Dynamic MMHC: A Local Search Algorithm for Dynamic Bayesian Network Structure Learning

Ghada Trabelsi[1,2], Philippe Leray[2], Mounir Ben Ayed[1], and Adel Mohamed Alimi[1]

[1] Research Group on Intelligent Machines,
National School of Engineers (ENIS) of Sfax, University of Sfax, Tunisia
[2] Knowledge and Decision Team, Laboratoire d'Informatique de Nantes
Atlantique (LINA), University of Nantes, France
{ghada.trabelsi,mounir.benayed,adel.alimi}@ieee.org,
philippe.leray@univ-nantes.fr

Abstract. Dynamic Bayesian networks (DBNs) are a class of probabilistic graphical models that has become a standard tool for modeling various stochastic time-varying phenomena. Probabilistic graphical models such as 2-Time slice BN (2T-BNs) are the most used and popular models for DBNs. Because of the complexity induced by adding the temporal dimension, DBN structure learning is a very complex task. Existing algorithms are adaptations of score-based BN structure learning algorithms but are often limited when the number of variables is high. We focus in this paper to DBN structure learning with another family of structure learning algorithms, local search methods, known for its scalability. We propose Dynamic MMHC, an adaptation of the "static" MMHC algorithm. We illustrate the interest of this method with some experimental results.

Keywords: Dynamic Bayesian networks, structure learning, scalability, local search methods.

1 Introduction

Bayesian networks (BNs) are one of the most complete and consistent formalisms for the acquisition and representation of knowledge and for reasoning from incomplete and/or uncertain data. Structure learning of these models from data is an NP-hard problem [1]. Many studies have been conducted on this subject, leading to three different families of approaches: (1) constraint-based methods, (2) score-based methods, and (3) hybrid methods combining the advantages of both previous families. These last methods deal with local structure identification and global model optimization constrained with these local information. These methods are able to scale to distributions with more than thousands of variables.

Dynamic Bayesian networks (DBNs) are a general and flexible model class for representing complex stochastic processes [9] and are used in several areas such as speech recognition, target tracking and identification or genetics. Because of the complexity induced by adding the temporal dimension, DBN structure learning is also a very complex task. Existing algorithms [6,16,17,20,8] are adaptations of score-based BN structure learning algorithms, but are often limited when the number of variables is high.

Some others more scalable algorithms [5,21,18] have been proposed for a subclass of DBNs.

We focus in this paper to DBN structure learning with local search methods, by adapting the MMHC algorithm proposed by Tsamardinos and al. [14], one of the "state of the art" algorithms of this family. We claim that these local search algorithms can easily take into account the temporal dimension.

Section 2 provides the background of our work with a brief introduction to DBN structure learning and to MMHC algorithm. In section 3, our proposed algorithm *Dynamic MMHC* is explained in three sub-algorithms. We present the related works in section 4. Section 5 describes our experimental results. Finally, section 6 presents conclusions and perspectives.

2 Background

2.1 Dynamic Bayesian Networks

A DBN is a probabilistic graphical model devoted to represent sequential systems [9]. More precisely, a DBN defines the probability distribution over $\mathbf{X}[t]$ where $\mathbf{X} = \{X_1 \dots X_n\}$ are the n variables observed along discrete time t.

In this work, we consider a special class of DBNs, namely the 2-Time slice BN (2T-BN). A 2T-BN is a DBN which satisfies the Markov property of order 1 $\mathbf{X}[t-1] \perp \mathbf{X}[t+1] \mid \mathbf{X}[t]$. As a consequence, a 2T-BN is described by a pair $(M_0, M_\rightarrow)$.

M_0 (**initial model**) is a BN representing the initial joint distribution of the process $P(\mathbf{X}[t=0])$ and consisting of a direct acyclic graph (DAG) G_0 containing the variables $\mathbf{X}[t=0]$ and a set of conditional distributions $P(X_i[t=0] \mid pa_{G_0}(X_i))$ where $pa_{G_0}(X_i)$ are the parents of variable $X_i[t=0]$ in G_0.

$M_\rightarrow$ (**transition model**) is another BN representing the distribution $P(\mathbf{X}[t+1] \mid \mathbf{X}[t])$ and consisting of a DAG $G_\rightarrow$ containing the variables in $\mathbf{X}[t] \cup \mathbf{X}[t+1]$ and a set of conditional distributions $P(X_i[t+1] \mid pa_{G_\rightarrow}(X_i))$ where $pa_{G_\rightarrow}(X_i)$ are the parents of variable $X_i[t+1]$ in $G_\rightarrow$, parents which can belong to time t or $t+1$.

2.2 Local Search Algorithms

Local search algorithms are hybrid BN structure learning methods dealing with local structure identification and global model optimization constrained with these local information.

Several local structure identifications for static BNs have been proposed, dedicated to discover the candidate Parent-Children (PC) set of a target node algorithm [13,10] or the Markov Blanket (MB) i.e. parents, children and spouses, of the target node [15,11,10]. If the global structure identification is the final goal, Parent-Children identification is sufficient in order to generate a global undirected graph which can be used as a set of constraints in the global model identification. For instance, the recent Max-Min Hill-Climbing algorithm (MMHC) (cf. algorithm 1) proposed by Tsamardinos and al. [14] combines the local identidication provided by Max-Min Parent Children (MMPC) algorithm [13] and a global greedy search (GS) where the neighborhood of a given graph

Algorithm 1. MMHC(D)

```
Require: Data ($D$)
Ensure: BN structure ($DAG$)
1: $G_c \leftarrow \phi, G \leftarrow \phi, S \leftarrow 0$
   % Local identification
2: for all $X \in \mathbf{X}$ do
3:     $CPC_X$=MMPC($X, D$)
4: end for
5: for all $X \in \mathbf{X}$ And $Y \in CPC_X$ do
6:     $G_c \leftarrow G_c \bigcup$ (X,Y)
7: end for
   % Greedy search (GS) optimizing score function in DAG space
8: $Test \leftarrow$ True, $S \leftarrow$ Score(G,D)
9: while $Test$=True do
10:    $N \leftarrow$ Generate_neighborhood($G$,$G_c$)
11:    $G_{max}$= $\arg\max_{F \in N}$Score(F,D)
12:    if Score($G_{max}$,D) > S then
13:        G $\leftarrow G_{max}$
14:        S $\leftarrow$ Score($G_{max}$,D)
15:    else
16:        $Test \leftarrow$ False
17:    end if
18: end while
19: return the DAG G found
```

Algorithm 2. MMPC(T, D)

```
Require: target variable ($T$); Data ($D$)
Ensure: neighborhood of $T$ ($CPC$)
1: $ListC = \mathbf{X} \setminus \{T\}$
2: $CPC = \overline{\text{MMPC}}(T, D, ListC)$
   % Symmetrical correction
3: for all $X \in CPC$ do
4:     if $T \notin \overline{\text{MMPC}}(X, D, \mathbf{X} \setminus \{X\})$ then
5:         $CPC = CPC \setminus \{X\}$
6:     end if
7: end for
```

is generate with the following operators: add_edge (if the edge belongs to the set of constraints and if the resulting is acyclic DAG), delete_edge and invert_edge (if the resulting is acyclic DAG) (this algorithm is not describe for lack of space).

The MMPC local structure identification, described in Algorithm 2, is decomposed into two tasks, the neighborhood identification itself ($\overline{\text{MMPC}}$), completed by a symmetrical AND correction (X belongs to the neighborhood of T if the opposite is also true). The neighborhood identification ($\overline{\text{MMPC}}$), described in Algorithm 3, uses the Max-Min Heuristic defined in Algorithm 4 in order to iteratively add (forward phase) in the candidate Parent-Children set (neighborhood) of a target variable T the variable the most di-

Algorithm 3. $\overline{\text{MMPC}}(T, D, ListC)$

Require: target variable (T); Data (D); List of potential candidates ($ListC$)
Ensure: neighborhood of T (CPC)

```
1:  $CPC = \emptyset$
    % Phase I: Forward
2:  repeat
3:      $<F, assocF>=$ MaxMinHeuristic($T, CPC, ListC$)
4:      if $assocF \neq 0$ then
5:          $CPC = CPC \bigcup \{F\}$
6:          $ListC = ListC \setminus \{F\}$
7:      end if
8:  until $CPC$ has not changed or $assocF = 0$ or $ListC = \emptyset$
    % Phase II: Backward
9:  for all $X \in CPC$ do
10:     if $\exists S \subseteq CPC$ and $assoc(X; T|S) = 0$ then
11:         $CPC \setminus \{X\}$
12:     end if
13: end for
```

Algorithm 4. MaxMinHeuristic($T, CPC, ListC$)

Require: target variable (T); current neighborhood (CPC); List of potential candidates ($ListC$)
Ensure: the candidate the most directly dependent to T given CPC (F) and its association measurement ($AssocF$)

```
1: $assocF = max_{X \in ListC} Min_{S \subseteq CPC} Assoc(X; T|S)$
2: $F = argmax_{X \in ListC} Min_{S \subseteq CPC} Assoc(X; T|S)$
```

rectly dependent on T conditionally to its current neighborhood (line 1 in algorithm 4). This procedure can potentially add some false positives which are then deleted in the backward phase. Dependency is measured with an association measurement function $Assoc$ like χ^2, mutual information or G^2.

3 Dynamic Max-Min Hill-Climbing

3.1 Principle

Local search methods have been proposed to solve the problem of the structure learning in high dimension for static BN. The dimensionality of the search space also increases for DBN, because of the temporel dimension. We think that these methods could be adapted and give relevant results for 2T-BN models.

In 2T-BN models, temporality is constrainted by the first order Markov assumption. We claim that local search algorithms can easily take into account this temporal constraint. We propose this adaptation as a general principle of hybrid structure learning methods (local identification with global search). This paper proposes a new DBN structure learning algorithm inspired from local search methods, by adapting the MMHC algorithm described in the previous section. But an adaptation of other local identification methods is also possible.

Algorithm 5. $\overline{\text{DMMPC}}(T, D)$

Require: target variable (T); Data (D)
Ensure: neighborhood of T in G_0 (Ne_0) and in $G_{\rightarrow}$ (Ne_+)

```
   % search $Ne_0$ of $T$ in $t = 0$
1: $ListC_0 = \mathbf{X}[0] \backslash \{T\} \bigcup \mathbf{X}[1]$
2: $Ne_0 = \overline{\text{MMPC}}(T, D, ListC_0)$
   % search $Ne_+$ of $T$ in $t > 0$
3: $ListC = \mathbf{X}[\text{t-1}] \bigcup \mathbf{X}[\text{t}] \backslash \{T\} \bigcup \mathbf{X}[\text{t+1}]$
4: $Ne_+ = \overline{\text{MMPC}}(T, D, ListC)$
```

Inspired from MMHC algorithm detailed in section 2.2, Dynamic MMHC algorithm proposes to identify independently these graphs by applying a GS algorithm (adapted by [6] for 2T-BN) (cf. Algorithm 3) constrained with local informations. These informations are provided by the identification of the neighborhood Ne_0 (resp. Ne_+) of each node in G_0 (resp. $G_{\rightarrow}$) (cf. Algorithm 5) .

By mimicking the decomposition procedure of MMHC, our local structure identification DMMPC will be decomposed into two tasks: the neighborhood identification itself ($\overline{\text{DMMPC}}$) completed by a symmetrical correction. We notice here that because of the non-symmetry of temporality, our local structure identification will be able to automatically detect some directed parent or children relationships if the corresponding variables do not belong to the same time slice.

3.2 Neighborhood Identification and Symmetrical Correction

$\overline{\text{DMMPC}}$ algorithm consists of two phases detailed in Algorithm 5 respectively dedicated to the identification of the neighborhood Ne_0 (resp. Ne_+) of a target variable T in G_0 (resp. $G_{\rightarrow}$).

$\mathbf{X}[0]$ and $\mathbf{X}[1]$ respectively denote the variables $\mathbf{X}$ for $t = 0$ and $t = 1$. We recall that the 2T-BN model is first-order Markov. Hence, it is possible that the neighborhood Ne_0 of a variable T in $\mathbf{X}[0]$ can belong to $\mathbf{X}[0]$ and $\mathbf{X}[1]$.

Let us define CPC_0 the parents or children of T in slice 0 and CC_1 the children of T in slice 1.

In $\overline{\text{DMMPC}}$, we propose using the static MMPC algorithm with the candidate variables $ListC_0 = \mathbf{X}[0] \cup \mathbf{X}[1] \backslash \{T\}$ in order to identify Ne_0. Because of the temporal information, we will then be able later to separate $Ne_0 = CPC_0 \cup CC_1$.

In the same way, $\mathbf{X}[\text{t-1}]$, $\mathbf{X}[t]$ and $\mathbf{X}[t+1]$ respectively denote the variables $\mathbf{X}$ for times $t-1, t$ and $t+1$. Ne_+ of a variable T in $\mathbf{X}[t]$ can belong to $\mathbf{X}[t-1]$, $\mathbf{X}[t]$ and $\mathbf{X}[t+1]$.

So let us define CPC_t the parents or children of T in slice t, CC_{t+1} the children of T in slice $t+1$ and CP_{t-1} the parents of T in slice $t-1$.

We propose using the static MMPC algorithm with the candidate variables $ListC = \mathbf{X}[t-1] \cup \mathbf{X}[t] \cup \mathbf{X}[t+1] \backslash \{T\}$ in order to identify Ne_+. Because of the temporal information, we will then be able later to separate $Ne_+ = CPC_t \cup CC_{t+1} \cup CP_{t-1}$.

Algorithm 6. DMMPC(T, D)

```
Require: target variable ($T$); Data ($D$)
Ensure: neighborhood of $T$ in $G_0$ ($Ne_0$) and $G_\rightarrow$ ($Ne_+$)
 1: $Ne_0 = \overline{\text{DMMPC}}(T, D, ).Ne_0$ % the set of all neighborhoods of T in $G_0$ returned by $\overline{\text{DMMPC}}$
 2: $Ne_+ = \overline{\text{DMMPC}}(T, D).Ne_+$ % the set of all neighborhoods of T in $G_\rightarrow$ returned by $\overline{\text{DMMPC}}$
    % symmetrical correction $Ne_0$ of $T$ in $t = 0$
 3: $CPC_0 = Ne_0 \bigcap \mathbf{X}[0]$,$CC_1 = Ne_0 \bigcap \mathbf{X}[1]$
 4: for all $X \in CPC_0$ do
 5:     if $T \notin \overline{\text{DMMPC}}(X, D).Ne_0$ then
 6:         $CPC_0 = CPC_0 \setminus \{X\}$
 7:     end if
 8: end for
 9: for all $X \in CC_1$ do
10:     if $T \notin \overline{\text{DMMPC}}(X, D).Ne_+$ then
11:         $CC_1 = CC_1 \setminus \{X\}$
12:     end if
13: end for
14: $Ne_0 = CPC_0 \bigcup CC_1$
    % symmetrical correction $Ne_+$ of $T$ in $t > 0$
15: for all $X \in Ne_+$ do
16:     if $T \notin \overline{\text{DMMPC}}(X, D).Ne_+$ then
17:         $Ne_+ = Ne_+ \setminus \{X\}$
18:     end if
19: end for
20: $CPC = Ne_+ \bigcap \mathbf{X}[t]$ ; $CC = Ne_+ \bigcap \mathbf{X}[t+1]$ ; $CP = Ne_+ \bigcap \mathbf{X}[t-1]$
```

Algorithm 7. DMMHC(D)

```
Require: Data $D$
Ensure: $G_0$ and $G_\rightarrow$
    % Construction initial model $G_0$
 1: for all $X \in \mathbf{X}[0]$ do
 2:     $CPC_X$=DMMPC$(X, D).CPC_0$
 3:     $CC_X$=DMMPC$(X, D).CC_1$
 4: end for
    % Greedy search (GS)
 5: Only try operator add_edge $Y \rightarrow X$ if $Y \in CPC_X$
    % Construction transition model $G_\rightarrow$
 6: for all $X \in \mathbf{X}[t]$ do
 7:     $CPC_X$=DMMPC$(X, D).CPC$ ; $CC_X$=DMMPC$(X, D).CC$ ; $CP_X$=DMMPC$(X, D).CP$
 8: end for
    % Greedy search (GS)
 9: Only try operator add_edge $Y \rightarrow X$ if $Y \in CPC_X$ and $\{X, Y\} \in \mathbf{X}[t]$
10: Only try operator add_edge $X \rightarrow Y$ if $Y \in CC_X$ and $X \in \mathbf{X}[t]$ and $Y \in \mathbf{X}[t+1]$
11: Don't try operator reverse_edge $X \rightarrow Y$ if $Y \in CC_X$ and $X \in \mathbf{X}[t]$ and $Y \in \mathbf{X}[t+1]$
```

As its static counterpart, DMMPC algorithm described in Algorithm 6 has to perform a symmetrical correction. Because of the non-symmetry of temporality, we have to adapt this correction. When $t = 0$, we have to apply separately the symmetrical correction on CPC_0 and CC_1 because for all $X \in CC_1$, X doesn't belong to slice $t = 0$ but to slice $t = 1$ and its temporal neighborhoods are given by $Ne_+(X)$.

3.3 Global Model Optimization

Our Dynamic MMHC algorithm described in Algorithm 7 proposes to identify independently these graphs by applying a greedy search algorithm constrained with local information. These information are provided by the identification of the neighborhood Ne_0 (resp. Ne$_+$) of each node in G_0 (resp. $G_\rightarrow$).

As its static counterpart, DMMHC will consider adding an edge during the greedy search, if and only if the starting node is in the neighborhood of the ending node. G_0 learning only concerns the variables in slice $t = 0$, so we can restrict the add_edge operator only to variables found in a CPC_0 of another variable.

$G_\rightarrow$ is a graph with variables in slices $t-1$ and t but this graph only describes temporal dependencies between $t - 1$ and t and "inner" dependencies in t. So we also restrict our operators in order to consider adding edges with these constraints and we don't authorize reversing temporal edges.

3.4 Time Complexity of the Algorithms

This section presents the time complexity of the algorithm. In the static case and according to the work from Tsamardinaos and al. [14] the number of independence tests for all variables with the target conditioned on all subsets of CPC (target parents and children set) is bound by O($|V|.2^{|CPC|}$), where $|V|$ the number of all variables in the BN and $|CPC|$ is the number of all variables in the parents/children set of target. The overall cost of identifying the skeleton of the BN is O($|V|^2\ 2^{|PC|}$), where PC is the largest set of parents and children overall variables in V.

In our dynamic (temporal) case, as for the static, we first identify the number of tests in the $\overline{DMMPC}$. In this part we have two cases to present (i.e. t=0 and t>0). We start with t=0, $\overline{DMMPC}$ will calculate the association of all variables in time slices t=0 and t=1 with target in slice t=0 conditioned on all subsets of Ne$_0$ (in the worst case). Thus, when t=0, the number of tests is bounded by O($|2V|.2^{|Ne_0|}$), where V is the set of all variables in a time slice. For t>0 case, the number of tests is bounded by O($|3V|.2^{|Ne_+|}$), because $\overline{DMMPC}$ calculates the association of all variables in three time slices t, t-1, t+1 with the target in slice t conditioned on all subsets of Ne$_+$ (in the worst case).

The total number of tests in both phases at $t = 0$ and $t > 0$ is bounded respectively by O($|2V|.2^{|Ne_0|}$) and O($|3V|.2^{|Ne_+|}$). Thus, the total number of tests in both phases is bounded by O($|3V|.2^{|Ne_+|}$). The overall cost of identifying the skeleton of initial and transition models in DBN (i.e., calling DMMPC with all targets in G_0 and $G_\rightarrow$) is O($|3V|^2.2^{|Ne|}$), where Ne is the largest set of neighborhood over all variables in the time slice t.

4 Related Works

Daly and al. [2] propose a recent and interesting state of the art about BN and DBN. We focus here in 2T-BN structure learning. Friedman and al. [6] have shown that this task can be decomposed in two independent phases: learning the initial graph G_0 as a static BN structure with a static dataset corresponding to $\mathbf{X}[t = 0]$ and learning the transition graph $M_{\rightarrow}$ with another "static" dataset corresponding to all the transitions $\mathbf{X}[t] \cup \mathbf{X}[t + 1]$. Then they proposed to apply usual score-based algorithms such as greedy search (GS) in order to find both graphs.

Tucker and al. [17] propose an evolutionary programming framework in order to learn the structure of higher order *k*T-BN. Gao and al. [8] develop another evolutionary approach for 2T-BN structure learning. Wang and al. [20] also look at using evolutionary computation in 2T-BN structure learning by incorporating sampling methods.

All these approaches are validated on benchmark models with about 10 variables. In a more general context, due to inherent limitations of score-based structure learning methods, all these methods will have a very high complexity if the number of variables increases. With the help of local search, DMMHC is able to constraint the search space in the final global optimization (GS). By this way, we can theoritically work in high dimensions like MMHC with static BNs.

Another way to deal with scalability is to restrict the class of 2T-BN by only considering parents of X_i[t+1] in time slice t (for 2T-BN) or any previous time slice (for kT-BN). Dojer [5] proposes a score based method named polynomial time algorithm for learning this class of DBN (with BDe and MDL scores). The implementation of this algorithm is given in [21]. An experimental study have been conducted with microarray data and about 23 000 variables. The time running is about 48 hours with the use of MDL score and 170 hours with the use of BDe score.

Vinh et al. [19] propose another polynomial time algorithm in the same context (equicardinality requirement) with other scoring function (MIT). Vinh et al. [18] propose another score based algorithm without equicardinality assumptions named MIT-global. Also, they propose another contribution in this work consist to an hybrid method with local Blanket identification (MIT-MMMB). An experimental study have been conducted with 1595 variables and show that the local search MIT-MMMB and global-MIT have similar results better than advanced score based algorithm (simulated annealing).

DMMHC and MIT-MMMB are both hybrid methods with local search and global optimization. When DMMHC try to identify the candidate parents/children CPC set of a target variable in a 2T-BN, MIT-MMMB try to identify the (more complex) Markov Blanket MB but in a restricted subclass of 2T-BNs. In one hand, this assumption permits to simplify both MB discovery and global optimization. In other hand, contrary to DMMHC, MMMB is not able to identify the intra-time dependencies.

5 Experimental Study

5.1 Algorithms

We have implemented the Greedy Search (GS) for 2T-BN such as described in section 4. This algorithm is considered as the reference algorithm for DBN structure learning. More complex evolutionary algorithms exist (cf. section 4) but there no available implementation for these specific algorithms.

We have also implemented our proposal, DMMHC, described in section 3. As a first step in our study, our algorithm uses a constrained greedy search, whereas the original

MMHC algorithm proposes using a Tabu search. Extending our approach by using this Tabu search is a simple task and will be one of our immediate perspectives.

We implemented these algorithms in our structure learning platform in C++ using Boost graph[1] and ProBT [2] libraries.

The greedy search used in GS and DMMHC optimizes the BIC score function. DMMPC also uses χ^2 independence test with $\alpha = 0.05$ as *Assoc* function.

Experiments were carried out on a dedicated PC with Intel(R) Core(TM) 2.20 Ghz, 64 bits architecture, 4 Gb RAM memory and under Windows 7.

5.2 Networks and Performance Indicators

Contrary to static BN, evaluating a DBN structure learning algorithm is more difficult. First reason is the unavailability of standard benchmarks, except for instance some reference networks with a small number of variables (less than 10). Second reason is the articles about DBN structure learning use different indicators to argue about the reliability of their proposals.

In [12], we provided tools for benchmarking DBN structure learning algorithms by proposing a 2T-BN generation algorithm, able to generate large and realistic 2T-BNs from existing static BNs. Our companion website[3] proposes some 2T-BNs generated from 6 well-known static BNs (Asia, Alarm, Hailfinder, Win95pts, Andes and Link) with a number of variables in $G_{\rightarrow}$ from 16 to 112 for the 3 first 2T-BNs and from 156 to 1448 for the 3 last ones. For each of these 2T-BNs, we have respectively sampled with Genie/Smile software [4] 2.000, 5.000 and 10.000 sequences of length equal to 6, which correspond to datasets of size 2.000, 5.000 and 10.000 for G_0 structure learning and 5x2.000, 5x5.000 and 5x10.000 for $G_{\rightarrow}$ structure learning. In [12], we also proposed a novel metric for evaluating performance of these structure learning algorithms, by correcting the existing Structural Hamming distance (SHD) in order to take into account temporal background information. As 2T-BNs are defined by two graphs G_0 and $G_{\rightarrow}$, the distance between one theoretical 2T-BN and the learnt one is defined as the pair of the SHD for initial and transition graphs.

Running time is also measured (in seconds). Experiments are canceled when computations did not complete within four days.

5.3 Empirical Results and Interpretations

Figure 1 presents the average results of SHD and running time obtained by GS and DMMHC algorithms with respect to sample size. Figure 1.(a) describes results for initial model corresponding to six (small and large) benchmarks (Asia, Alarm, Hailfinder, Win95pts, Andes, Link). Figure 1.(b) describes results for transition model corresponding to benchmarks used before. We can notice that for every benchmark, DMMHC algorithm obtains better SHD than GS. Also, we can observe than DMMHC overperforms GS running time. This situation is really significant even for benchmarks with a

[1] `http://www.boost.org/`
[2] `http://www.probayes.com/index.php`
[3] `https://sites.google.com/site/dynamicbencmharking/`
[4] `http://genie.sis.pitt.edu/`

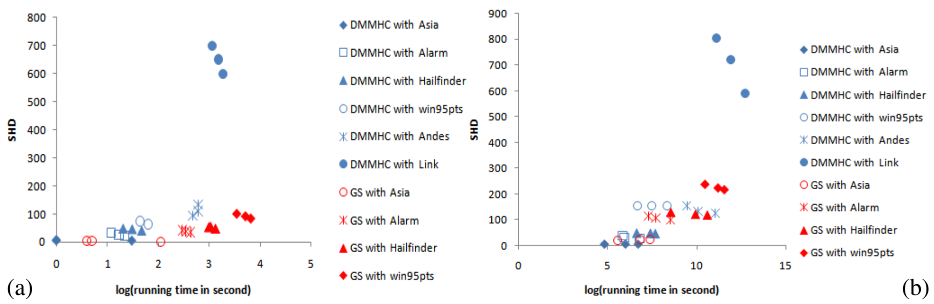

Fig. 1. Average SHD vs running time obtained by GS and DMMHC with respect to sample size. (a) Initial graph (b) Transition graph.

small number of variables. Unlike GS, DMMHC is able to provide results in a decent time for large benchmarks (Andes and Link).

From these results, we can see that DMMHC is an efficient algorithm for 2T-BN structure learning. The quality of the learnt structure is better for our reference algorithm (2T-BN greedy search) with a better scalability: results are obtained in a lower running time and DMMHC can manage high dimensional benchmarks such as 2T-BN generated from Andes and Link.

Comparison with existing works is a difficult task because existing works about 2T-BN structure learning deal with specific benchmarks or specific evaluation metric, such as reported in [12]. As a first comparison, learning the initial model is very similar to static structure learning. As we created our 2T-BN benchmarks from usual static BNs, we can compare the results of our implementations to static ones. [7] provides one comparative study involving GS and MMHC for Alarm, Hailfinder and Link benchmarks with 5000 samples, with BDeu score (and similar results for BIC score). [4] provides SHD results for Alarm and Hailfinder with 5000 samples for another structure learning algorithm, Greedy Thick Thinning (GTT) [3], a two-phases hill-climbing heuristic. Table 1 summarizes the SHD obtained for 3 different hill-climbling algorithms and 2 implementations of MMHC, in about the same contexts (5000 samples). We can observe than our implementation give similar results than concurrent ones for Alarm and Link benchmarks. Some strange results occur for Hailfinder benchmark. [7] reports SHD results equal to 114 (resp. 88) for GS (resp. MMHC) and our implementation obtains 54 (resp. 46) when [4] obtains 48. A deeper study is currently being conducted in order to understand this phenomenon.

It is the first time that existing static benchmarks have been extended for 2T-BN structure learning. It is also the first time that the SHD has been used for 2T-BN, with a temporal correction as described in [12]. For these reasons, comparisons with existing results obtained by concurrent 2T-BN structure learning algorithms are not possible. We intend to disseminate our benchmarks and performance indicators in order to propose a unified evaluation framework for 2T-BN structure learning.

Table 1. SHD comparison with existing static structure learning approaches with 5000 samples

Networks	GS[7]	GTT[4]	GS(G_0)	MMHC[7]	DMMHC(G_0)
Alarm	47	44	40	24	27
Hailfinder	**114**	48	54	**88**	46
Link				687	650

6 Conclusion and Perspectives

We propose in this paper a new 2T-BN structure learning algorithm dealing with realistic networks. Inspired from the MMHC algorithm proposed by Tsamardinos and al. [14] for static BNs, DMMHC algorithm is a local search algorithm dealing with local structure identification (DMMPC) and global model optimization constrained with these local information.

We have shown that the local structure identification can easily take into account the temporal dimension in order to provide some additional information about temporality that can then be used with a greedy search in order to learn the global structure of the initial model G_0 and the transition one $G_{\rightarrow}$. As far as we know, no method able to learn dynamic network models with such approaches has been proposed previously. We also tested DMMHC in high dimensional domains, with thousands of variables.

Our main immediate perspective concern local improvements of our algorithm. We think that our scalability can be improved during the local structure identification by using temporality constraints in DMMPC in order to decrease the number of *Assoc* calls. We also think that the quality of reconstruction can be improved during the global optimization by using a more evolved meta-heuristic such as a Tabu search. Moreover, we will extend our dynamic approach with the use of others local identification methods such as PCD algorithm [10] provably correct under faithfulness assuption. we have also seen that [18] local approch has good properties in a specific subclass of 2T-BNs. our last perspective is to adapt our algorithm in this context and compare it with this state of art scalable algorithms.

Acknowledgment. The first, third and fourth authors would like to acknowledge the financial support provided by grants from the General Department of Scientific Research and Technological Renovation (DGRST), Tunisia, under the ARUB program 01/UR/11/02.

References

1. Chickering, D., Geiger, D., Heckerman, D.: Learning bayesian networks is NP-hard. Tech. Rep. MSR-TR-94-17, Microsoft Research Technical Report (1994)
2. Daly, R., Shen, Q., Aitken, S.: Learning bayesian networks: approaches and issues. Knowledge Engineering Review 26(2), 99–157 (2011)
3. Dash, D., Druzdzel, M.: A robust independence test for constraint-based learning of causal structure. In: Proceedings of the Nineteenth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI 2003), pp. 167–174. Morgan Kaufmann (2003)

4. de Jongh, M., Druzdzel, M.: A comparison of structural distance measures for causal bayesian network models. In: Recent Advances in Intelligent Information Systems, Challenging Problems of Science. Computer Science series, pp. 443–456 (2009)
5. Dojer, N.: Learning bayesian networks does not have to be np-hard. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 305–314. Springer, Heidelberg (2006)
6. Friedman, N., Murphy, K., Russell, S.: Learning the structure of dynamic probabilistic networks. In: UAI 1998, pp. 139–147 (1998)
7. Gámez, J.A., Mateo, J.L., Puerta, J.M.: Learning bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. Data Mining and Knowledge Discovery 22, 106–148 (2011)
8. Gao, S., Xiao, Q., Pan, Q., Li, Q.: Learning dynamic bayesian networks structure based on bayesian optimization algorithm. In: Liu, D., Fei, S., Hou, Z., Zhang, H., Sun, C. (eds.) ISNN 2007, Part II. LNCS, vol. 4492, pp. 424–431. Springer, Heidelberg (2007)
9. Murphy, K.: Dynamic Bayesian Networks: Representation, Inference and Learning. Phd, University of California, Berkeley (2002)
10. Peña, J.M., Björkegren, J., Tegnér, J.: Scalable, efficient and correct learning of markov boundaries under the faithfulness assumption. In: Godo, L. (ed.) ECSQARU 2005. LNCS (LNAI), vol. 3571, pp. 136–147. Springer, Heidelberg (2005)
11. Rodrigues de Morais, S., Aussem, A.: A novel scalable and data efficient feature subset selection algorithm. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part II. LNCS (LNAI), vol. 5212, pp. 298–312. Springer, Heidelberg (2008)
12. Trabelsi, G., Leray, P., Ben Ayeb, M., Alimi, A.: Benchmarking dynamic bayesian network structure learning algorithms. In: ICMSAO 2013, Hammamet, Tunisia, pp. 1–6 (2013)
13. Tsamardinos, I., Aliferis, C., Statnikov, A.: Time and sample efficient discovery of markov blankets and direct causal relations. In: ACM-KDD 2003, vol. 36(4), pp. 673–678 (2003)
14. Tsamardinos, I., Brown, L.E., Constantin, F., Aliferis, C.F.: The max-min hill-climbing bayesian network structure learning algorithm. Mach. Learn. 65(1), 31–78 (2006)
15. Tsamardinos, I., Constantin, F.A., Statnikov, A.: Algorithms for Large Scale Markov Blanket Discovery. In: FLAIRS, pp. 376–380 (2003)
16. Tucker, A., Liu, X.: Extending evolutionary programming methods to the learning of dynamic bayesian networks. In: The Genetic and Evolutionary Computation Conference, pp. 923–929 (1999)
17. Tucker, A., Liu, X., Ogden-Swift, A.: Evolutionary learning of dynamic probabilistic models with large time lags. International Journal of Intelligent Systems 16(5), 621–646 (2001)
18. Vinh, N., Chetty, M., Coppel, R., Wangikar, P.: Local and global algorithms for learning dynamic bayesian networks. In: ICDM 2012, pp. 685–694 (2012)
19. Vinh, N., Chetty, M., Coppel, R., Wangikar, P.: Polynomial time algorithm for learning globally optimal dynamic bayesian network. In: Lu, B.-L., Zhang, L., Kwok, J. (eds.) ICONIP 2011, Part III. LNCS, vol. 7064, pp. 719–729. Springer, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-24965-5_81
20. Wang, H., Yu, K., Yao, H.: Learning dynamic bayesian networks using evolutionary mcmc. In: ICCIS 2006, pp. 45–50 (2006)
21. Wilczynski, B., Dojer, N.: Bnfinder: exact and efficient method for learning bayesian networks. Bioinformatics 25(2), 286–287 (2009)