Local Skeleton Discovery for Incremental Bayesian Network Structure Learning

Amanullah Yasin, Philippe Leray

Local Skeleton Discovery for Incremental Bayesian Network Structure Learning

Amanullah YASIN[1] and Philippe LERAY[2]

[1,2]*Knowledge and Decision Team,*
Laboratoire d'Informatique de Nantes Atlantique (LINA) UMR 6241,
Ecole Polytechnique de l'Université de Nantes, France
[1]*Balochistan University of Information Technology, Engineering*
and Management Sciences, Pakistan
{amanullah.yasin, philippe.leray}@univ-nantes.fr

***Abstract*—Nowadays there are a huge number of applications produce the immense amount of data in the form of a data stream, which needs real time analysis. Sensor networks, real-time surveillance and telecommunication systems are the examples of such applications. The real time analysis of the data stream leads to a number of computational and mining challenges. In this scenario new data arrives continuously and an efficient learning algorithm must be able to improve its learning accuracy by incorporating the time and memory constraints. This paper addresses the problem of incremental Bayesian network structure learning for high dimensional domains. The local skeleton discovery methods for Bayesian network structure learning are outperforming to deal with such domains. Here we transformed the local discovery algorithm Max-Min Parents and Childrens (MMPC) into an incremental fashion. We learned a set of candidate-parent-children for each variable by using incremental hill-climbing. The reduced search space saves a lot of computations and reduces the complexity. Our algorithm is then illustrated with a toy example.**

***Index Terms*—Statistical learning, Incremental learning, Data mining, Data stream, Bayesian network structure learning**

I. Introduction

Many sources produce data continuously like customer click streams, telephone call records, large sets of web pages, multimedia, scientific data and sets of retail chain transactions etc. Such type of data is called data stream so, it is a real time and continuous ordered sequence of items, which are not feasible to store nor possible to control the order [1].

Data stream mining means extracting useful information or knowledge structures from continuous data. It becomes the key technique to analyze and understand the nature of the incoming data. Typical data mining tasks, including association mining, classification, and clustering, helps to find interesting patterns, regularities, and anomalies in the data. However traditional data mining techniques cannot directly apply to data streams. This is because most of them require multiple scans of data to extract the information, which is unrealistic for data stream. More importantly the characteristics of the data stream may change over time and the evolving pattern needs to be captured [2]. Furthermore, we also need to consider the problem of resource allocation in mining data streams. Due to the large volume and the high speed of streaming data, mining algorithms must cope with the effects of a system overload. Thus how to achieve optimum results under various resource constraints becomes a challenging task.

Bayesian network (BN) is a graphical representation for joint probability distribution among a set of random variables. It has been used in many applications like medical diagnosis and financial forecasting. It represents by directed acyclic graphs (DAG), where nodes of the DAG represent the random variables and the arcs of the DAG are represents the direct influence of one variable to the other. The intensity of the influence between the variables is defined by the conditional probability. To find the best structure of the BN, which describes the data is called structure learning.

BN structure learning is proved to be a NP-Hard problem [3], it motivated to use of heuristic search methods to solve it. Many heuristics have been proposed in the literature, it can be classify in three categories for *Bayesian Network (BN) structure learning:*

- *Score and search* based methods are search over the space of all possible Bayesian networks in an attempt to find the network with a maximum score. Unfortunately, in such methods the search space is super-exponential in the number of random variables. It is very hard to compare all the structures specially in high dimensional domains [4]. Therefore, score-based methods are theoretically intractable despite the quality of the search heuristic in use.
- *Constraint based* methods, the main idea behind these types of methods is to exploit the independence semantics of the graph. They construct the graphical structure called "patterns" using statistical tests or information theoretic measures [5]. Later using different assumptions they direct the edges to get a directed acyclic graph (DAG). Its performance is limited with small conditioning set and criticized for complex structures.
- *Local Search* methods first search for the conditional independence relationships among the variables on a dataset and construct a local structure around a target variable i.e. Parent-Children (PC) or Markov Blanket (MB), using different heuristics like IAMB [6], MMPC [7], MBOR [8] and then they use another heuristic to learn the full BN structure. For instance MMHC [9] combines MMPC and greedy search approaches.

Applying BN structure learning in high dimensional domains e.g. biological or social networks, faces the problem of high dimensionality. These domains produce data sets with tens or hundreds of thousands of variables. The recent Max-Min Hill-Climbing algorithm (MMHC) [9] has been proposed to solve high dimensionality problem and it outperforms on a wider range of network structures. MMHC combines, both the local search and the score-and-search based approaches. In the first phase, it learns the possible skeleton of the network using the local discovery Max-Min Parent Children (MMPC) algorithm. While in the second phase, it orients the determined edges using the greedy hill-climbing search. Algorithms using local discovery for skeleton identification performs better than other leading non-hybrid structure learning algorithms in high dimensional domains [9]

Here we will focus on *Local Search* which is most scalable method. In this paper, we present an incremental local skeleton learning algorithm (iMMPC) to identify the set of candidate parent children (CPC) of a target variable in any BN which faithfully representing the distribution of data. We applied incremental hill climbing method to find a set of CPCs for a target variable and observed that it saves a considerable amount of computing time.

This paper is organized as follows: first we discussed the previous work already done in the field of incremental learning and data stream mining in the section [sec:Related-Work]. In section [sec:Backgroun] we recall the basics of the heuristics used in our proposed method. In section [sec:iMMPC-Approach] we present our incremental Local Search approach iMMPC. In section [sec:Toy-Example] we explained our method with an example. Finally we conclude in section [sec:Conclusion-and-Perspectives] with some proposals for future research.

II. Previous Work

In the field of incremental Bayesian Network Structure learning, some works proposed to iteratively revise the structure of a Bayesian network. It can be classified in two categories, approaches deal with the *stationary domains* and *non-stationary domains*. Algorithms which deal with *stationary domains* consider the data is drawn from single underlying distribution, which will not change with the passage of time. The models are also not very different when they evaluated with respect to similar data sets [10]. On the other hand, algorithms deal with non-stationary domains consider the underlying distribution of data may change with time so, drift or shift change may occur. Furthermore, we can divide these domains with respect to BN structure learning methods.

Buntine's [11] proposes a batch algorithm that uses the score-and-search based Bayesian approach, later he proposes some rules to convert it into an incremental algorithm. He considered two conditions, if there is time constraint for incremental or online learning then it updates the posterior probabilities only of the parent structure. Otherwise both structure and parameters will be updated.

Lam and Bacchus's [12] approach is also an extension of their batch algorithm and based on Minimal Description Length (MDL) principle. The idea is to first learn both partial network structure from the new data and existing network using the MDL learning method and then update the global old structure by using the newly learned partial structure.

Friedman and Goldszmidt [13] proposes three different approaches, first naive approach, where previously seen data is stored and then a batch learning procedure is called for each new example. Second approach based on Maximum Aposteriori Probability (MAP). Third approach called *incremental*. In this approach, a set of network candidates is maintained, which called the *frontier* of the search process. And at the arrival of new examples, it updates the stored information. Later it runs the search process to verify if any network in the frontier seems more appropriate than a current model.

Alcobe [14] adopted Chow and Liu [15] tree structure algorithm and proposed a heuristic ACO (Arches in Correct Order) to trigger the updating process when data invalidates the current structure. It rebuilds the network structure from the branch which is found to be invalidated. Later he proposed two heuristics to change a batch Hill-climbing search into an incremental one. Section III-B will describe more deeply these heuristics.

All above approaches deal with the stationary domains and use the scoring methods to learn BN structure incrementally.

A recent work in this field done by Nielsen and Nielsen [16] considers the non-stationary domains where a concept shift may occur. This approach consists of two mechanisms, first monitoring and detecting when and where the model should be changed and second relearning and using a local search strategy integrating the parts of the model that conflict with the observations.

As a conclusion we see that most of the algorithms use the *scoring* methods and treat the incremental process in different ways for *stationary domains*. Score based methods are not scalable for data stream mining, the judicious choice for handling a great number of variables is a local search approach. It's why we take advantages of local search approaches and proposed *i*MMPC algorithm: a scalable incremental method for stationary domains with high dimensionality, in section IV.

III. Background

A. MMPC(T): Local Search approach for BN Structure Learning

The MMPC(T) algorithm discovers the set of CPC (candidate parent-children, without distinguishing among both) for a target variable T. It is a combination of $\overline{MMPC}(T)$ algorithm and additional correction for symmetric test.

Algorithm $\overline{MMPC}(T)$ (cf. Algo. 1) has also two phases. In *forward phase* it adds variables in CPC for a target variable T and in *backward phase* it removes the false positives. The pivotal part of the MMPC algorithm is the *forward phase* of $\overline{MMPC}(T)$ algorithm. This phase starts from the empty set of CPC(T) for a target variable T, then sequentially adds the variables in CPC which have strong direct dependencies.

Function *min(Assoc)* used to measure the association between two variables given CPC, it is zero if these two

variables are independent conditionally any subset of CPC. So variables having zero associations given any subset of already calculated CPC, can never enter in CPC(T). This function estimates the strength of association by using any measure of association *Assoc* like χ^2, mutual information (MI) or G^2. Further *"MaxMinHeuristic"* [9] selects the variables which maximize the *min(Assoc)* with target variable T conditioned to the subset of the currently estimated CPC. *Forward phase* stops when all remaining variables are independent of the target variable T given any subset of CPC.

Algorithm 1 $\overline{MMPC}\,(T)$

Require: *target variable T; data D; threshold* θ
Output: *a set of Candidate Parent and Children (CPC) of T*

**** *Forward Phase: MaxMinHeuristic* ****
$CPC = \varnothing$
Repeat
$< F, assocF >= max_{x \in X \setminus CPC}\,(min_{S \subseteq CPC} Assoc\,(x;\, T \mid S))$
if $assocF \neq 0$ **then**
Add $(CPC,\, F)$
Endif
Until CPC has not changed

**** *Backward Phase:* ****
For all X $\in$CPC
if $\exists S \subseteq CPC$, s.t. $Assoc\,(X;\, T \mid S) < \theta$ **then**
Remove $(CPC,\, X)$
Endif
Return CPC

B. Incremental Adaption of Score Based BN Structure Learning

In the category of *score based methods*, here we are discussing the Alcobe [10] approach. In his approach, he proposed two heuristics to change a batch Hill-climbing search (HCS) into an incremental algorithm. We will first describe the usual non incremental HCS and then introduce Alcobe's heuristics.

1) Hill Climbing Search *:* HCS (cf. Algo. 2) methods traverse the search space called neighborhood by examining only possible local changes at each step and applying the one that maximizes the scoring function. The ***neighborhood*** of a model *M* consists of all models which can be built using the *operators op* and *argument pairs* A*, where* the operator can be *Add Edge*, *Delete Edge* or *Reverse Edge*. A ***scoring*** function $f(M,\, \mathcal{D})$ is used to measure the quality of the model. ***Search path*** or traverse path is a sequence of operators and argument pairs added on each step to obtain a final model M_f, in other words, it is a sequence of intermediate models.

Definition 1. (Search Path) Let M_0 be an initial model, *op* be an operator and A be an argument pair. Then the final modelM_f, is the model with highest score to the neighborhood, obtained by a hill-climbing search algorithm as:

$$M_f = op_n(...(op_2, (op_1,\, A_1),\, A_2), ...,\, A_n)$$

So search path is the sequence of operators and argument pairs $\mathcal{O}_{op} = \{(op_1,\, A_1)\,,\, (op_2,\, A_2)\,, \ldots\, (op_n,\, A_n)\}$ used to build M_f.

Models in a search path are in increasing quality score order.

$$f(M_0,\, \mathcal{D}) < f(M_1,\, \mathcal{D}) < f(M_2,\, \mathcal{D}) \cdots < f(M_f,\, \mathcal{D})$$

Algorithm 2 Hill Climbing Search (HCS)

Require: *Data* $\mathcal{D}$*; scoring function* $f(M,\, \mathcal{D})$*; a set of operators* $OP = \{op^1, \ldots, op^k\}$
Output: *DAG*: a model M of high quality

i = 0
$M_i = \varnothing$
Repeat
$oldScore = f(M_i,\, D)$
i++
$M_i = op(M_{i-1},\, A_i)$
** where $op(M_{i-1},\, A_i) = arg\,max_{(op^k,\, A) \in \mathcal{G}_n} f\left(op^k(M_{i-1},\, A_i),\, D\right)$**\
Until $oldScore \geq f(M_i,\, D)$
Return M_i

2) Incremental Hill Climbing Search (iHCS): In *iHCS* (cf. Algo. 3) Alcobe discussed two main problems: firstly, when and which part needs to update, secondly to calculate and store sufficient statistics. At each iteration, it repeats the search path by traversing the reduced DAG space. He proposed two heuristics *"Traversal Operators in Correct Order"* (TOCO) and "*Reduced search space*" (RSS). It is assumed that data is sampled from same distribution. The search space is also supposed not to change too much when few data are added in current dataset or new data slightly change the underlying distribution so, the scoring function imagined being a continuous over the space of datasets.

First heuristic verifies the learned model and its search path for new data. If the new data alter the leaning (search) path then it is worth to update an already learned model. Second heuristic applies when TOCO found the path is not valid so, it is necessary to revise the current structure. At each step of the search path, it stores the *k* models in a set $\mathcal{B}$ having the score closer to the best one. The set $\mathcal{B}$ reduces the search space by avoiding to explore those parts of the space where low quality models were found during former search steps.

IV. Incremental MMPC(T) Approach

Now we can present our local search algorithm iMMPC(T) for stationary domains. As already discussed the pivotal part of the MMPC(T) algorithm is the forward phase of a $\overline{MMPC}(T)$ algorithm. In this phase variables enter incrementally in the set of CPC(T).

Algorithm 3 Incremental Hill Climbing Search (*i*HCS)

Require: *data $\mathcal{D}$; scoring function $f(M, \mathcal{D})$, a set of operators $OP = \{op^1, \ldots, op^m\}$* and $\mathcal{B}_i$ is a set of *k* best operators and argument pairs for model M_i
Output: *DAG*: a model M of high quality

** TOCO **\

Verify previous search path: After evaluation of scoring function f over new data $\mathcal{D} \cup \mathcal{D}'$, let (op_j, A_j) be a last pair where the new data agree with previously learned search path.
$M_{ini} = (op_j, A_j)$
i = 0
M_i = M_{ini}

** RSS **\

Repeat
$oldScore = f(M_i, \mathcal{D})$
i++
$M_i = op(M_{i-1}, A_i)$
** where $op(M_{i-1}, A_i)= arg\,max_{(op^m, A) \in \mathcal{B}_i} f(op^m(M_{i-1}, A_i), \mathcal{D})$ **\
if ($M_i \neq M_{final}$) **then**
Recalculate $\mathcal{B}_k$
Endif
Until $oldScore \geq f(M_i, \mathcal{D})$

Forward phase of the $\overline{MMPC}(T)$ algorithm as a HCS:: First of all, let's demonstrate the forward phase of the $\overline{MMPC}(T)$ algorithm can be transformed into a Hill Climbing search using a specific search space (set of models). The idea of $\overline{MMPC}(T)$ (forward phase) is also the same as HCS, stepwise it generates a model by improving the quality function on each step. In $\overline{MMPC}(T)$ a model can be defined as:

Definition 2. (Model) Let T be a target variable and CPC is a set of candidate parent-children of T, without distinguishing among both parent and children variables. Then model M is an undirected graph which is defined by variables $V = T \cup CPC$ and edges $E = \{< T, x >, \forall\, x \in CPC\}$.

To measure the quality of a model, we define a scoring function as:

Definition 3. (Quality Measuring) Let D be a dataset with a faithful distribution P and M is a model. The quality of model M can be measured by a function $f(M, \mathcal{D})$, where $f(M, \mathcal{D}) = MI(T, CPC)$ and its value increases when good variables are added to CPC.

Operator also defined as:

Definition 4. (Operator) Let T is a target variable. Operator can be defined *by* $AddUndirectedEdge(T, X)$ which corresponds to add X to the set of CPC(T).

Mutual Information (MI) has the property that $MI(X, Y \cup W) \geq MI(X, Y)$ means MI always increasing by including additional variables. Furthermore, following property of conditional mutual information justify our choice of MI [17]:

Conditional mutual information between X and Y given a set of variables Z defined as:

$$MI(X, Y \cup W \mid Z) = MI(X, Y \mid Z) + MI(X, W \mid Z \cup Y) \quad (1)$$

If Z is an empty set and Mutual Information (MI) used to measure the strength of the *Assoc* (*Assoc* = *MI*) then equation 1 can be written as:

$$Assoc(T, CPC \cup X) = Assoc(T, CPC) + Assoc(T, X \mid CPC)$$

So to maximize $f(M, \mathcal{D})$ it needs to maximize $Assoc(T, X \mid CPC)$ and it is the *"MaxMinHeuristic"* of the MMPC(T) algorithm. At each step $\overline{MMPC}(T)$ algorithm searches within the *neighborhood* to improve the quality function and select the best model. This *neighborhood* space over the models obtained by applying the one operator *AddUndirectedEdge*(T, X).

With the above three definitions, we can easily describe the forward phase of the$\overline{MMPC}(T)$ algorithm works as hill climbing search.

A. Our Proposal of Incremental MMPC(T)

After showing the HCS behavior of the$\overline{MMPC}(T)$ algorithm (forward phase) we are able to adapt TOCO and RSS heuristics of Alcobe approach (Algo. 3) with model M, scoring function f and *operator* as previously defined. Our *Initial model* M_0 corresponds to an empty set of CPC ($M_0 = \{T\}$). Incremental MMPC(T) starts from an *initial model* M_0 and then searches in the neighborhood to find the best one. This iterative process continues until all remaining variables find the weak dependencies. So the *search path* in iMMPC is a sequence of the variables added in the set of CPC(T) incrementally. Each step of the search path can be called intermediate model and for each intermediate model (or search step) we store the *k* neighboring models having the association value very close to the best one, as a set $\mathcal{B}$ where *k* is a user input value, greater the value of *k* guarantees more significant model.

On the arrival of new data iMMPC(T) first verify the previously learned path and define the initial model M_i from which the search will be resumed. If the initial model is a best model then it will continue to improve an existing model in the light of a set $\mathcal{B}$ (set of best arguments) to introduce new variables in the final model, otherwise it will recalculate the set of best arguments. We maintain the set $\mathcal{B}$ in descending order. Relearning process limits the search space by using the set $\mathcal{B}$.

Backward phase of $\overline{MMPC}$ has no concern as calculations of *Assoc* remains the same in *i*MMPC.

V. Toy Example

Now we are providing a simple example for the better understanding of an incremental $\overline{MMPC}$(T) algorithm. The

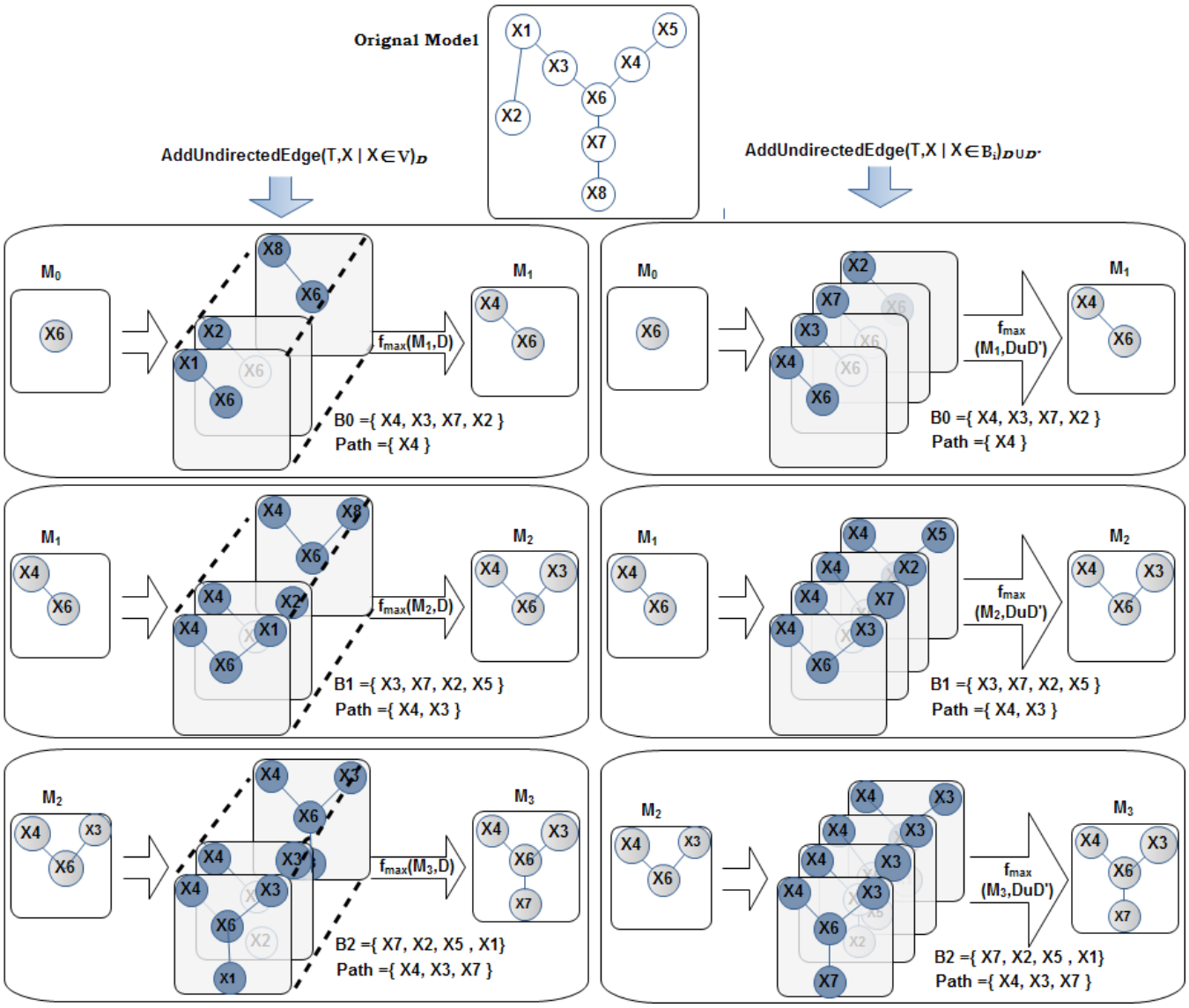

Figure 1. Toy example

original undirected acyclic graph of the incoming data stream is shown in the figure 1.

The data fed to the algorithm is sampled from the distribution of the original graph and we are interested in identifying $CPC(X6)$.

We are going to present two cases correspond to the two columns in figure 1. In the first one we handle data $\mathcal{D}$ and other is to handle data $\mathcal{D} \cup \mathcal{D}'$. First case will start by searching the whole neighborhood space and storing the reduced search space (with k = 4) $\mathcal{B}$. In second case the incremental algorithm will reduce the search space by using the reduced search space $\mathcal{B}$. So it starts from an initial model M_0 which contains an only target variable and empty set of *CPC*. In the first iteration, it generates the neighborhood by applying the operator $AddUndirectedEdge(M_0, X)$ and then calculate the association between $X6$ and each of the seven other variables. Suppose the maximum value is obtained for $X4$ variable, and $f(M_0) < f(M_1)$, so $X4$ is added in the $CPC(T)$. At this step, we store the four best variables which have the association value closest to the maximum one so, for instance set $\mathcal{B}0 = \{X4, X3, X7, X2\}$.

Next iteration starts from a model M_1 and repeats the same process. Suppose the maximum association value found by applying the $AddUndirectedEdge(M_0, X3)$ operator and $f(M_2) > f(M_1)$, so X3 is added in the $CPC\,(T)$. We store the four best variables which have the association value closest to the maximum one so, set $\mathcal{B}1 = \{X3, X7, X2, X5\}$. And the same in third iteration where $X7$ is added in the list of CPC, also the set of the best operator is stored as $\mathcal{B}2$.

Now we cannot proceed because there is no other model having the maximum value greater than $f(M_2)$. In other words we can say that the remaining variables have the zero association with the target variable given any subset of CPC.

On the arrival of new data $\mathcal{D}'$, our algorithm will re-learn the model for data $\mathcal{D} \cup \mathcal{D}'$. Again it starts from the initial model and generates the neighborhood by applying

the $AddUndirectedEdge(M_0, X)$ operator for only those variables which are stored in the set of best arguments found in the previous learning process. So here the search space reduced for only four models. Again supposed the variable *X*4 has a maximum association value so, it added in the list of *CPC*(T). Here we also verify the search path, if the search path is not in the same sequence then we need to recalculate the set of best operators.

The complexity of our algorithm if we consider the above the best case then at first time forward phase needs $3(n-2)$ comparisons and for next incremental stage it requires only $3k$ comparisons. In average case if new data changes then the complexity will be $\mathcal{O}(2k+n)$. For the worst case when the distribution of the entire data is changed or model learning process starts from scratch then it needs maximum comparisons $\mathcal{O}(3n)$.

This simple example illustrates the interest of our incremental approach. In high dimension, $k \ll n$ and lot of *Assoc* computations are saved.

VI. Conclusion and Perspectives

In this paper, we have proposed an incremental version of local search Max-Min Parent-children algorithm of Bayesian Network structure learning. We applied incremental hill climbing approach to discover the set of CPC of a target T. We store the most strong dependencies as a set of best arguments. It reduces the search space for new data by considering only the strong dependencies. This approach improves the performance of the algorithm systematically and reducing the complexity significantly.

In the further, we plan to carry out more systematic experimentation on real datasets to confirm the interest of applying TOCO and RSS heuristics in CPC discovery and then extend this work by incrementally identifying the whole structure. We also plan to apply optimize measuring by storing sufficient statistics. Furthermore, dealing with non-stationary domains by handling shift or drift detection.

Acknowledgment

We would like to thank the Higher Education Commission (HEC) of Pakistan and the KOD team of Polytech Nantes, France for supporting this research.

References

[1] L. Golab and M. T. Özsu, "Issues in data stream management," *SIGMOD Rec.*, vol. 32, no. 2, pp. 5–14, 2003.

[2] J. Gama, *Knowledge Discovery from Data Streams*. CRC Press, May 2010.

[3] D. Chickering, "Learning bayesian networks is NP-complete," in *Proceedings of AI and Statistics, 1995.*, 1995, pp. 121–130.

[4] D. Chickering, D. Geiger, and D. Heckerman, "Learning bayesian networks: Search methods and experimental results," in *Proceedings of Fifth Conference on Artificial Intelligence and Statistics*, 1995, pp. 112–128.

[5] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*, 2nd ed., ser. Adaptive Computation and Machine Learning Series, 2nd, Ed. The MIT Press, 2000.

[6] I. Tsamardinos, C. F. Aliferis, and E. Statnikov, "Algorithms for Large Scale Markov Blanket Discovery," in *The 16th International FLAIRS Conference*, 2003, pp. 376–380.

[7] I. Tsamardinos, C. F. Aliferis, and 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*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 673–678.

[8] S. Rodrigues De Morais and A. Aussem, "A novel scalable and data efficient feature subset selection algorithm," in *Proceedings of the European conference on Machine Learning and Knowledge Discovery in Databases - Part II*, ser. ECML PKDD '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 298–312.

[9] I. Tsamardinos, L. E. Brown, and C. F. Aliferis, "The max-min hill-climbing bayesian network structure learning algorithm," *Mach. Learn.*, vol. 65, no. 1, pp. 31–78, 2006.

[10] J. R. Alcobé, "Incremental hill-climbing search applied to bayesian network structure learning," in *First International Workshop on Knowledge Discovery in Data Streams. (KDDS)*, 2004.

[11] W. Buntine, "Theory refinement on bayesian networks," in *Proceedings of the seventh conference (1991) on Uncertainty in artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1991, pp. 52–60.

[12] W. Lam and F. Bacchus, "Using new data to refine a bayesian network," in *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann, 1994, pp. 383–390.

[13] N. Friedman and M. Goldszmidt, "Sequential update of bayesian network structure," in *Proc. 13th Conference on Uncertainty in Artificial Intelligence (UAI 97)*. Morgan Kaufmann, 1997, pp. 165–174.

[14] J. R. Alcobé, "An incremental algorithm for tree-shaped bayesian network learning," in *Proceedings of the 15th European Conference on Artificail Intelligence*. IOS Press, 2002, pp. 350–354.

[15] C. I. Chow and C. N. Liu, "Approximating discrete probability distributions with dependence trees," *IEEE Transactions on Information Theory*, vol. 14, pp. 462–467, 1968.

[16] S. H. Nielsen and T. D. Nielsen, "Adapting bayes network structures to non-stationary domains," *Int. J. Approx. Reasoning*, vol. 49, no. 2, pp. 379–397, 2008.

[17] L. M. de Campos, "A scoring function for learning bayesian networks based on mutual information and conditional independence tests," *J. Mach. Learn. Res.*, vol. 7, pp. 2149–2187, December 2006.