Incremental Bayesian network structure learning in high dimensional domains

Conference Paper · April 2013

DOI: 10.1109/ICMSAO.2013.6552635

2 authors:

Amanullah Yasin
8 PUBLICATIONS **11** CITATIONS
SEE PROFILE

Philippe Leray
University of Nantes
213 PUBLICATIONS **856** CITATIONS
SEE PROFILE

Incremental Bayesian network structure learning in high dimensional domains

Amanullah Yasin[1,2]
[2]Balochistan University of IT, Engineering and Management Sciences Quetta, Pakistan
Email: amanullah.yasin@univ-nantes.fr

Philippe Leray[1]
[1]Knowledge and Decision Research Group
Laboratoire d'Informatique de Nantes Atlantique (LINA)
Ecole Polytechnique de l'Université de Nantes, France
Email: philippe.leray@univ-nantes.fr

***Abstract*—The recent advances in hardware and software has led to development of applications generating a large amount of data in real-time. To keep abreast with latest trends, learning algorithms need to incorporate novel data continuously. One of the efficient ways is revising the existing knowledge so as to save time and memory. In this paper, we proposed an incremental algorithm for Bayesian network structure learning. It could deal with high dimensional domains, where whole dataset is not completely available, but grows continuously. Our algorithm learns local models by limiting search space and performs a constrained greedy hill-climbing search to obtain a global model. We evaluated our method on different datasets having several hundreds of variables, in terms of performance and accuracy. The empirical evaluation shows that our method is significantly better than existing state of the art methods and justifies its effectiveness for incremental use.**

I. Introduction

A rapid evolution took place in hardware and software technologies over recent years. Now applications are generating a huge amount of data continuously leading to rapid growth of databases, e.g., telecommunication and real-time surveillance systems, sensor networks, set of retail chain transactions, etc. In such applications, data is generated continuously and is supplied to decision support systems. These systems have to update their existing knowledge in the light of novel data. Incremental data mining has been an active area of research over the last few years. The problem aggravates for incoming data with high dimensional domains (several hundreds or thousands of variables).

In traditional batch learning scenario, the whole dataset is available for the algorithms. In a scenario where dataset is growing continuously and we cannot wait for its completion, the obtained model at a certain time will be outdated. So, revising the existing model by re-executing the batch learning algorithm will not only take a lot of resources as well as it is a costly venture. For this reason, incremental learning algorithms are needed in order to efficiently integrate the novel data with the existing knowledge.

Bayesian networks (BNs) are a powerful tool for graphical representation of the underlying knowledge in the data but BN structure learning is a proven NP-hard problem [5]. Therefore, various heuristics have been proposed for this task. Some of them have been recently developed to learn models from high dimensional domains, i.e *Max-Min hill-climbing (MMHC)* [20], *Bayesian substructure learning* [13], *Local to Global search* [10] and *Model based search applied for BN structure learning* [9]. Most of these algorithms use a local search heuristic where a local model is built around every single variable. Then, the global model is learned with the help of these local models. Unfortunately, all these works consider batch learning and need to relearn the model on availability of novel data, which is a very costly task.

Regardless of the traditional data mining techniques, incremental Bayesian network structure learning is not a mature field. There are few works addressing this issue and they differs with respect to the way of dealing with incrementality. Some works consider sliding windows, i.e. [3], [12], [7], [14], [23], and others considers landmark windows, i.e. [16], [15], [6]. These incremental approaches are mostly based upon the traditional "score-and-search" method, which is not realistic in high dimensional domains.

In this paper, we propose a novel method for incremental BN structure learning from high dimensional domains over a landmark window. We adopted the principle of *Max-Min hill-climbing* algorithm [20], one of the most robust structure learning algorithms for high dimensional domains.

This paper is organized as follows: section II describes some background. In section III we present our incremental method ($iMMHC$) for Bayesian network structure learning. In section IV, experiments varying the $iMMHC$ parameters are presented to show the flexibility of the algorithm and empirical evaluation of $iMMHC$ to justify its effectiveness. Finally, we conclude in section V with some proposals for future research.

II. Background

A. Bayesian network structure learning

A Bayesian network (BN) is a graphical representation of a probabilistic relationship among a set of random variables. It is composed of a directed acyclic graph (DAG) and a set of parameters, θ. In a DAG $\boldsymbol{G} = (X, E)$, X is a collection of nodes or vertices corresponding to the set of random variables $\{x_1, x_2, ...x_n\}$ and dependencies among these variables are expressed by the set of edges E. The parameters θ represent the probabilities of each random variable given its set of parents: $\theta_i = P(x_i|X_{Parents(i)})$

BN structure learning aims at selecting a probabilistic model that explains a given set of data. This task is a NP-hard problem [5]. There are three classical approaches often used for BN structure learning: First, *"constraint"* based methods consist in detecting (in)dependencies between the variables by performing conditional independence tests on data. The performance of these methods is limited to a small size of conditioning set and criticized for complex structures. Second, *"score-and-search"* based approaches are widely explored to find a BN structure. These methods, such as the *Greedy Search* (GS) algorithm proposed by [4], use a score function to evaluate the quality of a given structure and an heuristic to search in the solution space. The third *"hybrid"* approach merges the two previous approaches (constraint based and score based). The most recent hybrid methods dealing with high dimensional domains are decomposed into two phases. In first phase, they identify local structures around each variable. In second phase, they discover one final model by using some score based global optimization technique constrained with the previous local informations. Let us cite $MMHC$ [20] and some recent extensions [13], [9]. $MMHC$ first applies "constraint-based" heuristic *Max-Min parent-children (MMPC)* in order to identify local structures, candidate parents-children (CPC) set for every target variable. Then it uses a greedy search in order to optimize the global model, starting from an empty graph, with usual operators (*add_edge*, *remove_edge* and *reverse_edge*), but the *add_edge* operator can only add an edge $Y \rightarrow X$ if and only if $Y \in CPC(X)$ found in the previous phase.

B. Incremental BN structure learning with landmark windows

Algorithms presented in this section assume that data has been sampled from stationary domains, or with only small changes in the underlying probability distribution. These algorithms incrementally process data over a *landmark window* [8] where a window w_i contains data from initial to current time $i * \Delta_w$ where Δ_w is the window size.

In [16], Roure proposed two heuristics to transform a batch "score-based" BN structure learning technique (GS) into an incremental one. First heuristic is called *Traversal Operator in Correct Order (TOCO)*. It keeps the search path in former learning step and at the arrival of new data, it checks the order of the search path (i.e. the sequence of intermediate models). If the order is still held then there is no need to revise the structure, otherwise it triggers the GS to obtain a new model. Second heuristic is called *Reduced Search Space (RSS)* and it applied when the current structure needs to be revise. It reduces the search space by considering only high quality models found in the previous search step and avoid exploring low quality models.

Recently, Shi and Tan [18] proposed an hybrid method for incremental BN structure learning. It builds a local structure in two ways, using maximum spanning tree ($MWST$) algorithm (called *TSearch*) or using feature selection techniques. Later, for global optimization, greedy search is applied on these local structures by starting from previously obtained BN. In this algorithm, handling incremental datasets is limited to the greedy search and not performed in the local search phase, core of hybrid methods. The same authors proposed another (very similar) technique in [19], which is also not feasible for high dimensional domains in incremental environments due to large conditioning set of variables for independence test. The results presented deal with a limited number of variables (maximum of 39 variables).

In a preliminary work, we proposed an incremental version of the "constraint-based" local discovery $MMPC$ algorithm [22] which uses $TOCO$ and RSS heuristics to discover the local structure over a target variable.

III. Incremental MMHC

The main idea of our proposal, *incremental MMHC* (*$iMMHC$*) is to incrementally learn a high quality BN structure by reducing the learning time. It can be achieved by re-using the previous knowledge and reducing the search space. Like $MMHC$ method, $iMMHC$ is also a two phases hybrid algorithm as described in Algorithm 1 and illustrated in Figure 1. Both phases are incremental in their nature.

A. Incremental local search

This first phase discovers the possible skeleton (undirected graph) $\mathcal{G}$ of the network for a given window, by using *iMMPC* method (cf section II-B). It learns a local structure around a given target variable by using the previous knowledge, and avoids exploring parts of the search space which were previously found with low quality. For this purpose, it maintains a list of search paths (order of the variables included in the set of candidate parent-children "CPC"). At the same time, it stores a set of top *K* best variables that have more chance to be considered in CPC. The minimum size of set *K* depends upon the average degree of the graph. When new data arrive, $iMMPC$ checks every search path. If they are yet validated, there is no need to re-learn the CPCs (local models). On the contrary, it triggers the re-learning process. The detail of the algorithm and the description of the parameters can be found in [22]. Hence, we build the local structures incrementally by shrinking the search space, then we build an undirected graph $\mathcal{G}$ (skeleton) by merging these local structures.

B. Incremental global optimization

In the second phase, a greedy search is initiated for global optimization. A naive application of $MMHC$ would start this greedy search from the empty graph. Here we propose an incremental optimization phase by starting the greedy search process from the graph obtained in the previous time window (like Shi and Tan [18]) . The greedy search considers adding new edges discovered in the new skeleton $\mathcal{G}$ but also removing the outdated edges. We apply the operators *add_edge*, *remove_edge* and *reverse_edge*, where *add_edge* can add an edge $Y \rightarrow X$ if and only if Y is a neighbor of X in $\mathcal{G}$. By this way, $iMMHC$ keeps the sense of incremental learning by considering the previously learned model and revising the existing structure in the light of new data, as summarized in Figure 1.

Algorithm 1 iMMHC(D_w)

```
Require: Data of time window $w_i$ ($D_{w_i}$), previous top $K$ best neighbors
    for each variable ($\mathcal{B}$), previous BN structure ($BN_{i-1}$)
Ensure: BN structure ($BN_i$)
    % Incremental local identification
 1: for j = 1 to n do
 2:     $CPC(x_j)$=iMMPC($x_j, D_{w_i}, \mathcal{B}$)
 3: end for
    % Incremental greedy search
 4: starting model $\mathcal{M}$ = empty graph or $BN_{i-1}$.
 5: Only try operators add_edge ($Y \to X$) if $Y \in CPC(X)$
 6: (no constraint for remove_edge or reverse_edge)
 7: and return $BN_i$ the highest scoring DAG found
```

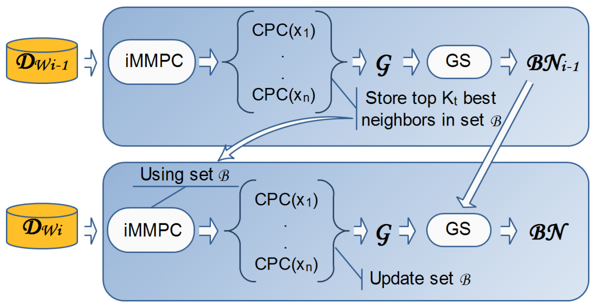

Fig. 1. iMMHC outline : dependencies between $iMMHC$ execution for window w_i and previous results.

C. Comparison with existing methods

iMMHC vs TSearch: $iMMHC$ is a two phase hybrid algorithm as $TSearch$ but both phases of $iMMHC$ are incremental while only the second phase of Tsearch deal with this problem. Another significant factor is the robustness of the first phase of the $iMMHC$, by using $iMMPC$, better than a simple tree approximation.

iMMHC vs Roure: In $iMMHC$, greedy search has two constraints, a previous graph and a CPC skeleton $\mathcal{G}$, therefore it has already very limited search space. These constraints increase the accuracy of classic greedy search as well as reduce its search space to a great extent.

IV. Empirical Evaluation

We carry out several experiments comparing $iMMHC$ with the most recent state of the art. Our goals are to evaluate its ability to deal with high dimensional data (up to hundreds of variables) and characterizing the situation where it outperforms the other algorithms.

A. Experimental protocol

Benchmarks: We took four well-known networks "Alarm", "Barley", "Hailfinder" and "Pathfinder" from GeNIe/SMILE network repository[1]. To test the performance of our algorithm in high dimensional domain, We also generated a network with thousands of variables using BN tiling [21] method (implemented in Causal Explorer[2]). For this purpose, 28 copies of Alarm network are tiled so that obtained network "Alarm28" containing 1036 variables. Subsequently, we sampled five datasets from all these Bayesian networks. We also used five "Gene" datasets from $MMHC$ source website[3]. Numerical characteristics of the networks and datasets are summarized in Table I. To create an incremental scenario, we feed data to the algorithms with window sizes Δ_w = 1000, 2000 and 3000.

Algorithms: We tested *iMMHC* algorithm in two different scenarios $iMMHC_{\emptyset}$ and $iMMHC_G$ as proposed in (cf section III), (starting the greedy search phase from an empty graph or the previously obtained model). We compared our algorithm with batch *MMHC* (without TABU search) and with the incremental *TSearch* described in section II-B. We implemented the original algorithms as described in their articles, in C++ using Boost graph[4] and ProBT[5] libraries.

In greedy search, we used the BIC score function. Independence is measured with the help of Mutual Information (MI). The confidence level, alpha has traditionally been set to either 0.01 or 0.05 in the literature on constraint-based learning, so we have decided to test 0.05 in the experiment.

Benchmark	#Variables	#Edges	Cardinality	#Instances	Av. Deg.	Max. Deg.
Alarm	37	46	2-4	20,000	2.5	6
Barley	48	84	2-67	20,000	3.5	8
Hailfinder	56	66	2-11	20,000	2.4	17
Pathfinder	109	195	2-63	20,000	1.7	106
Alarm28	1036	1582	2-4	10,000	3	8
Gene	801	972	3-5	5,000	2.4	11

TABLE I
Name of the benchmark, number of variables and edges, cardinality and degree of the each graph used for our experiments.

B. Evaluation measures

We used two metrics to evaluate our algorithm in terms of *computational efficiency* and *model accuracy*. The main task in learning algorithms is to compute score functions. The complexity of the algorithm depends upon the number of total score *function calls* (i.e. in our case, it is equal to MI calls for independence tests and local score function calls during the greedy search). We remind the readers that the skeleton discovery phase of $iMMHC$ algorithm use constraint based approach, which has already less complexity than score-and-search based techniques. The total function calls are cumulative sum over each window and are logarithmic scaled for better understanding. We used the Structural Hamming Distance (SHD) proposed in [20] for *model accuracy*. This distance compares the learned and original network structures,without penalizing an algorithm for structural differences that cannot be statistically distinguished. The results presented

[1]http://genie.sis.pitt.edu/networks.html
[2]http://www.dsl-lab.org/causal_explorer/

[3]http://www.dsl-lag.org/supplements/mmhc_paper/mmhc_index.html
[4]http://www.boost.org/
[5]http://www.probayes.com/index.php

in this paper are the mean and standard deviation of each metric obtained over five datasets corresponding to a given benchmark model.

C. Results and discussion

Figure 2 describes the SHD obtained at each time window and Figure 3 shows the $log10$ values of the total number of function calls over an incremental process for the algorithms $iMMHC_G$, $iMMHC_\varnothing$, batch $MMHC$ and $TSearch$.

***Best parameters*:** We can observe from these figures that the computational complexity of $iMMHC_G$ is very low as compared to $iMMHC_\varnothing$, with a higher accuracy except for datasets having high cardinality ("Barley " and "Pathfinder").

Parameter *K* is used in $iMMPC$ fro caching top *K* most associated variables with the target variable for later use in novel data, because they have more chances to become parent or children of a target variable. The behavior of $iMMHC$ with different *K* values is shown in Figure 5. We can see that the value of K could be between average and maximum degree of a theoretical graph. Logically, the complexity linearly increases with respect to K. As a comparison, usual structure learning algorithms have to limit the maximum number of parents in order to be scalable. Our algorithm only has to limit the number of neighbors to be stored in the cache, which is independent of the number of parents of the final DAG. We are then able to control the scalability of our algorithm without controlling the complexity of the final model. Figure 4 shows the role of different window sizes on incremental learning. The window should have sufficient amount of data, so that algorithm can find reliable set of K variables, having a low window size can negatively effects the learning process. We found that the ideal window size for the benchmarks used here is about 2000.

iMMHC for incremental learning: Figures 2 and 3 shows that $iMMHC_G$ outperforms $TSearch$ with respect to complexity and accuracy, except in "Hailfinder" where *TSearch* has a similar complexity as $iMMHC_G$ and also has a better accuracy than $iMMHC_\varnothing$ and batch $MMHC$. We observed that during $TSearch$ learning, the skeleton discovery phase ($MWST$) contains a lot of false-positive edges. Consequently, these errors propagate to the final structure. Since this structure is also used as an input for the next time window, the local errors in skeleton discovery will mislead the incremental structure learning process. As another consequence of false-positive edges, the complexity of the global optimization phase is also increased.

The robustness of $iMMHC$ can be explained by the fact that it uses an incremental adaptation of $MMPC$ algorithm for skeleton discovery phase which has been proven as a robust solution, limiting the number of false-positives.

Roure compared the accuracy of his algorithm by the ratio of incremental and batch score of the final model, i.e. *score(incremental)/score(batch)*. If this ratio is less than one, its means that an incremental algorithm is better than a batch one and vice versa. He obtained "1.002" value for alarm network in [16]. In contrast, the ratio obtained by $iMMHC_G$ is "0.9561", with same experimental design. Therefore, the ratio value smaller than one denotes that the model obtained by $iMMHC_G$ is better than a model obtained by batch $MMHC$.

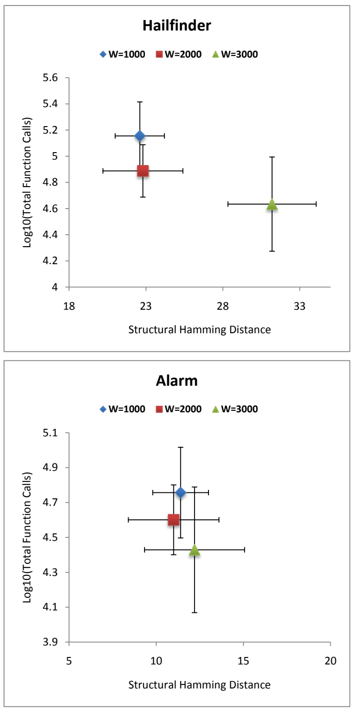

Fig. 4. Performance of iMMHC algorithm for windows size 1000, 2000 and 3000 (average and standard deviation for final models)

Incremental versus batch MMHC: We can observe from the Figure 2 that the model obtained by $iMMHC_G$ has a better quality than the model obtained by the batch $MMHC$, except for the datasets having high cardinality ("Barley " and "Pathfinder"). Despite this, the $iMMHC_\varnothing$ obtained the same accuracy as batch $MMHC$ with a low complexity. This result is consistent with Roure's work in [16]. With respect to the high dimension of the search space, incremental algorithms can avoid being trapped in some local optima as could be their batch counterparts. Despite this, we can see at Figure 3 that the number of total function calls are a little bit higher than batch one, but it is acceptable in the cost of better quality.

iMMHC for high dimensional domains: To check the ability of our algorithm to deal with high dimensional domains, tests were conducted with "Alarm28 " (1028 variables) and "Gene" (801 variables) benchmarks. We can observe that the results of $iMMHC_G$ in Figures 2 and 3 are much better than others. $iMMHC$ is an incremental algorithm where the first iteration has the same complexity as the batch algorithm but in the succeeding iterations, this complexity rapidly decreases. For this reason, $iMMHC$ can be adopted for several thousands of variables in incremental environment.

V. Conclusions and future work

In this paper, we proposed an incremental approach $iMMHC_G$ for BN structure learning in high dimensional domains over landmark windows. The both phases of

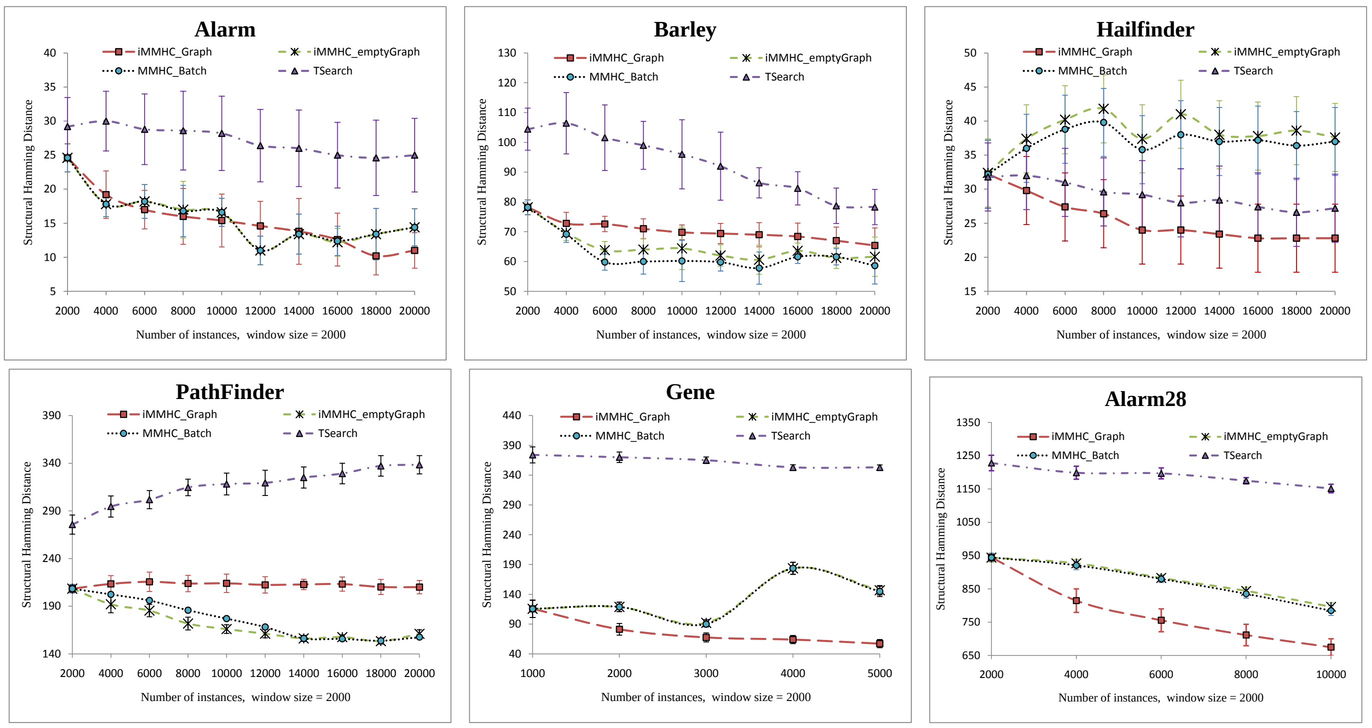

Fig. 2. Comparisons of Structural Hamming distance (SHD) for $MMHC, iMMHC_{\varnothing}, iMMHC_G$ and $TSearch$ algorithms using window of size 2000 (1000 for "Gene")

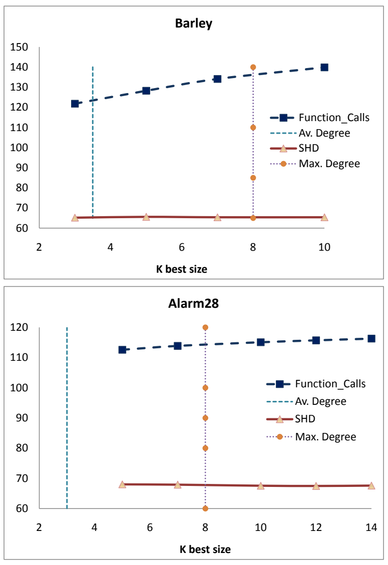

Fig. 5. Performance of iMMHC algorithm for different K values (Function calls are divided by 10^2 for "Barley" and 10^5 for "Alarm28", and SHD by 10^1 for "Alarm28" for ease of exposition)

$iMMHC_G$ deal with data in an iterative way by using (1) our $iMMPC$ algorithm for incremental local structure identification and (2) a constrained greedy search for global model optimization starting from the previously obtained model. By this way, we are able to take into account the previous knowledge in order to reduce the search space for incremental learning. Our experimental results illustrate the good behavior of $iMMHC_G$ compared to a similar incremental algorithm ($TSearch$) and the batch original one ($MMHC$). As a consequence $iMMHC_G$ could also be an interesting alternative to batch learning for large databases.

The $iMMHC$ algorithm, but also other BN incremental learning algorithms, learns over landmark windows. This is not sufficient for real applications such as data stream mining. One immediate perspective is the adaptation of $iMMHC$ in order to deal with *sliding windows* for unbounded data streams by keeping the less possible information about past data and dealing with non-stationarity. In this context, storing sufficient statistics of past data with ADtrees [2] can be possible, with a sequential update of these models as proposed in [17]. Using forgetting coefficient [1] is also a first solution for taking into account non-stationarity.

References

[1] Christoforos Anagnostopoulos, Dimitris K. Tasoulis, Niall M. Adams, and David J. Hand. Temporally adaptive estimation of logistic classifiers on data streams. *Adv. Data Analysis and Classification*, 3(3):243–261, 2009.

[2] Brigham Anderson and Andrew Moore. Ad-trees for fast counting and for fast learning of association rules. In *Proceedings Fourth International Conference on Knowledge Discovery and Data Mining*, pages 134–138. ACM Press, 1998.

[3] Wray Buntine. Theory refinement on Bayesian networks. In *Proceedings of the seventh conference (1991) on Uncertainty in artificial intelligence*, pages 52–60, San Francisco, CA, USA, 1991. Morgan Kaufmann Publishers Inc.

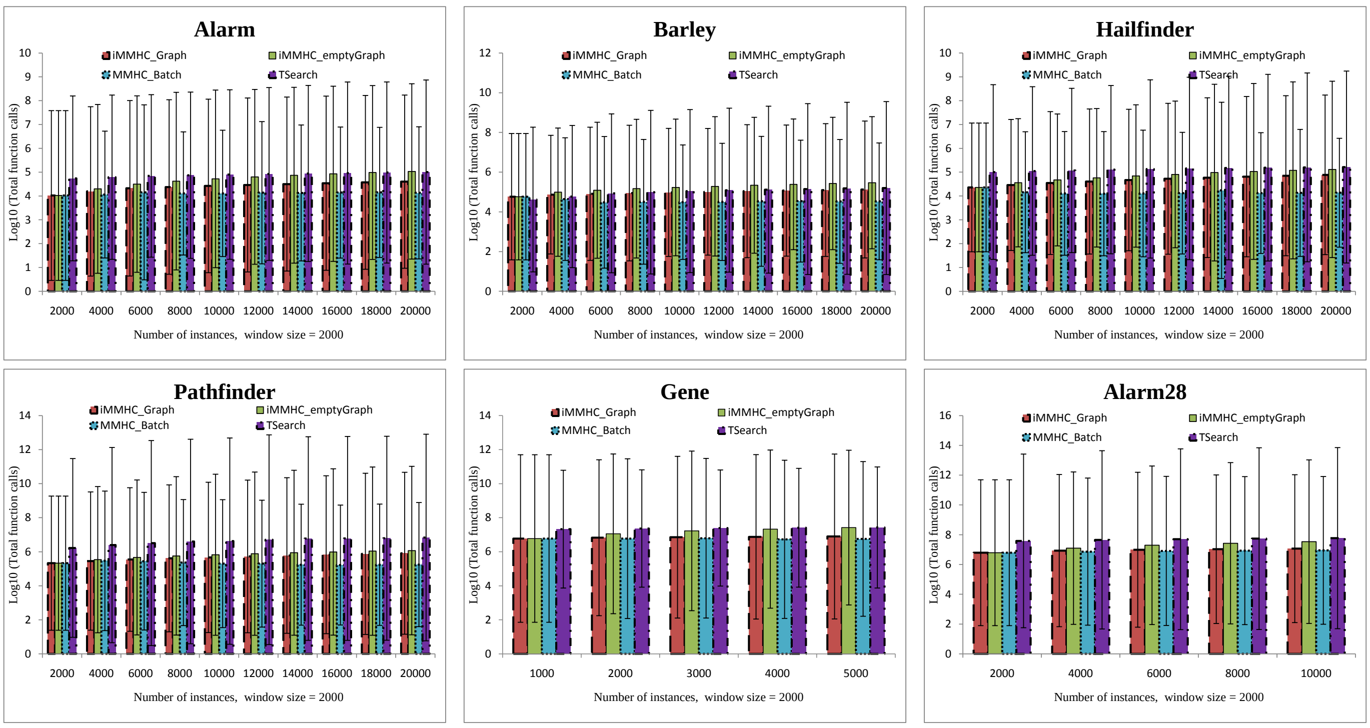

Fig. 3. Comparisons of Log10(total function calls) for $MMHC, iMMHC_{\varnothing}, iMMHC_G$ and $TSearch$ algorithms over an incremental process (using window of size 2000 (1000 for “Gene”))

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

[5] David M. Chickering. Learning Bayesian networks is NP-complete. In *Proceedings of AI and Statistics, 1995.*, pages 121–130, 1995.

[6] Denver Dash and Marek J. Druzdzel. A hybrid anytime algorithm for the construction of causal models from sparse data. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, UAI’99, pages 142–149, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[7] Nir Friedman and Moises Goldszmidt. Sequential update of Bayesian network structure. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence (UAI 97)*, pages 165–174, 1997.

[8] Joao Gama and Mohamed Gaber. *Learning from Data Streams – Processing techniques in Sensor Networks*. Springer, 2007.

[9] Avi Herscovici and Oliver Brock. Improving high-dimensional Bayesian network structure learning by exploiting search space information. Technical report, Department of Computer Science, University of Massachusetts Amherst, 2006.

[10] Kyu Baek Hwang, Jae Won Lee, Seung-Woo Chung, and Byoung-Tak Zhang. Construction of large-scale Bayesian networks by local to global search. In *Proceedings of the 7th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*, PRICAI ’02, pages 375–384, London, UK, UK, 2002. Springer-Verlag.

[11] Markus Kalisch and Peter Bühlmann. Estimating high-dimensional directed acyclic graphs with the pc-algorithm. *J. Mach. Learn. Res.*, 8:613–636, May 2007.

[12] Wai Lam and Fahiem Bacchus. Using new data to refine a Bayesian network. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 383–390, 1994.

[13] Andreas Nägele, Mathäus Dejori, and Martin Stetter. Bayesian substructure learning - approximate learning of very large network structures. In Joost Kok, Jacek Koronacki, Raomon Mantaras, Stan Matwin, Dunja Mladenic, and Andrzej Skowron, editors, *Machine Learning: ECML 2007*, volume 4701 of *Lecture Notes in Computer Science*, pages 238–249. Springer Berlin / Heidelberg, 2007.

[14] Søren Holbech Nielsen and Thomas D. Nielsen. Adapting Bayes network structures to non-stationary domains. *Int. J. Approx. Reasoning*, 49(2):379–397, 2008.

[15] Josep Roure. An incremental algorithm for Tree-shaped Bayesian network learning. In *Proceedings of the 15th European Conference on Artificail Intelligence*, pages 350–354. IOS Press, 2002.

[16] Josep Roure. Incremental hill-climbing search applied to Bayesian network structure learning. In *First International Workshop on Knowledge Discovery in Data Streams. (KDDS-ECML)*, pages 61–62, 2004.

[17] Josep Roure and Andrew W. Moore. Sequential update of ADtrees. In *ICML ’06: Proceedings of the 23rd international conference on Machine learning*, pages 769–776, New York, NY, USA, 2006. ACM.

[18] Da Shi and Shaohua Tan. Hybrid incremental learning algorithms for Bayesian network structures. In *Proc. 9th IEEE Int Cognitive Informatics (ICCI) Conf*, pages 345–352, 2010.

[19] Da Shi and Shaohua Tan. Incremental learning bayesian network structures efficiently. In *Proc. 11th Int Control Automation Robotics & Vision (ICARCV) Conf*, pages 1719–1724, 2010.

[20] Ioannis Tsamardinos, Laura E. Brown, and Constantin F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. In *Machine Learning*, pages 31–78. Springer Netherlands, Inc., 2006.

[21] Ioannis Tsamardinos, Alexander R. Statnikov, Laura E. Brown, and Constantin F. Aliferis. Generating Realistic Large Bayesian Networks by Tiling. In *Proceedings of the Nineteenth International Florida Artificial Intelligence Research Society Conference*, pages 592–597, California, USA, 2006. AAAI Press.

[22] Amanullah Yasin and Philippe Leray. iMMPC: a local search approach for incremental Bayesian network structure learning. In *Proceedings of the 10th international conference on Advances in intelligent data analysis X*, IDA’11, pages 401–412, Berlin, Heidelberg, 2011. Springer-Verlag.

[23] Yifeng Zeng, Yanping Xiang, and S. Pacekajus. Refinement of Bayesian network structures upon new data. In *The 2008 IEEE International Conference on Granular Computing, GrC 2008, Hangzhou, China, 26-28 August 2008*, pages 772–777. IEEE, 2008.