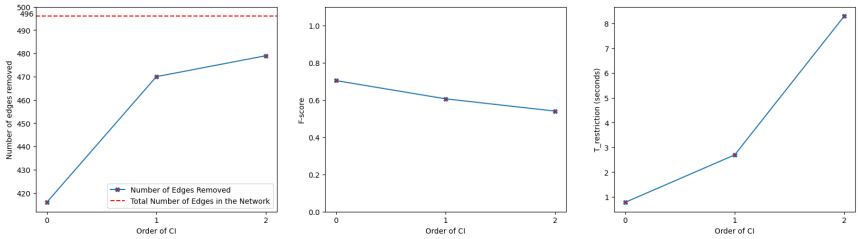
(a) Asia Network



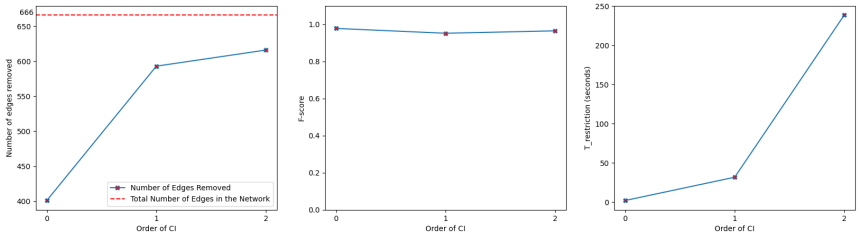
(b) Child Network



(c) Insurance Network



(d) Water Network



(e) Alarm Network

Figure 9: The Effects of the Order of CI on Runtime and Accuracy

Figure 9 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 F-score and the third showing the effects of the CI order on the runtime of the **restriction** phase of our 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, therefore those networks have been excluded from our 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 F-score does not vary significantly except for the Water network.

While it is clear that orders of CI higher than 1 is almost not worth considering for most networks, the following question remains, is it better to use 0^{th} order CI or 1^{st} order CI. To answer this question, we produced a simple plot comparing the **overall** runtime on our networks.

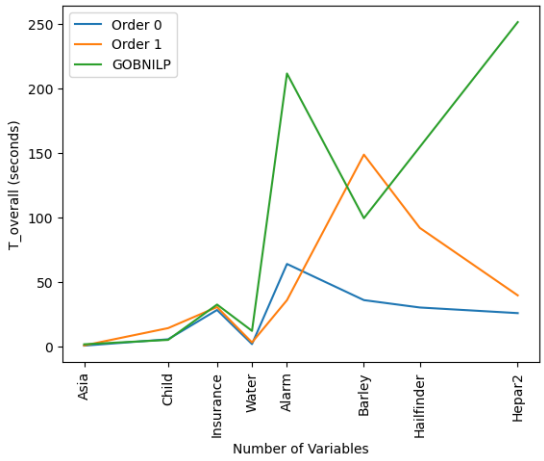


Figure 10: Comparing Different Orders

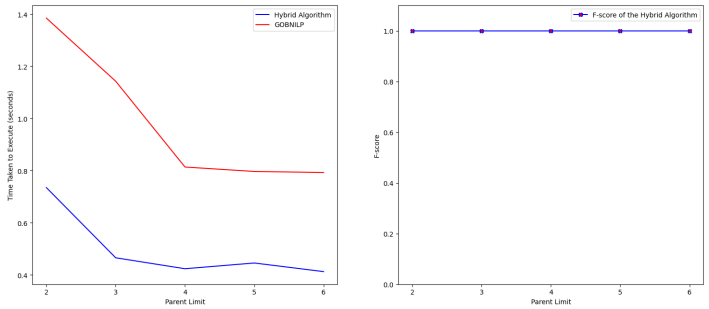
Surprisingly, the plot shows that running the algorithm at the 0^{th} order CI produces the fastest overall runtime. To summarise this section, while the PC_{SL} algorithm allows us to search for higher orders of CIs in a 'reasonable' amount of time, it may not always be necessary to do so. Depending 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.

6.4 The Hybrid Algorithm on Higher Parent Limit

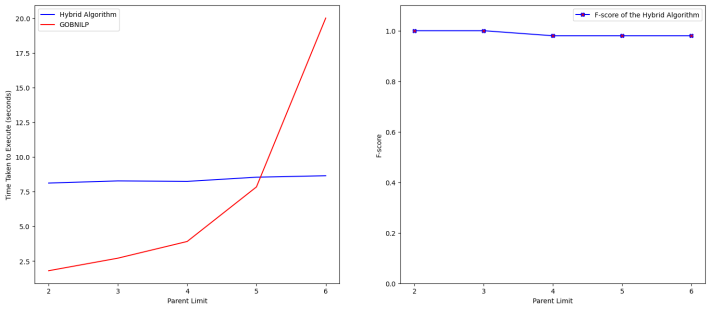
Lastly, we evaluate the performance of the hybrid algorithm on Bayesian networks with higher parent limit. Previously, all of our experiments were performed on a parent limit on size 2. This was neccessary because if I were to increase the parent limit for GOBNILP, it would increase the search space of available DAGs massively. However in this section, I compare the performance of the hybrid algorithm to GOBNILP on different parent limits.

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

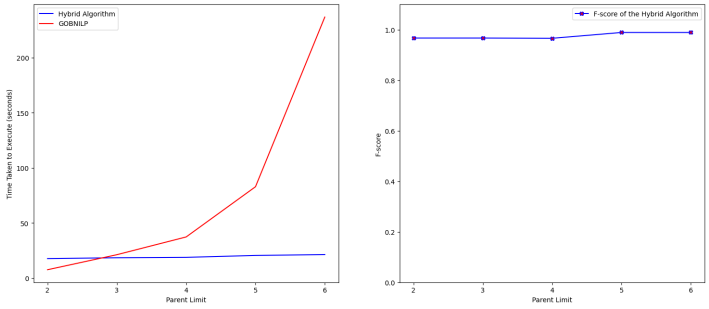
Once again, we plot some figures which compares the performance of the hybrid algorithm to GOBNILP for higher parent limits. We performed the experiment on 1^{st} order CI and set the value of α to 0.05.



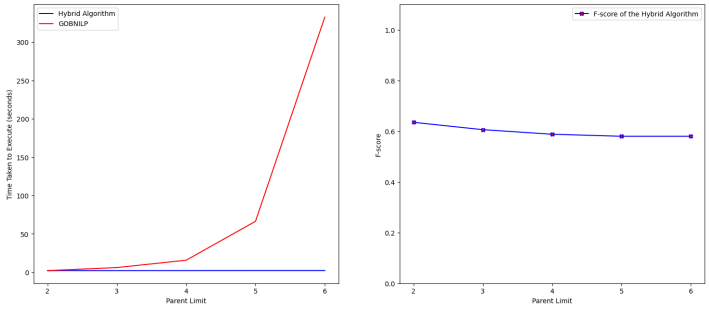
(a) Asia Network



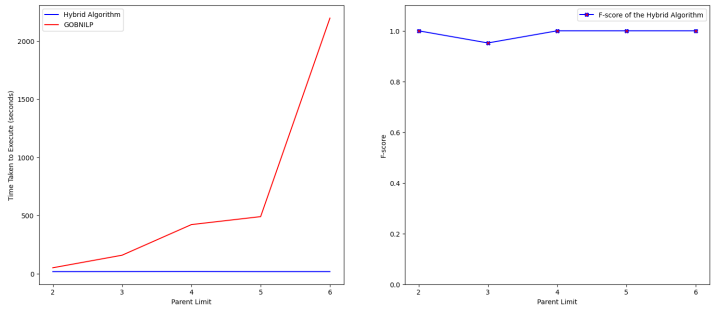
(b) Child Network



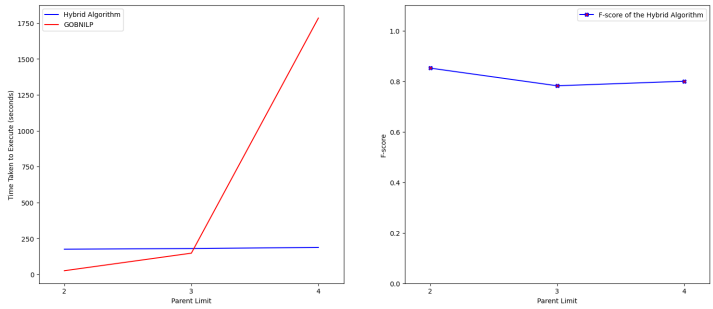
(c) Insurance Network



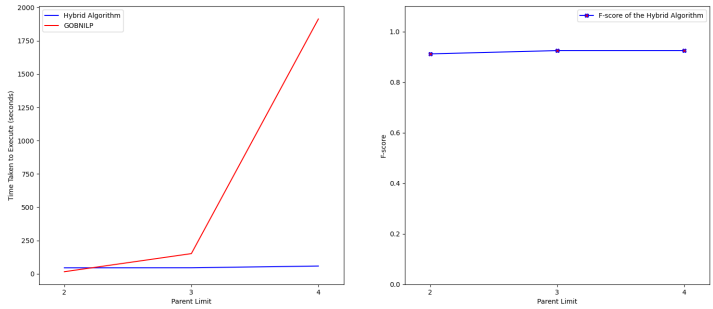
(d) Water Network



(e) Alarm Network



(f) Barley Network



(g) Hepar2 Network

Figure 11: The Difference in Runtime Between T_{hybrid} And T_{score}, as Well as the Accuracy of G_{hybrid} Compared to G_{score}.

Figure 11 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 F-score of G_{hybrid} compared to G_{score} for the same parent limits. As the parent limit increases, GOBNILP shows an exponential trend in its runtime, while the runtime of the hybrid algorithm remains very 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 with minor penalties on higher parent limit, the consistent performance of the hybrid algorithm in terms of runtime

and F-score suggests that it may be a more reliable choice than GOBNILP, particularly when dealing with larger datasets and higher parent limits.

7 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 challenge when it comes to combinatorial optimization problems, 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 F-score as a scoring metric is insufficient to evaluate the accuracy, because the F-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.

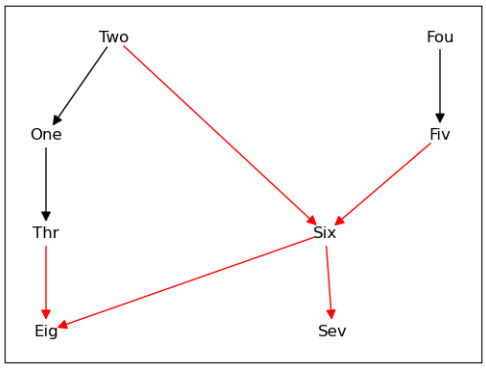


Figure 12: GOBNILP on the Asia Network

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. Regrettably, a feasible solution to this issue cannot be currently determined. Furthermore, due to the utilization of only one comparison metric, a thorough examination of the results may be necessary to ensure G_{hybrid} performs well enough.

Aside from the limitations of the algorithm,

8 Conclusion

To summarise this

References

[1] 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.

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

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