Spring 2018

# Hierarchical clustering based structural learning of Bayesian networks

Nikhita Sharma
*Iowa State University*, nsharma@iastate.edu

**Hierarchical Clustering based Structural Learning of Bayesian Networks**

by

**Nikhita Sharma**

A creative component submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Computer Science

Program of Study Committee:
Jin Tian, Major Professor
Xiaoqiu Huang
Wei Le

Iowa State University

Ames, Iowa

2018

# TABLE OF CONTENTS

Page

# LIST OF FIGURES

Page

# LIST OF TABLES

Page

## ACKNOWLEDGMENTS

I would like to thank my major professor, Dr. Jin Tian, for his guidance and patience during my research. His insights and knowledge of the subject kept me on the right track. I would also like to express gratitude towards my committee members, Dr. Xiaoqiu Huang and Dr. Wei Le for their support.

Also, I would like to take this opportunity to thank my parents, for their unconditional love and support. Without them, I would not have been able to come to Iowa State University to pursue my master's degree.

## ABSTRACT

Bayesian networks are being used in various domains, such as data mining, diagnosis, bioinformatics/computational biology, etc. One problem associated with Bayesian networks is to learn their structures from training data. In this paper, we introduce a new approach to structural learning of Bayesian networks, based on hierarchical clustering. We learn the network in hierarchical stages, learning over a subset of the random variables at each stage. Experiments show that this approach learns Bayesian networks faster as compared to curriculum-based learning methods. We show a comparison of our networks with curriculum based learned Bayesian networks over different evaluation metrics as well. Also, performance of hierarchical clustering vs an existing ordering-based algorithm is observed.

# CHAPTER 1. INTRODUCTION

Bayesian networks are a class of probabilistic graphical models that provide a compact representation of a joint distribution. They support algorithms that answer probabilistic queries and are used for causal modeling. A Bayesian network consists of two parts: Directed Acyclic Graph (DAG) and Conditional Probability Distribution (CPD). We learn a Bayesian network by first learning the DAG structure and then estimating the parameters of the conditional distributions. The latter is a well-studied problem in statistics. However, the former is an active research area. Learning the DAG structure from training data, i.e. structure learning, is the problem at hand. Our aim is to find a Bayesian network that best conforms with the training data.

A major idea that inspired this work is Curriculum Learning. Curriculum based learning methods [1] basically learn a network in stages. At each stage, a subnet is learned over a subset of random variables, conditioning on the rest. Eventually, all the variables are learned as the subset grows. This is similar to the way humans learn. For instance, if we are learning a new language, instead of looking at complete sentences, one will start by learning certain common phrases. Slowly, one will progress to understand complete sentences. Humans start with easier and more common examples, and then more on to the complex and rare ones. The basic idea is to decompose the target structure into multiple components and learn one component at each stage. Hierarchical Clustering involves assigning each random variable its own cluster. We then compute the similarity between each of the cluster and join the most similar clusters. This process is repeated till only a single cluster is left.

Initially, we assign each random variable its own cluster. Next, we use mutual information to compare which clusters are the most connected. Merge the highly connected clusters. Once we get a cluster of a reasonable size, we start learning the network, starting with this biggest cluster. Multiple clusters could be learnt in parallel. This process is continued till all the variables are in a single cluster, thereby learning the entire network. Next, we save and evaluate the learnt network.

In further sections, we start with some background knowledge about Bayesian networks, the structure learning problem and hierarchical clustering in chapter 2, including the problem definition. Some related work is explored in Chapter 3. In chapter 4, we look at the algorithmic details of our approach, followed by the supporting experiments and evaluation in chapter 5. In the end, we conclude in chapter 6.

# CHAPTER 2. BACKGROUND

This section covers the necessary background on Bayesian networks and the structure learning problem. A brief introduction to curriculum learning and hierarchical clustering is also given.

## 2.1 Bayesian Networks

A Bayesian network is essentially a Directed Acyclic Graph(DAG), G and a joint probability distribution, P over a vector of random variables X= ($X_1$,…, $X_n$). It can be represented as B= (G, P) where each node of the graph represents a variable in G. The DAG can be represented as a vector G = ($Pa_1$…, $Pa_n$) where each $Pa_i$ is a subset of $X/X_i$ and specifies the parents of $X_i$ in the graph. Each node and its parents in the DAG is associated with a conditional probability distribution (CPD) P ($X_i$ | $Pa_i$). Then the joint distribution P(X) must be factorized as follows:

$$P(X) = \prod_{i=1}^{n} P(X_i | Pa_i)$$

Bayesian networks encode conditional independence relations via a DAG G over all the variables X. Fig. 1 shows an example of a Bayesian network, with four random variables, Toothache, Catch, Cavity and Weather. It can be observed that

P (Toothache, Catch, Cavity, Weather) =P (Toothache, Catch, Cavity) P(Weather)
Also, Catch is conditionally independent of Toothache, given Cavity:

P (Catch | Cavity, Toothache) = P (Catch | Cavity)

This makes sense to humans intuitively, since there is no relation between how the weather is and whether you will get a cavity or not.

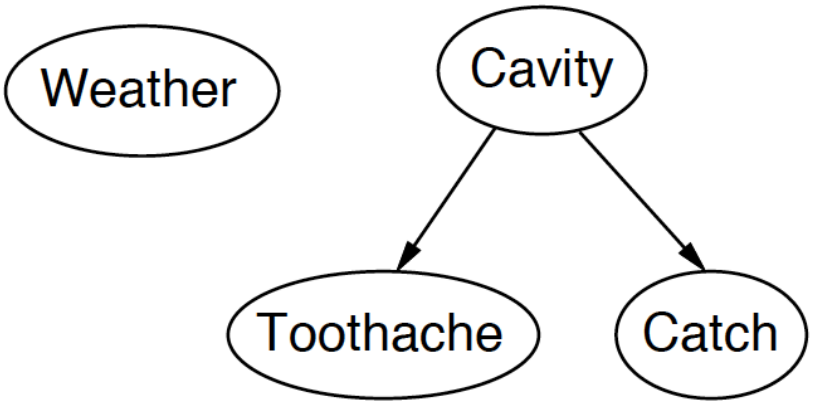


Figure 1 An example of a Bayesian network

### 2.2 Structure Learning of Bayesian Networks

Learning the DAG structure of a Bayesian network from training data is a challenging task. Assuming that the data D are generated independently and identically from an underlying distribution P(X) which is induced by some Bayesian network B*(G*; P*). The goal is to find a perfect map G* for P*. This is called the structure learning problem of Bayesian networks.

There are two main approaches for solving this problem: constraint-based [2] and score based. Constraint based algorithms estimate from the data whether certain conditional independencies (CIs) between the variables hold. They perform statistical or information theoretic measures to directly identify conditional independencies (CIs) between variables and build a DAG structure consistent with the CIs. The CI constraints are propagated throughout the graph and the DAGs that are inconsistent with them are eliminated. There is a disadvantage to this approach, since there is a probability of making errors, given limited data samples. An example is the Peter–Clark algorithm.

Score based algorithms convert the learning problem to an optimization problem. The idea is to optimize a scoring function that estimates how well the DAG fits the data. The algorithm is score based [3]. The score used is the BDeu score, which is defined in further sections. However, finding a best Bayesian network is NP-hard when using the BDeu scoring criterion [4].

### 2.3 Hierarchical Clustering

Hierarchical clustering is widely used in data analysis. The key idea is to build a binary tree of the data that successively merges similar groups of points. In our case, the points are going to be clusters containing random variables. The similarity measure for these clusters will be mutual information, described in chapter 3. Each random variable X is placed into its own singleton cluster initially. Then, iteratively two closest clusters are merged. This continues till all the random variables are merged into a single cluster.

# CHAPTER 3. RELATED WORK

This work is primarily inspired by [3], that presents an algorithm to learn Bayesian networks using an incremental construction curriculum. This approach is score based. Curriculum based learning methods [1] basically learn a network in stages. For each stage, they learn the network only over a subnet of random variables, while conditioning on the rest of the variables. This process continues till all the variables are learned as the size of the subset increases. This is similar to the way humans learn. For instance, if we are learning a new language, instead of looking at complete sentences, one will start by learning certain common phrases. Slowly, one will progress to understand complete sentences. Humans start with easier and more common examples, and then more on to the complex and rare ones. Incremental curriculum learning construction decomposes the target structure into multiple components and learns one component at each curriculum stage.

A curriculum is a sequence of weighting schemes of the training data (W1, W2, …,Wm). The first scheme W1 assigns more weight to simpler samples, gradually increasing the difficulty, till Wm assigns equal weightage to all samples. Learning is iterative, each time from the training data weighted by the current weighting scheme and initialized with the learning result from the previous iteration.

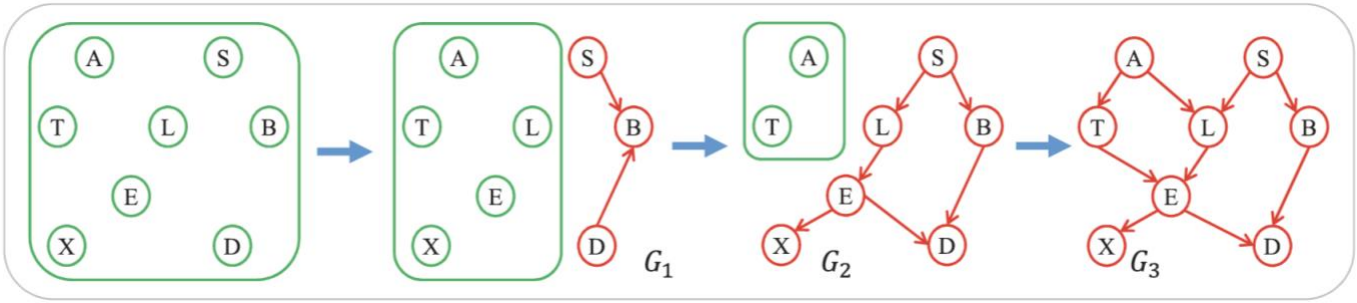


Figure 2 An illustrative example of curriculum learning of a Bayesian network structure

Figure 2 [3] illustrates how we learn a Bayesian network over the given set of nodes: {S, B, D, L, E, X, A, T}. We learn the Bayesian network structure in three stages. In the first stage, we learn a subnet G1 over {S, B, D} from scratch. Next, learn a larger subnet G2 over {S, B, D, L, E, X} with G1 as the start point of search. Finally, learn a full network with G2 as the start point. Each subnet, outside the rounded rectangle, is conditioned on the rest of the variables, inside the rounded rectangle. The stages ({S, B, D}, {S, B, D, L, E, X}, and {S, B, D, L, E, X, A, T}) are called a curriculum.

This idea of curriculum learning motivated our algorithm to learn the Bayesian network in stages, over subset of variables, rather than all the variables at once.

Another paper of relevance in Bayesian structure learning is [9], learning Bayesian networks with thousands of variables, without constraints on the in-degree. First, this algorithm finds possible parent sets of a node by searching effectively. A similar idea is used in our algorithm, to restrict the search space while learning the Bayesian network. Second, it improves an existing ordering-based algorithm for structure optimization. On very large datasets containing up to ten thousand nodes, this approach consistently out performs the state of the art.

Ordering-based search (OBS) has been proposed in [11]. Given an ordering over the variables, the optimal network with respect to that ordering can be found in time O(Ck), where $C = \sum_{i=1}^{n} |C_i|$ and k denotes the maximum in-degree. $C_i$ for each variable $X_i$ a list of candidate parent sets. Once an ordering has been sampled, the highest-scoring structure given the order is quickly obtained.

We show a comparison of the network learned by one of these algorithms from the BLIP package [8], called WinOBS, with our algorithm. WinOBS expands the idea of iterated local search, adapting it in a way that it employs the window insertion operator with increasing window size.

# CHAPTER 4. HIERARCHICAL CLUSTERING BASED STRUCTURAL LEARNING

In this chapter, the hierarchical clustering based structural learning algorithm is explained in detail with explanation on design choices. The scoring function used, Bayesian score, is also elaborated.

## 4.1 Generation of Parent-Children Sets

First, generate Parent-Children sets called PC sets for each set of training data. The MMPC algorithm [5] is run to generate the PC set $S_i$ for each $X_i$. Let S = ($S_1$, ..., $S_n$). Given a target variable T and statistical data D, MMPC returns $PC_T$, provided there is a graph that conforms to the data distribution and the statistical tests performed return reliable results. A node may be a parent of T in one network and a child of T in another. However, the set of parents and children of T, i.e., {X}, remains the same in both.

## 4.2 Compute Mutual Information

Second, the curriculum is initialized with individual clusters for every variable. Each variable in X is its own cluster. Next, based on mutual information, the most similar clusters are decided and merged. The average mutual information is calculated by dividing the pairwise mutual information (for every possible pair between any two clusters) by the product of the sizes of the clusters. The average mutual information for a pair of clusters, and mutual information for a pair of variables is given by:

$$AveMI(X,Y) = \sum_{x \in X} \sum_{y \in Y} MI(x,y) \,/\, |X||Y|$$

$$MI(x,y) = \sum_{y=b}\sum_{x=a} p(a,b)\ \log \frac{p(a,b)}{p(a)p(b)}$$

Once the clusters with best average mutual information (c1, c2) are found, they are merged into a single cluster at position of cluster c1. Cluster c2 is removed. There is a limit or threshold that decides when to learn the Bayesian network. Rather than learning small clusters, clusters are merged until a cluster of size 20 is created. After meeting this criterion, the Bayesian network learning process begins. Various values of this limit were tested [3, 5, 10, 15, 20, 25,…] and the best one was selected.

### 4.3 Hierarchical Clustering

At each learning stage, the starting point of the search is set to graphs learned over c1 and c2 say, G(c1) and G(c2), without any edge between them. Next, a score-based search is used to find a good network over variables in $c1 \cup c2$. This search can be implemented as any search algorithm that optimizes a scoring function.

An example of the learning process for a dataset Asia, which has 8 nodes and a maximum in-degree of 2, can be seen below, in Figure 3. In the beginning, all the eight random variables are individual clusters. Next, the pair of clusters with maximum mutual information is found, i.e. lung and either. These clusters are combined. The Bayesian network over these two random variables is learnt. At the next stage, the random variables with maximum mutual information is asia and tub, so, these clusters are merged. This process continues till all the variables are in a single cluster, thereby learning the entire Bayesian network. It is possible for multiple clusters to be merged at a single stage since they must have the same maximum mutual information.

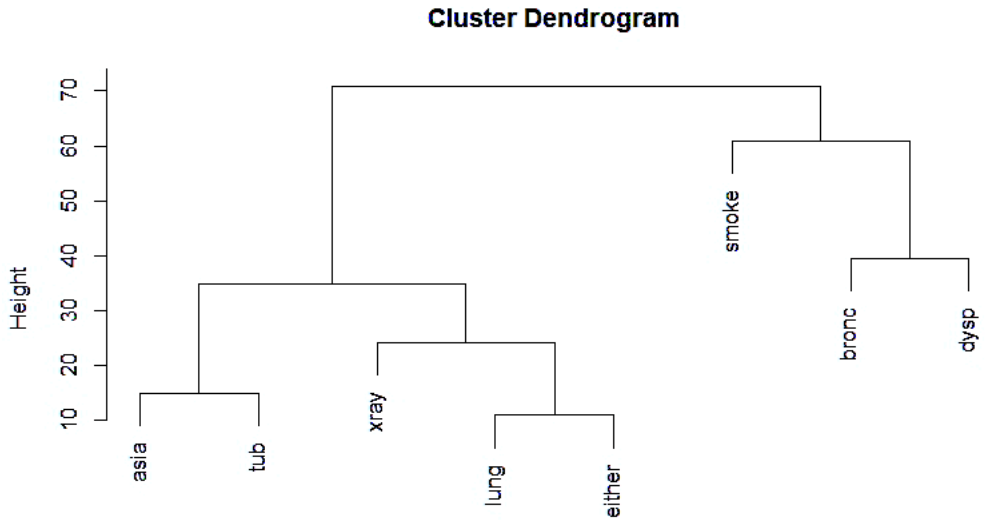


Figure 3 Learning Hierarchy for Asia1000 Dataset

### 4.4 Scoring function

The following scoring function from [3] is used in the algorithm: where SS is the sample size, $V(G_i)$ is the number of variables and $E(G_i)$ is the number of edges. The score used is the BDeu score. $D_i$ are the data segments by grouping the samples based on the values of X' (newly merged cluster to be learnt).

$$score_{BDeu}(G_i, D_i) = \log P\left(G_i, D_{i,j}\right)$$

$$score\ (G_i, D_i) = \sum_{j=1}^{q} score_{BDeu}\ (G_i, D_{i,j}) - \left(\frac{a}{SS} + \frac{V(G_i)}{b}\right) E(G_i)$$

The scoring function controls overfitting and penalizes the score function when the number of variables is too large, or the sample size is small. This scoring function makes use of all the training samples.

The algorithm uses greedy hill climbing as the search algorithm. The search space is however constrained by the PC set S. Only edges included in the PC set are considered for the search. Thus, only the necessary edges are added. The complete algorithm is as shown in Figure 4.

**Hierarchical Clustering based learning of Bayesian Networks**
**Input:** *(variable set $X$, training data $D$)*
**Result:** *learned net $G$*
**for** $i \leftarrow 1$ ***to*** $m$ **do**
  $S_i \leftarrow MMPC(X_i, D)$
**end**
$S \leftarrow \{S_i, ...S_n\}$
MI $\leftarrow$ null
**for** $i \leftarrow 0$ ***to*** $m-1$ **do**
  **for** $j \leftarrow 0$ ***to*** $m-1$ **do**
    $MI[i,j] \leftarrow$*MI(i,j)*
  **end**
**end**
$c \leftarrow \emptyset$ **for** $i \leftarrow 0$ ***to*** $m-1$ **do**
  $y.add(X_i)$
  $c.add(y)$
**end**
$limit \leftarrow 20$
**while** $c.size() > 1$ **do**
  **for** $i \leftarrow 0$ ***to*** $c.size()-1$ **do**
    cluster1 $\leftarrow$ c.get(i)
    **for** $j \leftarrow i+1$ ***to*** $c.size()$ **do**
      cluster2 $\leftarrow$ c.get(j)
      clusterSum $\leftarrow$*AverageMI(cluster1, cluster2)*
      $maxSum \leftarrow argmax_{(c1,c2)} clusterSum$
    **end**
  **end**
  p $\leftarrow$ *Merge*(c1, c2)
  **if** *(p.size() $\geq$ limit)* **then**
    Generate set of data segments $D_k$ based on the values of X\X$\in$p
    G $\leftarrow$ search($D_k, p, S$)
  **end**
**end**
**return** $G$

Figure 4 Algorithm: Hierarchical Clustering based learning of Bayesian Networks

# CHAPTER 5. EXPERIMENTAL EVALUATION

In this section, an empirical evaluation of our algorithm (referred to as HC) with learning of Bayesian network structures under incremental construction curricula (referred to as CL) [3] is shown. The experiments are aimed at testing the ability of Bayesian network structure learning algorithms to recover Bayesian network structures from training data randomly sampled from the ground truth networks. Various evaluation metrics are used to measure the quality of the learned Bayesian networks. Also, a comparison of HC and WinOBS algorithm is shown.

## 5.1 Datasets

Table 1 Bayesian Networks used in experiments.

| Networks | Nodes | Arcs | Parameters |
|---|---|---|---|
| **Alarm** | 37 | 46 | 509 |
| **Andes** | 223 | 338 | 1157 |
| **Child** | 20 | 25 | 230 |
| **Hailfinder** | 56 | 66 | 2656 |
| **Hepar2** | 70 | 123 | 1453 |
| **Insurance** | 27 | 52 | 984 |
| **Pigs** | 441 | 592 | 5618 |
| **Water** | 32 | 66 | 10083 |
| **Win95** | 76 | 112 | 574 |

Nine benchmark Bayesian networks, collected from the bnlearn repository [6], are used in this project. The characteristics of the networks are as shown, in Table 1.

### 5.2 Experimental Setup

For each dataset, there are five different sample sizes, SS= (100, 500, 1000, 5000, 10000). The default parameters for curriculum learning algorithm, as mentioned in [3] are used. The step size is varied to 1, 2 and 3. The Bayesian network with the highest BDeu score is selected. The hyper-parameters a and b in the penalty function are set to 1000 and 100 respectively. For our algorithm of hierarchical clustering, the Bayesian network is learned once. The hyper-parameters a and b in the penalty function are set to 200 and 5000 respectively. The limit of minimum size of a cluster, from where learning starts is set to 20.

### 5.3 Evaluation Metrics

Four Metrics are used to evaluate the performance of the algorithms: Bdeu, BIC, KL and Time taken. These are calculated over a separate test dataset with 5000 samples for every Bayesian network. The BDeu is given by:

$$score_{BDeu}(G_i, D_i) = \log P\left(G_i, D_{i,j}\right)$$

In practice, the BDeu score is very sensitive with respect to the equivalent sample size. The BIC (Bayesian Information Criterion) score is only reasonable under certain assumptions. It is given by:

$$BIC(G:D) = \sum_{i=1}^{n}\sum_{j=1}^{q_i}\sum_{k=1}^{r_i} N_{ijk}\log\frac{N_{ijk}}{N_{ij}} - \frac{1}{2}\log(N)\sum_{i=1}^{n}(r_i - 1)q_i$$

where N: denotes the number of samples in the dataset,

n: denotes the number of variables,

$r_i$ : the number of values that $X_i$ can take,

$q_i = \prod_{X1 \in Pa_i} r1$ indicates the number of values that the parent set $Pa_i$ of $X_i$ can take,

$N_{ijk}$ is the number of samples in D where $X_i = k$ and $Pa_i = j$, and $N_{ij}$ is the number of samples with $Pa_i = j$ in D

BIC has two parts: the likelihood of the network structure after having seen the data, and a regularization term that constrains the model complexity that is measured by the number of parameters.

Kullback–Leibler divergence or KL divergence is a measure of how one probability distribution $P_L$ diverges from a second, expected probability distribution $P_T$ . It is given by:

$$KL(P_T, P_L) = \sum_X P_T(X) \log \frac{P_T(X)}{P_L(X)}$$

We use another simpler form of KL divergence [1] calculated using Shannon entropy and report only the last term of the equation, which is negative, so, the larger it is, the smaller the KL divergence. It is given as:

$$KL(P_T, P_L) = -H_{P_T}(X) + \sum_{X_i \in X} H_{P_T}(X_i) - \sum_{X_i \in X, Pa_L(X_i) \neq \emptyset} MI_{P_T}(X_i, Pa_L(X_i))$$

where $H_{P_T}$ denotes the Shannon entropy with respect to $P_T$. The values for BDeu and BIC are negative. The less negative the values of BDeu and BIC, the better the network. For KL score, the more positive the value, the better in our case.

### 5.4 Results

Table 2 gives a comparison between CL and HC on three metrics, BDeu, BIC and KL. Each number in the table is an average normalized score, i.e., the average ratios of the raw scores to that of CL (averaged over 9 networks). For BDeu and BIC, smaller ratios indicate better learning results; for KL, larger numbers indicate better learning results. Each number in parentheses indicates the number of winning networks among the 9 networks, i.e., on how many networks the algorithm produced better results than its competitor. For example, for BIC score for sample size 100, CL value is 1.00 and HC value is 0.928. Thus, HC does better in this case, in all 9 networks. On the contrary, the KL scores are positive. The higher the raw KL score, the better the network. For instance, for sample size 100, the average KL for CL is 1.00 and for HC is 1.662. Thus, HC is better, in all the nine networks.

| **Score** | **Algorithm** | **Sample Size** | | | |
|---|---|---|---|---|---|
| | | **100** | **500** | **1000** | **5000** |
| **BDeu** | HC | 0.980(7) | 1.003(2) | 1.002(1) | 1.005(2) |
| | CL | 1.000(2) | 1.000(7) | 1.000(8) | 1.000(7) |
| **BIC** | HC | 0.928(9) | 1.003(4) | 1.000(5) | 1.007(1) |
| | CL | 1.000(0) | 1.000(5) | 1.000(4) | 1.000(8) |
| **KL** | HC | 1.662(9) | 0.999(5) | 0.995(3) | 0.989(1) |
| | CL | 1.000(0) | 1.000(4) | 1.000(6) | 1.000(8) |

Table 2 Comparison between CL and HC on three metrics

The time taken should be less for better performance. It can be observed from Figure 5, that shows a comparison between the total time taken by curriculum learning and hierarchical clustering, for the largest sample size = 5000, HC finds a network in 9% of the time taken by CL. Hence, HC is faster.

Therefore, it can be observed that the hierarchical clustering algorithm finds Bayesian network much faster than the original curriculum learning algorithm. The point to note here is that there is a significant difference in the time taken for larger sample sizes. The largest difference is as huge as 3460-313=3145 seconds. For smaller sample sizes, the difference in time taken is rather trivial.

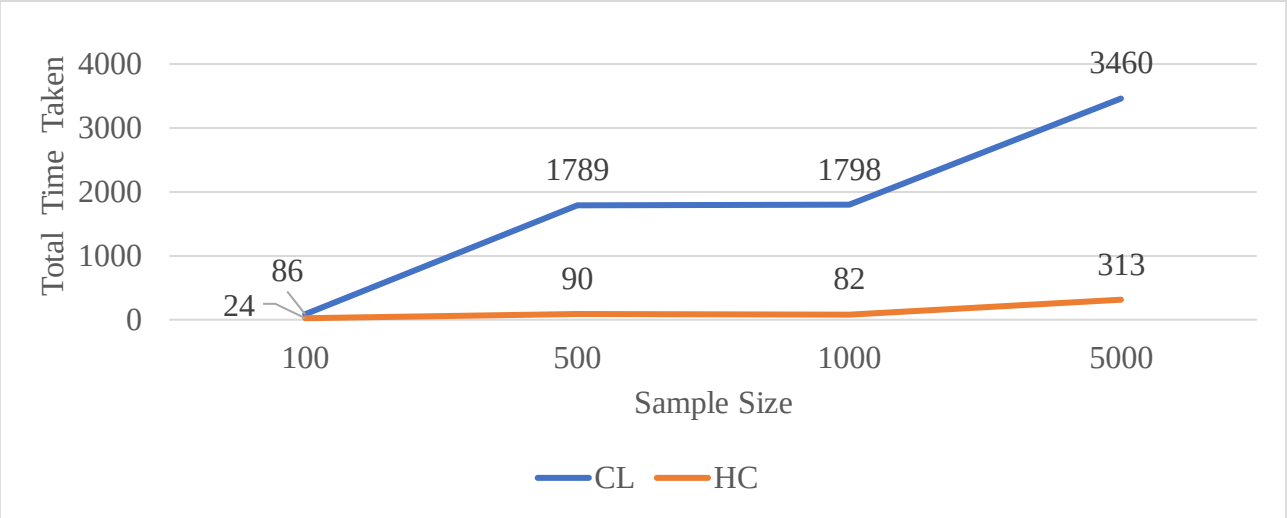


Figure 5 Comparison of Total Time taken by Curriculum Learning vs Hierarchical Clustering

Table 3 shows the results of comparing the performance of an ordering-based algorithm in the BLIP package [9], WinOBS. It is currently one of the best algorithms for structure learning. The BIC scores of Hierarchical Clustering and the WinOBS algorithm from BLIP package are shown for 9 datasets. The sample size of each file is 5000. The HC algorithm has

the same configuration. The WinOBS algorithm is run for one hour per file to generate the results. The other parameters are set to default.

The less negative the BIC score, the better is the learned Bayesian network. It can be observed that the HC algorithm learns better Bayesian networks for 6/9 datasets. For the remaining three datasets, the score for Insurance is very close for both the algorithms. However, for Pigs and Win95 datasets, there is a considerable difference.

| **Networks** | **BIC Scores** | |
|---|---|---|
| | HC | WinOBS |
| **Alarm** | -53147 | -206019 |
| **Andes** | -469472 | -572009 |
| **Child** | -60754 | -79543 |
| **Hailfinder** | -248763 | -251281 |
| **Hepar2** | -163156 | -164339 |
| **Insurance** | -67556 | -67455 |
| **Pigs** | -1667279 | -1070522 |
| **Water** | -64683 | -65235 |
| **Win95** | -50360 | -46439 |

Table 3 Comparison of HC with WinOBS

Hence, we show two sets of experiments. The first being a comparison of the Bayesian networks learned by hierarchical clustering with an incremental curriculum-based learning

method. We show various evaluation metrics including time taken. The second set of experiments shows competency with an ordering-based search algorithm.

# CHAPTER 6. CONCLUSION

Several approaches and ideas were used to get to the final hierarchical clustering algorithm. Instead of using the PC set from MMPC, another PC set, from BLIP package [8] was tried out. Another idea was to first learn a globally optimal small network of say 15 nodes, and then start the hierarchical clustering and learning process. Scoring functions other than the BDeu score were taken into consideration. A similarity measure other than mutual information was used to learn the Bayesian network, called log likelihood. These ideas did not yield the desired results, the only idea that stood out was of starting structural learning after a considerable sized cluster is formed. Hence, this resulted in our current approach of hierarchical clustering-based learning of Bayesian networks.

Overall, from the experimental results for different sample sizes over nine benchmark datasets, it can be concluded that there is a substantial difference between the time taken by our hierarchical clustering algorithm and the existing curriculum-based learning methods. This specially makes a difference when there are large datasets, or large sample sizes. The hierarchical clustering algorithm takes only 9% of the time of what incremental curriculum learning took to lean Bayesian networks from a sample size of five thousand. It gives a comparable performance with respect to an ordering-based algorithm, WinOBS, for six out of the nine datasets we test on.

Currently, work is going on to further improve the learned Bayesian network scores, using the idea of Community Detection. Once we learn a DAG using hierarchical clustering, community detection algorithms, like Fast Modularity can be applied on the learnt DAG, to extract communities of variables that are closely linked to each other. Bayesian networks are

learnt over each community and then the greedy search is run, learning a single Bayesian network. This idea has improved our results a bit.

## REFERENCES

[1] Y. Zhao , Y. Chen , K. Tu , J. Tian ,Curriculum learning of Bayesian network structures, in: Proceedings of the Seventh Asian Conference on Machine Learning, 45, JMLR, 2015, pp. 269–284 .

[2] Spirtes, P., Glymour, C., and Scheines, R. (2000). Causation, prediction, and search, volume 81.The MIT Press.

[3] Y. Zhao , Y. Chen , K. Tu , J. Tian , Learning Bayesian network structures under incremental construction curricula, in: Neurocomputing 258 (2017) 30–40

[4] Chickering, D. M. (1996). Learning Bayesian networks is NP-complete. In Learning from data, pages 121{130. Springer}.

[5] I. Tsamardinos , C.F. Aliferis , A. Statnikov , Time and sample efficient discovery of Markov blankets and direct causal relations, in: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2003, pp. 673–678 .

[6] http://www.bnlearn.com/bnrepository/

[7] Chen, Yetian, "Structure Discovery in Bayesian Networks: Algorithms and Applications" (2016). Graduate Theses and Dissertations.
15678.

[8] http://blip.idsia.ch/

[9] M. Scanagatta, G. Corani, M. Zaffalon, Cassio P. Campos, Learning Bayesian Networks with Thousands of Variables, in: Advances in Neural Information Processing Systems 28 (NIPS 2015).

[10] M. Scanagatta, G. Corani, M. Zaffalon, Improved Local Search in Bayesian Networks Structure Learning: Proceedings of Machine Learning Research vol 73:45-56, 2017.

[11] M. Teyssier, D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, UAI-05, pages 584–590, 2005.