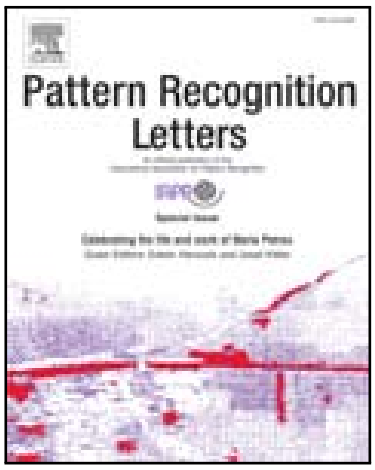

Predictive complex event processing based on evolving Bayesian networks

Yongheng Wang , Hui Gao , Guidan Chen

PII: S0167-8655(17)30148-4
DOI: 10.1016/j.patrec.2017.05.008
Reference: PATREC 6814

To appear in: *Pattern Recognition Letters*

Received date: 21 January 2017
Revised date: 3 April 2017

Please cite this article as: Yongheng Wang , Hui Gao , Guidan Chen , Predictive complex event processing based on evolving Bayesian networks, *Pattern Recognition Letters* (2017), doi: 10.1016/j.patrec.2017.05.008

Highlights

- Bayesian network model uses Gaussian mixture model and EM algorithm for approximate inference.
- Incremental calculation of score metric for updating Bayesian network structure.
- Evolving Bayesian network structure using hill-climbing algorithm
- Hill-climbing algorithms that explores multiple peaks parallel.
- Updating score metrics based on changed edges only.

Predictive complex event processing based on evolving Bayesian networks

Yongheng Wang ∗, Hui Gao and Guidan Chen

College of Computer Science and Electronic Engineering, Hunan University, Changsha 410082, China

ABSTRACT

In the Big Data era, large volumes of data are continuously and rapidly generated from sensor networks, social network, the Internet, etc. Predicting from online event stream is an important task since users usually need to predict some future states and take some actions in advance. Many applications need online prediction models which can evolve automatically with data distribution drift and algorithms which can support single-pass processing of data, which are still faced with many challenges. In this paper, the authors propose a predictive complex event processing method based on evolving Bayesian networks. The Bayesian model is designed based on event type and time with inference method based on Gaussian mixture model and EM algorithm. When learning the structure of Bayesian network from event streams, this method supports calculating score metric incrementally when new data is arrived or edges in the network are changed. Evolving Bayesian network structure is supported based on hill-climbing method. The system can continuously monitor the Bayesian network model and modify it if it is found to be not appropriate for the new incoming data. The method of this paper is evaluated in road traffic domain with both real application data and data produced by a simulated transportation system. The total percentage error is 8.12% for real data and 7.78% for simulated data, while the best result for other methods is 11.79% for real data and 14.59% for simulated data. The experimental evaluations show that this method is effective for predictive complex event processing and it outperforms other popular methods when processing traffic prediction in intelligent transportation systems.

Keywords: Event Stream; Predictive Complex Event Processing; Evolving Bayesian Networks

1. Introduction

In the Big Data era users require new technology to process data stream with high speed and variety of data type. Besides the traditional stream processing technology, it is important to catch the relations inside online streaming data. Most of the data streams are composed of primitive events that produced by sensor networks, internet, social networks, etc. The semantic information inside primitive events is quite limited. In real application, people usually pay more attention to higher level information such as business logic and rules. For example, enormous events are generated in a trading system, but a fraud detecting system only care about the events that can cause frauds. Complex Event Processing (CEP) [1] is the technology that interprets and combines streams of primitive events to identify higher level composite events. CEP has been used in many areas, such as sensor networks for environmental monitoring, continuous analyzing of stocks to detect trends, etc.

Predictive Complex Event Processing is the technology to identify events before they have happened, so that they can be eliminated or their effects mitigated [2]. For example, a financial institution wishes to detect frauds or a financial regulator wishes to catch illegal trading patterns in advance. Another example is to predict the traffic status in road networks and take some actions proactively to mitigate or eliminate undesired future states. Simple predictive CEP can be supported by rule-based method which means users define some patterns and the system then continuously monitor the event streams to predict future events. However, for complex cases it is not easy to define predictive CEP patterns exactly. In such circumstance, predictive analytics technology can be applied to support predictive CEP. Predictive analytics applies several statistical and data mining techniques such as clustering, classification, regression and so on. Recently, neural networks (especially deep neural networks) and Bayesian Networks (BN) are commonly used as prediction models. BN and it variations, such as Dynamic Bayesian Networks (DBN) [3], Adaptive Bayesian Networks (ABN) [4], etc., are widely used in predictive analytics because they have the following advantages: (i) they allow to express directly the fundamental qualitative relationship of direct causation, (ii) there exists mathematical methods to estimate the state of certain variables given the state of other variables, (iii) there are methods in order to explain to the user how the system came to its conclusions [5].

Currently predictive CEP with predictive analytics has some challenges. First, traditional predictive analytics methods are designed for database which means they assume all data is available at any time. However, in predictive CEP the system can only process data on single-pass and cannot control over the order of samples that arrive over time. Stream-based predictive analytics methods are needed which is more difficult. Second, the distribution of data can change over time. A model learned from historical data may not fit the new coming data well. This means real time modeling and learning algorithms are needed. In order to support predictive CEP with BN, an Evolving Bayesian Networks (EBN) model is needed that can adjust itself automatically when the distribution of data changes. There are some works on learning BN model incrementally but they simply

∗ Corresponding author. Tel.: +86-13975813465; e-mail: wyh@hnu.edu.cn

assume all data is in memory and add new incoming data into existing data in each step. Finally, event streams often have high incoming rate, especially the event streams produced by wireless sensor networks. Predictive CEP is usually used to support online decision support system which need high performance even for large scale distributed event streams.

When used to handle the predictive CEP issue, the main weakness of the current predictive analytics methods is that they cannot get acceptable result due to the distribution drift of streaming event data. To address the challenges and overcome the weakness of current work, in this paper the authors propose a Predictive Complex Event Processing method based on Evolving Bayesian Networks (PCEP-EBN). The main contribution of this paper includes the following:

— The authors propose a predictive CEP framework based on BN model which uses Gaussian mixture model and EM algorithm for approximate inference.

— When learning and updating the BN structure, the authors propose incremental calculation methods for the score metric which support single-pass processing of event stream.

— The authors propose algorithms to support evolving BN structure based on hill-climbing method. The hill-climbing algorithm of this paper supports exploring multiple peaks on parallel and updating the score metric based on changed edges only. Evolving BN parameters is also supported using an incremental EM algorithm for Gaussian mixture model.

The authors evaluated the PCEP-EBN method in road traffic domain with both real application data and data produced by a simulated transportation system. The results show that PCEP-EBN is an effective method for predictive complex event processing and it outperforms other popular methods when processing traffic prediction in intelligent transportation systems.

2. Related work

2.1. Complex event processing and predictive analytics

CEP is a technique that recognizes complex events based on a set/sequence of occurrences of single events by continuously monitoring the event stream. Etzion et al. defined the basic concepts and architecture of complex event processing in their book [2]. The main component in CEP is Event Processing Agent (EPA) which can generate a set of complex events as output from a set of input events by applying some logic. Event Processing Network (EPN) is composed of a collection of EPAs, event producers, and event consumers linked by channels. EPN is used to define the event processing flow execution. Luckman introduced the event processing network first in the field of modeling [1]. Based on this idea, the conceptual model of EPN was further elaborated by Sharon and Etzion [6]. The key ideas of CEP have four steps: (1) Primitive events are extracted from large volume data stream; (2) Event correlation or event aggregation is detected to create business logic with event operators according to specific rules; (3) Primitive or composite events are processed to extract their time, causal, hierarchical and other semantic relationships; (4) High level complex events are sent to the subscribers which can be used for decision making.

CEP engine needs to process streams of events with time stamp and, therefore, numerous event pattern recognition methods are proposed based on Nondeterministic Finite Automaton (NFA). SASE [7] is a high performance query-plan based complex event processing method that uses NFA. Agrawal et al. proposed an improved pattern matching model and a query evaluation framework based on NFA to support more powerful event processing [8]. In their work a suite of techniques are proposed to improve runtime efficiency by exploiting sharing in storage and processing.

The basic CEP method detects event patterns from online event stream. Another type of CEP is predictive CEP which can predict future events before they have happened [2]. Etzion et al. believe predictive CEP and intelligent CEP are main parts of future directions in event processing technology. Currently most of the work on predictive CEP considers the task as a combine of CEP and predictive Analytics. Fülöp et al. proposed a conceptual framework of predictive CEP by combining CEP and predictive analytics [9]. In their framework, prediction models are trained using historical events. Online complex events produced by EPN are sent to predictive analytics component to predict future events. The predicted future events are then sent back to EPN for further step process. Tóth et al. proposed a framework that includes the predictive analytics technologies into CEP applications [10]. They extended a CEP solution with predictive capabilities, defined the key aspects of the combination of these techniques, and summarized how CEP and PA could gain from the joint solution.

Predictive analytics has been studied for many years and a series of models and algorithms are proposed. Some work [35,36] used optimized values/methods to improve the results which is related to this paper. Recently, deep neural networks and BN are widely used and be more suitable for predictive CEP. Huang et al. proposed a predictive analytics method based on deep belief networks [11]. Their method uses deep belief networks in the bottom level and a regression model is used on the top level to support the final prediction. Multitask learning is supported in the top regression model to improve performance. In the work of Lv et al., a stacked autoencoder model is used to learn generic event stream features, and it is trained in a greedy layerwise fashion [12]. Predictive analytics with deep learning model usually can achieve good accuracy, but the training of the model is complex and it prone to the problem of over fitting.

As a valid model for uncertain knowledge representation and inference, Bayesian Network is widely used in predictive analytics. Castillo et al. proposed a Bayesian network model for traffic flow prediction which is able to take into account the random character of the level of total mean flow and the variability of origin-destinations (OD) pair flows, together with the random violation of the balance equations for OD pairs [13]. This model provides the joint density of all unobserved variables and in particular the corresponding conditional and marginal densities, which allow not only joint predictions, but also probability intervals. Zhu et al. proposed a Bayesian network model for traffic prediction using prior link flows [14]. Their model sets link flows as parents of the OD flows. Using prior link flows, the prior distribution of all the variables is determined. To remove inconsistencies in OD matrices estimation and traffic assignment, a combined BN and stochastic user equilibrium model is proposed, in which the equilibrium solution is obtained through iterations.

2.2. Evolving systems and evolving Bayesian networks

Evolving systems are developed to address the distribution drifts in data stream. Angelov et al. proposed an evolving intelligent system which supports learning and adjusting model from data stream [30]. Recently many incremental machine learning methods [31,32] are proposed which are suitable for learning from data stream with distribution drifts. Another related work is evolving neural network which uses evolutionary algorithms to optimize the structure and parameters of neural networks [33,34].

When using BN, predictive analytics for event stream can be looked upon as the learning of BN with respect to these inherent characteristics of the sample data. Learning BN from data is the key technique for modeling uncertain knowledge, the basis of uncertainty inferences, and corresponding applications. Learning the BN structure from data is an NP-hard problem as the number of possible structures grows super-exponentially by the number of nodes. Approximate structure learning algorithms are usually categorized into two groups of constraint-based and search-and-score (S&S) approaches. Constraint-based algorithms rely on a number of conditional independency tests to determine whether two variables are independent given a set of conditioned variables. Constraint-based approaches are generally asymptotically correct and have a well-defined stopping criterion [15]. Being a more popular method, S&S approach involves a strategy for searching through the space of possible structures and a scoring function measuring the fitness of each structure to the data. The commonly used score metrics include Minimum Description Length (MDL) [16], Bayesian Information Criterion (BIC) [17], Bayesian Dirichlet equivalent (BDe) [18], etc. In order to address the problem of huge search space, most of the S&S methods use heuristic-based optimization or approximate searching algorithms. Some recently examples include hill climbing [19], simulated annealing [20] and Genetic algorithms [15].

The issue of distribution drift reflects a situation in which the event stream does not follow a fixed and predictable data distribution. In order to address the distribution drift in event stream, EBN is needed. EBN offers a flexible working framework which adapts to any variation in data trends or expansion of system states by incremental structure and parameter adaption. For abrupt drifts, a common idea for EBN is to learn different BN models and switch among models when data distribution is changed. Pascale et al. proposed an adaptive BN model for traffic flow prediction [4]. In their work, two major stationary areas are recognized as principal phases of traffic flows. BN models are learned for each phase and the adaptive method select BN model based on mutual information learning.

Incremental drifts are more difficult to be handled, because it cannot be detected by global drift detection approaches in the structural learning scenario and cannot be addressed by the parameter learning scenario either. These drifts gradually interfere with current data distribution, which undermines the predictive accuracy more severely. Some researchers use sampling-based method for EBN. For example, Robinson et al. proposed a Markov chain Monte Carlo (MCMC) sampling algorithm for learning the structure of the model from time-series data under different assumptions [21]. Sampling methods allow the probability or importance of individual edges to be evaluated over all possible networks. However, since only the best network structure is needed in predictive CEP, heuristic search-based methods will typically find them more quickly. Yasin et al. proposed an incremental algorithm for BN structure learning which can deal with high dimensional domains [22]. Their algorithm learns local models by limiting search space and performs a constrained greedy hill-climbing search to obtain a global model. Yue et al. proposed a parallel and incremental approach for data-intensive learning of BNs from massive, distributed, and dynamically changing data by extending the classical scoring and search algorithm and using MapReduce [23]. They used MDL scoring metric and proposed a two-pass MapReduce-based algorithms for computing the required marginal probabilities and scoring the candidate graphical model from sample data. An extended hill-climbing algorithm is then used to obtain the optimal structure. Li et al. proposed a framework of incremental BN structure learning and a new evaluation criterion "ABIC" (Adopt Bayesian Information Criterion) based on the BIC metric [24]. Then a three phase algorithm is used to learn the BN structure. Compared to this paper, their methods are not designed for data stream and cannot process data in single-pass manner. Acharya et al. studied the issue of modeling causal relationships over event streams where data are unbounded. Their method infers causality by learning the temporal precedence relationships using an incremental temporal network construction algorithm called Incremental Hill-Climbing Monte Carlo [25]. Compared to this paper, the work of Acharya et al. did not considered how to calculate the score metric incrementally. Furthermore, their incremental hill-climbing algorithm that starts from only the highest local peak is prone to miss best result near other local peaks.

3. Predictive CEP with Bayesian Networks

3.1. System architecture and basic CEP

The system architecture is shown in Fig. 1. Primitive event streams generated by various event sources are processed by CEP engine. The BEPA means basic event processing agent which recognize complex event based on patterns from event stream. Multiple EPAs are connected by event channels (ECs) to create an EPN. The skeleton in PEPA is a statistical summary of the dataset which contains all information necessary to calculate the scores of the Bayesian network. The predictive complex events generated by PEPA are sent to the event consumer, which is usually decision support system, to make some decisions. Based on the decisions some actions can be executed to affect the event sources proactively to eliminate some unwanted states. The decision support system is beyond the scope of this paper.

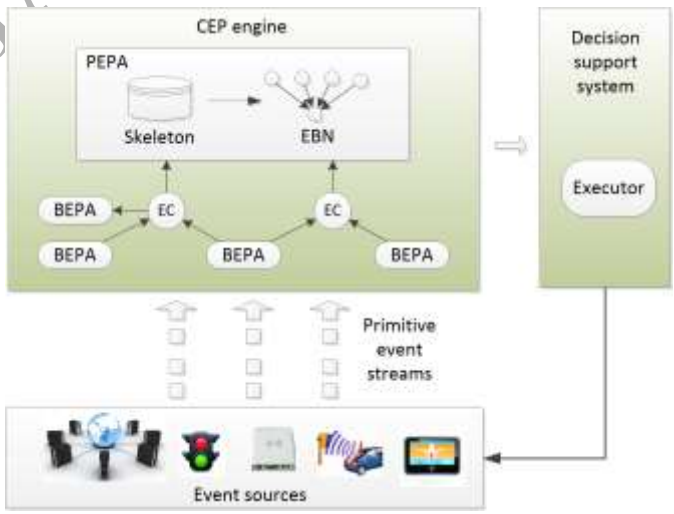

Fig. 1. The system architecture.

Definition 1 (primitive event). A primitive event is an atomic occurrence of interest in time. A primitive event is represented as $\langle A,T \rangle$ where A is the set of attributes and T is the timestamp that the event occurs.

Definition 2 (complex event). A complex event is a combination of primitive events or complex events by some rule. A complex event is represented as $\langle E,R,T_s \rangle$ where E represents the elements that compose the complex event, R represents the rule of the combination, T_s represents the time span of the complex event.

For example, each read operation of RFID reader generates a primitive event. Event type is a specification for a set of events that have the same semantic intent and the same structure. An

event context is a named specification of conditions that groups event instances so that they can be processed in a related way. The main complex event patterns include ALL, ANY, COUNT, SEQ, etc. The detailed information about the concepts and patterns of CEP is described in [2]. The authors have developed a high-performance Distributed Probabilistic Complex Event Processing method (DPCEP) [26]. Based on NFA, parallel algorithm is designed to improve the performance for both single and distributed streams. A query plan based method is used to process hierarchical complex event from distributed event streams. Query plan optimization is proposed based on query optimization technology of probabilistic databases.

3.2. Bayesian network model for predictive CEP

Definition 3 (predictive complex event). A predictive complex event is a future event that can be predicted from historical event stream. A predictive complex event is represented as e_{t+1}=h(e_t, e_{t-1},…; Θ) where t is time stamp, h() is the prediction function and Θ is the set of parameters.

Definition 4 (Bayesian network, BN). A Bayesian network is represented as $B=\langle G,\Theta\rangle$ where G is a directed acyclic graph (DAG) that represent the BN structure and Θ is the set of parameters. The DAG can be represented by $G=\langle V,E\rangle$ where V is the set of random variables which makes up the nodes of the network, and E is the set of directed edges connecting pairs of nodes.

An arrow from node X to node Y means that X has a direct influence on Y (X, $Y \in V$ and $X \neq Y$). The parameters represent the probabilities of each random variable given its set of parents: $\theta_i = P(x_i|X_{pa(i)})$, where x_i is a random variable and $X_{pa(i)}$ is the set of parents of x_i.

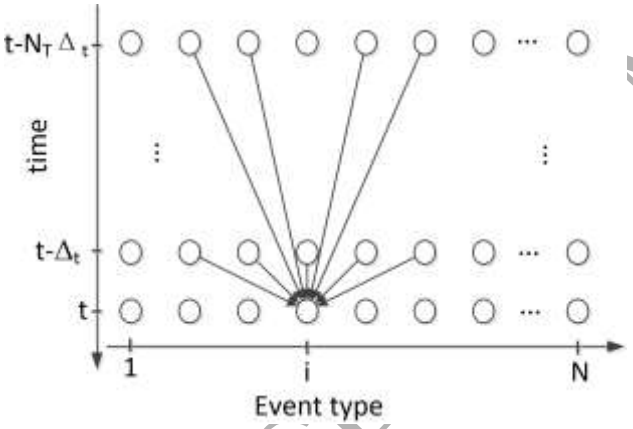

Fig. 2. The Bayesian network structure. Only the links related to node (i,t) are displayed. The horizontal coordinate is abstract and it can be instantiated as attributes of objects, states of the system, etc.

The BN model in this paper is shown in fig. 2 which has two dimensions: event type and time. The event type is related to some attributes of objects, e.g., the position of objects, or some states of the system, e.g., the traffic flow of roads. Assume the event type number is N, in a prediction task the state of node (i,t) is related to a set of states before time t (parent nodes). Let $x_{i,t}$ represent the state variable of node (i,t) and pa(i,t) represent the set of parent nodes of (i,t). N_P denotes the number of nodes in pa(i,t). $N_T\Delta_t$ is the time window which means only nodes from time t-$N_T\Delta_t$ to t-Δ_t are considered as the parents of nodes in time t. The set of state variables for pa(i,t) is $X_{pa(i,t)}=\{x_{j,s} \mid (j,s) \in pa(i,t)\}$. According to the BN theory, the joint distribution of all nodes in the network can be represented as:

$$p(X)=\prod_{i,t} p(x_{i,t}|X_{pa(i,t)}) \quad (1)$$

The conditional probability $p(x_{i,t}|X_{pa(i,t)})$ can be calculates as:

$$p(x_{i,t}|X_{pa(i,t)})=p(x_{i,t},X_{pa(i,t)})/X_{pa(i,t)} \quad (2)$$

Since it is not easy to calculate the joint distribution $p(x_{i,t},X_{pa(i,t)})$, the authors use Gaussian Mixture Model (GMM) [27] to approximate it like the following:

$$p(x_{i,t},\ X_{pa(i,t)}) \ = \ \sum_{m=1}^{M} \pi_m g_m(x_{i,t},X_{pa(i,t)}|\mu_m,\Sigma_m) \quad (3)$$

where M is the number of Gaussian models and $g_m(\cdot|\mu_m,\Sigma_m)$ is the m-th Gaussian distribution with $(N_P + 1) \times 1$ vector of mean values μ_m and $(N_P + 1) \times (N_P + 1)$ covariance matrix Σ_m. Using EM algorithm, the parameters $\{\pi_m,\mu_m,\Sigma_m\}_{m=1}^{M}$ can be inferred from historical data [27]. Then the conditional distribution $p(x_{i,t}|\ X_{pa(i,t)})$ can be derived from $p(x_{i,t},\ X_{pa(i,t)})$ and the prediction $\hat{x}_{i,t}$ can be calculated from $X_{pa(i,t)}$ with minimum mean square error (MMSE) method.

4. Evolving Bayesian Network

4.1. Bayesian network structure learning

Definition 5 (evolving Bayesian network, EBN). An evolving Bayesian network is a Bayesian network whose structure and parameters are changed according to the drift of data distribution. An EBN is represented by $\langle G(t),\Theta(t)\rangle$ where $G(t)$ and $\Theta(t)$ are the structure and parameters at time t respectively.

The EBN includes evolving of BN structure and parameters. For structure evolving, since search-and-score method is more suitable for evolving than constraint-based method, the authors use a search-and-score method with BDe metric [18] score function. The BDe metric is defined as following:

$$P(D|G)=\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})}\prod_{k=1}^{r_i}\frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})} \quad (4)$$

Where D is data set, q_i is the number of configurations of the parent set pa_i, r_i is the number of discrete states of variable x_i, N_{ijk} is the number of times x_i took on the value k given the parent configuration j, $N_{ij}=\sum_{k=1}^{r_i} N_{ijk}$, α_{ij} and α_{ijk} are Dirichlet hyper-parameters on various entries in the BN parameters, and Γ(v) is the Gamma function which for $v \in \mathbb{N}$ is given by Γ(v) = (v-1)!. In the BDe metric, α_{ijk} is set to $\alpha/(q_i r_i)$ which means parameter priors are all controlled by a single hyper-parameter α.

BN structure learning is the task of selecting a BN structure G that explains a given data set D. It has been shown that searching the huge space of DAGs for the optimal structure is a NP-hard problem. The BN model designed in this paper shown in Fig. 2 is a constrained DAG. When a directed edge from node (j,t_1) to node (i,t_2) is drawn, it must satisfies $t_1 < t_2$. Even the maximum value of the $\Delta t = t_2 - t_1$ is set, it is still a NP-hard problem. Therefore greedy local search method is used in this paper.

In order to calculate the BDe metric incrementally, an algorithm is proposed as shown in algorithm 1. In this algorithm the BN structure is assumed to be static. The idea behind this algorithm is that, when new data comes the metric is not needed to be calculated from scratch but "replace" the part that affected by new data, as shown in fig. 3.

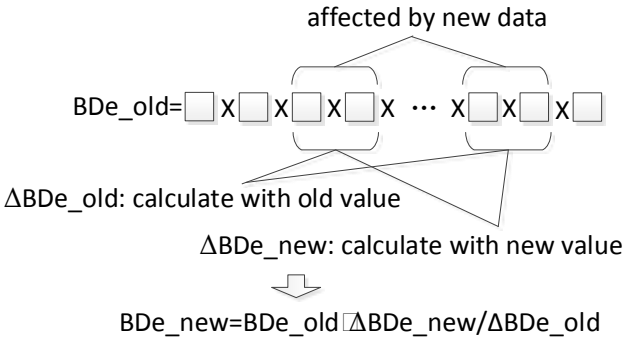

Fig. 3. The idea of calculating BDe metric incrementally (algorithm 1).

From equation (4) it can be found that the BDe metric is the multiplication of many factors. Algorithm 1 tries to find the factors that affected by the new data and replace the old values of these factors with new values. The sufficient statistics in the skeleton, including the values of N_{ijk}, N_{ij}, etc., are stored in memory and are updated when new data is inputted. In lines 2 – 5 of algorithm 1, if the proportion of edges affected by new data accounted for the total edge number is larger than a threshold value θ, the BDe metric can be calculated directly without using the incremental method. The reason is that in such circumstance a large part of factors in the BDe metric need to be replaced. For each edge (i,j) that affected by new data, the new value of N_{ij} is calculated (lines 10 – 12). Then the old value and new value of the factors affected by new data are calculated (lines 13 – 19). Finally the old value is replaced by new value (line 21). From equation (4) and algorithm 1, the authors concluded that if *p* percent of the BN structure is affected by new data, the running time of algorithm 1 is almost *p* percent of the running time of calculating equation (4) directly.

Algorithm 1: Incremental calculation of BDe metric with new data (BN structure is not changed)

Input: Bayesian network structure G, old data set D, old metric BDe_old, new data get ΔD
Output: new metric BDe_new

1 Δedge ←{e | edge e is affected by ΔD}
2 **if** |Δedge|/EdgeNumber(G) > θ) **then**
3 | calculate BDe_new directly from skeleton
4 | **Return** BDe_new
5 **end if**
6 Δp ← {i,j,k | N_{ijk} is changed by ΔD}
7 ΔBDe_old ← 1, ΔBDe_new ← 1
8 for each pair (i,j) in Δp
9 | $N_{ij_}$new ←N_{ij}
10 | **for each** k in (i,j,k) triple of Δp
11 | | $N_{ij_}$new ←$N_{ij_}$new + N_{ijk_new}
12 | **end for**
13 | **for each** k in (i,j,k) triple of Δp
14 | | $\Delta BDe\_old \leftarrow \Delta BDe\_old \cdot \Gamma(\alpha_{ijk}+N_{ijk})/\Gamma(\alpha_{ijk})$
15 | | $\Delta BDe\_new \leftarrow \Delta BDe\_new \cdot \Gamma(\alpha_{ijk}+N_{ijk\_new})/\Gamma(\alpha_{ijk})$
16 | **end for**
17 | $\Delta BDe\_old \leftarrow \Delta BDe\_old \cdot \Gamma(\alpha_{ij})/\Gamma(\alpha_{ij}+N_{ij})$
18 | $\Delta BDe\_new \leftarrow \Delta BDe\_new \cdot \Gamma(\alpha_{ij})/\Gamma(\alpha_{ij}+N_{ij\_new})$
19 | update N_{ij} with N_{ij_new}
20 **end for**
21 BDe_new ← BDe_old·ΔBDe_new/ΔBDe_old
22 **return** BDe_new

The hill-climbing search algorithm (HCS) works in a step-by-step fashion to generate a model. At each step it makes the maximum possible improvement in an objective quality function. In this paper the objective quality function is the BDe metric. It is also needed to define a set of traverse operators OP = {op^1 ,…, op^k} that can be executed at each step. Based on the property of the BN model in fig. 2, the authors define a simple but effective one edge neighborhood (OEN).

Definition 6 (one edge neighborhood, OEN). An one edge neighborhood of a given model *M* is the set of all the alternative models that can be built from *M* through adding or removing one edge, OEN(*M*)={*M'*|*M'*=op(*M*) ∧ op∈{add an edge, remove an edge}}.

In algorithm 1 the BN structure is assumed to be static. In order to calculate the BDe metric incrementally with new edge when data is not changed, the authors use the following equation:

$$\Delta BDe(u,v) = \frac{\Gamma(\alpha_{vu})}{\Gamma(\alpha_{vu}+N_{vu})}\prod_{k=1}^{r_v}\frac{\Gamma(\alpha_{vuk}+N_{vuk})}{\Gamma(\alpha_{vuk})} \quad (5a)$$

$$BDe_{new} = BDe_{old} \cdot \Delta BDe(u,v), \text{ if edge } (u,v) \text{ is added} \quad (5b)$$

$$BDe_{new} = BDe_{old} / \Delta BDe(u,v), \text{ if edge } (u,v) \text{ is removed} \quad (5c)$$

where (u,v) is the edge that changed.

Now the problem is how to generate a BN structure from data D without considering new incoming data. The authors modified the max-min hill-climbing (MMHC) algorithm [28] as shown in algorithm 2. A list (localMaxList) is used to keep the structures that maximum the score function locally. The algorithm repeats the step of changing an edge and finding the structure that maximum the score function. After MAXTRIALS changes without any increase in score function, the algorithm will be terminated. The graph is set to be edgeless at the beginning. In each step, the OEN of the current graph is generated by adding an edge randomly, and a structure G_{max} is selected if it is not in localMaxList and it maximums the BDe metric (line 4). The calculation of BDe metric when adding an edge is described in equation (5a-5c).

Algorithm 2: MMHC

input: data set D, a constant MAXTRIALS
output: BN structure G

1 G_{final} ← empty graph, G ← edgeless graph
2 localMaxList ← empty list
3 **repeat**
4 | choose G_{max} from OEN(G) where G_{max} is not in localMaxList and it maximums the BDe metric
5 | add G_{max} into localMaxList
6 | G ← G_{max}
7 | ΔS ← BDe(G,D) - BDe(G_{final},D)
8 | **if** (ΔS > 0) **then**
9 | | G_{final} ← G
10 | **end if**
11 **until** G_{final} has not changed last MAXTRIALS times
12 return G_{final}

4.2. Evolve Bayesian network structure

The authors use an evolving MMHC algorithm (EMMHC) to evolve BN structure as shown in algorithm 3. The current BN structure G_{cur} has been learnt by the MMHC algorithm. Existing methods usually merge the new data into existing data which makes the system has bad scalability. The BDe value of the BN structure is updated using equation 5, and the skeleton is also updated. The EMMHC algorithm tries to find the optimized BN structure give an incremental data set ΔD. If the change of score when applying ΔD is smaller than a threshold value δ, then the current BN structure need not be changed (lines 4 – 6). The BDe(G_{new}, ΔD) is calculated using algorithm 1. The EMMHC

algorithm creates k subtasks to incrementally search for optimized BN structure starting from the structures with top-k score in the localMaxList (lines 7 – 9). The subtasks are executed parallel and the result with highest score is selected.

Algorithm 3: EMMHC

input: current BN structure G_{cur}, statistic information in skeleton I_{stat}, incremental data set ΔD, localMaxList
output: new BN structure G_{new}
1 currentScore ←BDe(G_{cur})
2 $G_{new} \leftarrow G_{cur}$
3 **if** (|BDe(G_{new}, ΔD) - currentScore |/currentScore < δ) **then**
4 return G_{new}
5 **end if**
6 **for each** G in top-k of localMaxList
7 create a new subtask by calling LocalSearch with parameters G, I_{stat}, ΔD, currentScore
8 **end for**
9 run all sub tasks parallel and wait the result from all sub tasks
10 $G_{max} \leftarrow$ the BN structure with the highest score in the sub tasks
11 **if** (BDe(G_{max}) > BDe(G_{new})) **then**
12 $G_{new} \leftarrow G_{max}$
13 **end if**
14 **return** G_{new}

The subtask LocalSearch is shown in algorithm 4. The idea behind this algorithm is that, since only part of the graph is affected by the new data, the algorithm only need to add or remove edges inside the affected part to maximum the score. This algorithm has two stages: expansion stage and contraction stage. During the expansion stage, an edge is repeatedly added that can maximum the score function until the score cannot be increased. During the contraction stage, an edge is repeatedly removed that can maximum the score function until the score begin to decrease. In line 2, the getCandidateEdgesToAdd function gets the node pairs that in affectedNodes but have no edge between them (edge can be added according to the structure of fig. 2). In line 3, the getCandidateEdgesToRemove function gets the edges that are in current BN structure and affected by incremental data set ΔD.

Algorithm 4: LocalSearch

input: current BN structure G_{cur}, statistic information in skeleton I_{stat}, incremental data set ΔD, current score value currentScore
output: new BN structure G_{new}
1 affectedNodes ← getAffectedNodes(ΔD)
2 candidateEdgesToAdd ← getCandidateEdgesToAdd (affectedNodes, G_{cur})
3 candidateEdgesToRemove ← getCandidateEdgesToRemove (affectedNodes, G_{cur})
4 expansionStage, contractionStage ← true
5 **while**(expansionStage)
6 | maxScore ← the max score of the BN by adding an edge in candidateEdgesToAdd
7 | **if** (maxScore > currentScore)
8 | $G_{new} \leftarrow$ the BN structure that get max score
9 | **else**
10 | expansionStage ← false
11 | **end if**
12 **end while**
13 **while**(contractionStage)
14 | maxScore ← the max score of the BN by removinging an edge in candidateEdgesToRemove
15 | **if** maxScore !< currentScore **then**
16 | $G_{new} \leftarrow$ the BN structure that get max score
17 | **else**
18 | contractionStage ← false
19 | **end if**
20 **end while**
21 **return** G_{new}

The total learning process is divided into phases p_0, p_1, …, p_n. Phases p_0 is a static phase in which the BN structure is learned using algorithm 2 from existing data. Phases p_1, …, p_n are dynamic phase in which BN structure is updated using algorithm 3 based on new incoming data.

4.3. Evolve Bayesian network parameters

Finally, the parameter evolving method is proposed. Since GMM is used to approximate the BN inference, the key problem is to update the GMM parameters. Based on the EM algorithm for GMM [27], the BN parameters updating algorithm is shown in algorithm 5. When using EM algorithm to learn the BN parameters, the final values of parameters and intermediate results are stored in memory. In order to update the parameters of GMM, the current values of π_m, μ_m and Σ_m are used to calculate the distribution of latent variables from the updated data (line 4). Then new π_m, μ_m and Σ_m can be calculated using maximum likelihood based on the distribution of latent variables and parameters are updated (lines 5 – 6). If the parameters are not changed obviously, the algorithm will terminate (line 7 – 9). The algorithm repeats the steps for many times until the log likelihood converges.

Algorithm 5: update BN parameters

input: current data set D, incremental data set ΔD, current BN parameters
output: new BN parameters
1 D ← removeExpiredData(D)
2 D ← D +ΔD
3 noChange ← true
4 use current parameters π_m, μ_m and Σ_m to calculate the distribution of latent variables based on D
5 calculate new π_m, μ_m and Σ_m using maximum likelihood based on the distribution of latent variables
6 update π_m, μ_m and Σ_m
7 **if** ($\forall$ $p_m \in \{\pi_m, \mu_m, \Sigma_m\}$ $|\Delta p_m|/p_m < \lambda \wedge$ noChange) **then**
8 **return** $\{\pi_m, \mu_m, \Sigma_m\}$
9 noChange ← false
10 **if** (the log likelihood does not converges) **then**
11 goto 3
12 **return** $\{\pi_m, \mu_m, \Sigma_m\}$

5. Experimental Evaluations

5.1. Evaluations of EBN

The authors evaluated the EBN method separately since it is the key part of this paper. The BN structure in Fig.2 is a special structure and the authors can't find real data with corresponding result BN structures for accuracy evaluation. Therefore it is reasonable to draw samples from a known BN, apply structure learning on the synthetic data, and compare the learned structure with the original one. The original BN structure contains 783 event types (nodes) and 1752 edges. Since the authors want to evaluate an evolving BN model, sampling data from all phases is needed. In the first step, data set are sampled from the original BN structure. The BN structure is changed randomly at each dynamic phase and an incremental data set (a data window) is sampled based on the changes of BN structure. The time window N_T is set to 15. In order to evaluate the accuracy of BN structure learning and updating, the authors use the Structural Hamming Distance (SHD) [28]. The SHD is defined as the number of the following operators required to make the BNs match: add or delete an edge. This distance compares the learned and original BN equivalence class structure without penalizing an algorithm

for structural differences that cannot be statistically distinguished. Since the real BN structures of each phase are known, the SHD value can be calculated for each phase and the average value is used to evaluate the evolving model accuracy. In order to evaluate the running performance the mean of total learning time is used. A server with Xeon E3 processor and 16GB memory is used in the experiments for EBN.

In the first experiment, the parameters δ and θ are evaluated. The sample number for phase p_0 is 50,000 and the data window for each dynamic phase contains 2,000 samples. The phase number n is set to 30. The average SHD values and total running time values for different δ and θ are shown in table 1. It can be found that when δ increases, the accuracy is decreased but the running time is also decreased. The reason is that when δ becomes larger, more small changes of BDe metric are ignored. From table 1, it can also be found that the accuracy is almost not affected by θ. The performance is increased when θ is increased from 0.1 to 0.4 but decreased when θ is changed from 0.4 to 0.8. Since θ is a threshold value that controls when the BDe metric should be calculated directly from skeleton, it has an optimized value. In the following experiments the authors set δ = 0.03 and θ = 0.4 (because this setting can get acceptable compromise between accuracy and running time).

Table 1. Average SHD values and total running time values for different δ and θ. The values inside the parenthesis are total running time values (seconds)

θ \ δ	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08
0.1	155 (83.5)	162 (79.1)	168 (77.5)	179 (75.2)	194 (72.3)	211 (68.8)	227 (64.6)	245 (52.5)
0.2	159 (80.9)	157 (77.5)	174 (76.2)	186 (74.8)	192 (70.2)	205 (66.6)	229 (63.2)	239 (49.7)
0.3	162 (78.3)	154 (74.8)	178 (73.3)	188 (72.4)	188 (69.1)	214 (63.9)	232 (61.2)	238 (47.2)
0.4	148 (77.3)	169 (72.2)	162 (71.9)	172 (70.0)	205 (67.7)	218 (62.8)	219 (61.8)	244 (46.3)
0.5	151 (77.5)	155 (75.7)	169 (74.6)	171 (72.2)	192 (68.4)	203 (64.4)	218 (63.9)	251 (48.8)
0.6	161 (81.2)	170 (79.9)	158 (77.7)	168 (72.8)	206 (69.9)	209 (67.0)	223 (66.1)	248 (52.2)
0.7	149 (86.1)	164 (84.8)	155 (82.4)	182 (78.4)	191 (73.2)	212 (69.6)	225 (67.4)	239 (54.9)
0.8	153 (89.2)	158 (87.6)	163 (84.3)	175 (80.3)	187 (75.3)	215 (73.3)	231 (69.9)	244 (58.8)

In the next experiment the authors compare the accuracy and performance of the EBN method of this paper with a typical sampling-based method nsDBN [21] and a recent search-based method iHCMC [25]. Constraints are added to the other two methods to make sure directed edges can only be added according to the structure of fig. 2. For nsDBN, the "Unknown Number and Unknown Times of Transitions (UNUT)" setting is used and the parameters are set as λ_s=5, λ_m=1. The dynamic phrases number n is set to 30. The accuracy and performance of the three methods are evaluated for different window size and total data size. The window size value w is set to 1000, 2000, 3000, 4000 and 5000 at each time. The sample number in static phase p_0 is set to 50×w. The sampling number of nsDBN is set to the same as total data size and 20% of the samples are used for burn-in. In iHCMC, the parameter K is set to 8.

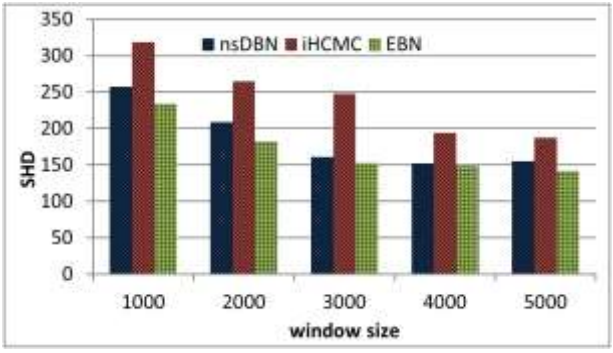

Fig. 4. The accuracy of 3 methods with different window size and data size. Dynamic phase number n is fixed at 30.

The accuracy evaluation result of the 3 methods with different window size and data size is shown in Fig. 4. The accuracy of all methods is increased when data size and windows size become larger. However, the increment of accuracy becomes unobvious when w > 3000. The accuracy of EBN outperforms iHCMC for about 4%. The authors believe the main reason is that EBN searches from top-k local results instead of the maximum result only. Although the accuracy of EBN outperforms nsDBN when window size is relatively small, when the data size (and the sampling size of nsDBN) is large enough, the accuracy of nsDBN and EBN has slight difference.

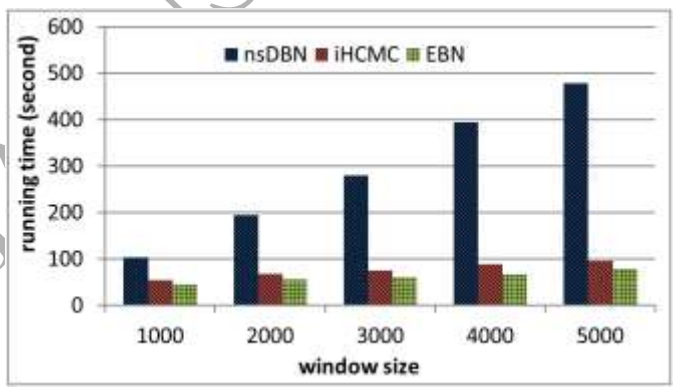

Fig. 5. The performance of 3 methods with different window size and data size

The performance of the 3 methods are evaluated with different window size and the result is shown in Fig. 5. It can be found that the performance of sampling method nsDBN is obviously worse than the other two methods. The running time of nsDBN increases almost linearly since the sampling increases linearly when the window size increases. If only best BN structure is needed (but not distribution of structures), search-based methods have better performance than sampling-based methods. However, sampling-based methods allow the probability or importance of individual edges to be evaluated over all possible networks. The running time of EBN and iHCMC also increases with window size since more calculations are needed to calculate the scores. However another important factor that affects the performance is how many times the score function is called. That factor is related to the BN structure and data set. It can also be found that the total performance of EBN outperforms iHCMC. The main reason is that EBN uses an incremental score calculation method. Although the structure updating algorithm of EBN searches from multiple local peaks, it searches parallel to improve the performance. From fig. 4 and fig. 5 the authors concluded that EBN outperforms the other methods in both accuracy and performance.

5.2. Evaluations of predictive CEP

In this section the experimental evaluation of the complete PCEP-EBN method using both real and simulated data is reported. The real application data comes from the PEMS traffic monitoring network in the area of the Highway 101 to Los Angeles[1]. The real data is stored in files and a player is used to produce event stream from the files. The distribution of this data set is relatively simple. In order to evaluate more complex data distribution, the authors developed a traffic simulation system based on the road traffic simulation package SUMO [29]. The simulation system supports getting event data from the traffic network and executing actions. SUMO supports "induction loops" which can detect vehicles that pass corresponding areas. Getting the location of every vehicle at any time is also supported (this can be used to simulate GPS). The event manager is connected to SUMO through the TraCI (Traffic Control Interface) to get primitive events such as induction loop information and vehicle location information. In the simulation system, the authors set a road network of 15×15 intersections and 80 thousand vehicles. Every road has 2 lanes and every intersection has traffic lights. In order to simulate real traffic system, a set of rules are defined. Every vehicle has a home location and an office location. A vehicle v_i runs between home and office with probability p_i. The vehicles also go to other places such as supermarket, hospital, etc., with corresponding probabilities. The data distribution is changed in different phases. A separate server is used to run the simulation system.

In the experiments, method of this paper is compared with Adaptive Dynamic Bayesian Network (ADBN) [4] and Deep Belief Network (DBN) [11] using both real data and simulated data. The performance is evaluated based on Mean Absolute Percentage Error (MAPE) which is defined by:

$$MAPE(y,y') = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_i - y_i'|}{y_i} \quad (6)$$

where N is the data size, y is the real value of traffic flow and y' is the predicted value. Mean accuracy is calculate by 1 – MAPE. In PCEP-EBN, the parameter λ is evaluated and λ=0.03 is selected for the following experiments. The total time span of PEMS data is 24 hours. The time window Δ_t is set to 10 minutes and the predicting phase time span is set to 0.5 hour. In SUMO, 2 days traffic is simulated with a time span of 480 minutes. The time window Δ_t is set to 2 minutes and the predicting phase time span is set to 10 minutes.

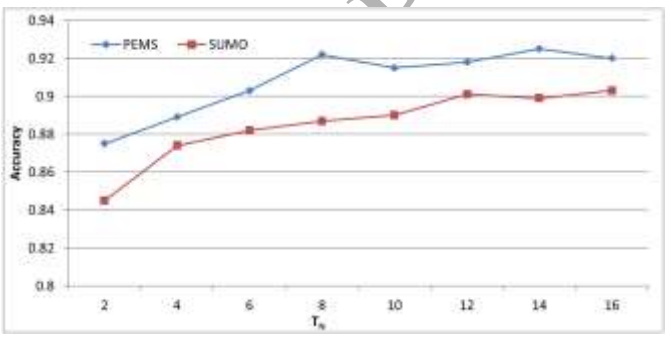

Fig. 6. Accuracy of PCEP-EBN with different time window number TN

In order to decide the time window number T_N of the model in fig. 2, different T_N values are evaluated and the result is shown in fig. 6. The accuracy increases with the increment of T_N until a threshold is reached. The reason is that, when T_N is too small, some nodes that should affect the current node are missed in the model. When T_N is too large, nodes that are too old will not affect the current node and they will be ignored in the learning process. In the following experiments, the authors set T_N = 8 for PEMS data and T_N = 12 for SUMO data.

The accuracy evaluation result for real data and simulated data is shown in fig.7 and fig. 8. The curve of traffic flow change with time and it can roughly be found that PCEP-EBN outperforms other methods. In order to display the result more clearly, the MAPE of the 3 methods for real data and simulated data is shown in fig. 9 and fig. 10. The total percentage error is shown in table 2.

Table 2. Total percentage error for 3 methods on 2 types of data

	ADBN	DBN	PCEP-EBN
PEMS data	15.95%	11.79%	8.12%
SUMO data	18.95%	14.59%	7.78%

Based on the total MAPE value and MAPE curve it can be found PCEP-EBN outperforms other methods obviously. The reason is that PCEP-EBN uses an evolving BN model which can adjust itself according to the change of data distribution. The ADBN model is an adaptive model which can cluster historical data, learn models from data clusters, and select appropriate model at run time. But this model cannot adjust it structure and parameters according to new data distribution. That is why the ADBN get worse accuracy than PCEP-EBN. Although the deep belief network method uses deep architecture to get better accuracy, it still cannot address the problem of distribution drift well. It can also be found that the advantage of PCEP-EBN is more obvious in SUMO data than in PEMS data. The reason is that more data distribution drift is simulated in SUMO data. From all experiments it can be found that PCEP-EBN is an effective method and it outperforms other popular methods when processing data with distribution drift. The percentage error is high at the begin of experiment for all methods because the denominator of MAPE is small at the begin.

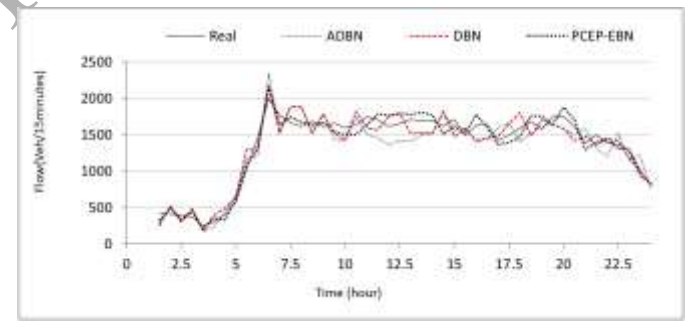

Fig. 7. The accuracy result for PEMS data

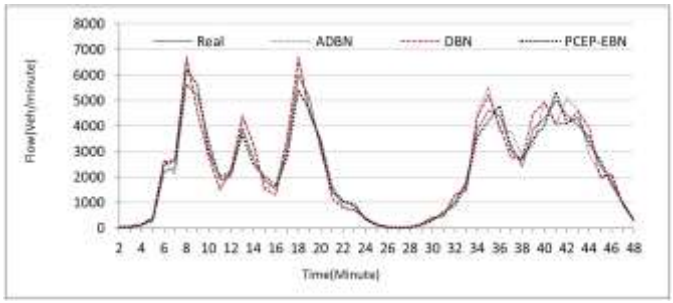

Fig. 8. The accuracy result for SUMO data

[1] PeMS project, https://pems.eecs.berkeley.edu/

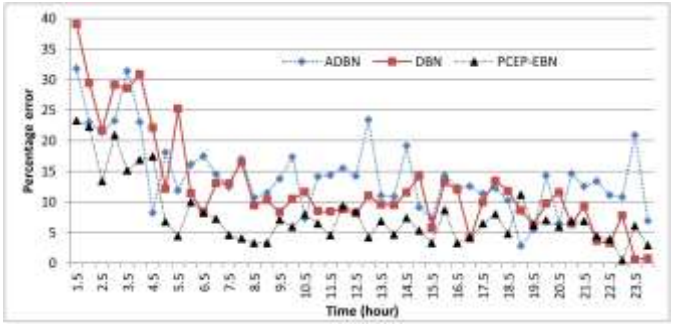

Fig. 9. Percentage error of the three methods on PEMS data

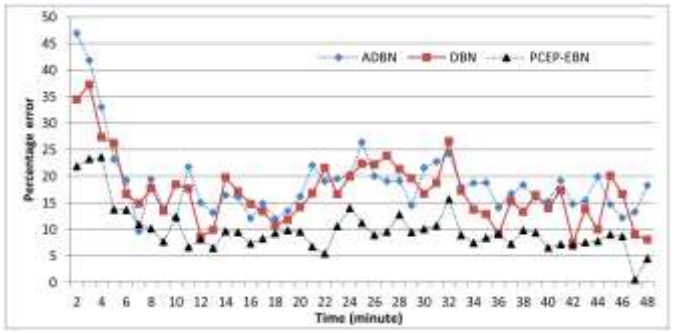

Fig. 10. Percentage error of the three methods on SUMO data

6. Conclusion and Future Work

In this paper the authors propose a predictive complex event processing method based on evolving Bayesian networks. The Bayesian model has two dimensions: event type and time. The Gaussian mixture model and EM algorithm are used as approximate method to infer the Bayesian model. This method supports calculating score metric incrementally. Bayesian network structure and parameters evolving algorithms are proposed. The evaluation results in road traffic domain with both real application data and simulated data shows that this method is effective for predictive complex event processing and it outperforms other popular methods when processing traffic prediction in intelligent transportation systems. This method can also be used in many other areas, especially in proactive event processing systems which support executing some actions to avoid unwanted states.

The performance of the BN structure evolving method still needs to be improved. In the future, the authors will try to use some heuristic methods in the search algorithm. The authors will also consider using parallel searching to improve the performance. The performance of the parameter updating algorithm is also need to be improved.

Acknowledgments

This project is supported by the "Study of Proactive Complex Event Processing for Large-scale Internet of Things" project of National Natural Science Foundation of China (61371116).

References

[1] D. C. Luckham, The power of events: an introduction to complex event processing in distributed enterprise systems, Addison Wesley, Boston, 2002.

[2] O. Etzion and P. Niblett. Event Processing in Action. Manning Publications, 2010.

[3] H. C. Cho and K. S. Lee. Nonlinear networked control systems with random nature using neural approach and dynamic Bayesian networks. Int. J. of Control, Automation, and Systems, 2008, 6(3), pp. 444-452.

[4] A. Pascale and M. Nicoli. Adaptive Bayesian network for traffic flow prediction. In Proc. of the Statistical Signal Processing Workshop (SSP), 2011 IEEE, pp.177-180.

[5] J. R. Alcobe. Incremental Hill-Climbing Search Applied to Bayesian Network Structure Learning. In Proc. of the 15th European conference on machine learning (ECML 2004), Pisa, Italy, Sep. 10 – 24, 2004.

[6] G. Sharon and O. Etzion. Event-processing network model and implementation. IBM System Journal, 2008, 47(2), pp. 321-344.

[7] E. Wu, Y. Diao and S. Rizvi. High-performance complex event processing over streams. In Proc. of the 2006 ACM SIGMOD international conference on Management of data. Chicago, IL, USA, June 27-29, 2006, pp.407–418.

[8] J. Agrawal, Y. Diao, D. Gyllstrom, et al. Efficient pattern matching over event streams. In Proc. of the 2008 ACM SIGMOD International Conference on Management of Data. Vancouver, BC, Canada, June 9-12, 2008, pp.147–160.

[9] L. J. Fülöp, G. Tóth, L. Vidács, et al. Predictive complex event processing: A conceptual framework for combining complex event processing and predictive analytics. In Proc. of the 5th Balkan Conference in Informatics (BCI 2012), Novi Sad, Seibia, September 16-20, 2012, pp. 26-31.

[10] G. Tóth, L. J. Fülöp, L. Vidács, et al. Complex event processing synergies with predictive analytics. In Proc. of the 4th ACM International Conference on Distributed Event-Based Systems (DEBS 2010), Cambridge, United kingdom, July 12-15, 2010, pp.95-96.

[11] W. Huang, G. Song, H. Hong, et al. Deep Architecture for Traffic Flow Prediction - Deep Belief Networks With Multitask Learning. IEEE Transactions on Intelligent Transportation Systems, 2014, 15 (5), pp. 2191 – 2201.

[12] Y. Lv, Y. Duan, W. Kang, et al. Traffic Flow Prediction with Big Data: A Deep Learning Approach. IEEE Transactions on Intelligent Transportation Systems, 2015, 16 (2), pp. 865 – 873.

[13] E. Castillo, J. M. Menéndez, S. Sánchez-Cambronero. Predicting traffic flow using Bayesian networks. Transportation Research Part B: Methodological. 42(2008), pp.482–509.

[14] S. Zhu, L. Cheng, Z. Chu. A Bayesian network model for traffic flow estimation using prior link flows. Journal of Southeast University (English Edition), Vol.29, No.3, 2013, pp.322-327.

[15] V. Fatemeh. Learning the Structure of Large-scale Bayesian Networks using Genetic Algorithm. In Proc. of the 2014 Genetic and Evolutionary Computation Conference, July 12 - 16, 2014, Vancouver, BC, Canada, pp. 855-862.

[16] W. Lam and F. Bacchus. Learning bayesian belief networks: An approach based on the MDL principle. Computational intelligence, 1994, 10(3), pp. 269–293.

[17] Y. M. Shtarkov. Universal sequential coding of single messages. Problems of Information Transmission, 1987, 23(3), pp. 3–17.

[18] D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. Machine Learning, 1995, 20(3), pp. 197–243.

[19] J. Arias, J. A. Gámez, J. M. Puerta. Structural learning of Bayesian networks via constrained Hill Climbing algorithms: Adjusting trade-off between efficiency and accuracy. International Journal of Intelligent Systems. 30(3), 2015, pp.292-325.

[20] A. Lin, B. Xiao, Y. Zhu. An algorithm for bayesian network structure learning based on simulated annealing with adaptive selection operator. In Proc. of the 2013 International Conference on Computer Engineering and Network(CENet 2013),Shanghai, China, July 20-21, 2013, pp.525-532.

[21] J. W. Robinson, A. J. Hartemink. Learning Non-Stationary Dynamic Bayesian Networks. Journal of Machine Learning Research, 2010, vol 11, pp.3647-3680.

[22] A. Yasin, P. Leray. Incremental Bayesian network structure learning in high dimensional domains. In Proc. of the 5th International Conference on Modeling, Simulation and Applied Optimization (ICMSAO 2013), Hammamet, Tunisia, April 28-30, 2013.

[23] K. Yue, Q. Fang, X. Wang, et al. A parallel and incremental approach for data-intensive learning of Bayesian networks. IEEE Transactions on Cybernetics, 45(12), 2015, pp.2890-2904.

[24] S. Li, J. Zhang, B. Sun, et al. An Incremental Structure Learning Approach for Bayesian Network. In Proc. of the 26th Chinese Control and Decision Conference (CCDC 2014), Changsha, China, May 31 - June 2, 2014, pp.4817-4822.

[25] S. Acharya, B. Lee. Causal network construction over event streams. Information Sciences, Vol 261, issue 2014, pp.32-51.

[26] Y. Wang, K. Cao and X. Zhang. Complex Event Processing over Distributed Probabilistic Event Streams. Computers and Mathematics with Applications, 2013, 66 (10), pp.1808-1821.

[27] C. Bishop. Pattern recognition and machine learning (2nd edition). Springer 2010.

[28] I. Tsamardinos, L. E. Brown, C. F. Aliferis. The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning, 65(1), 2006, pp.31-78.

[29] M. Behrisch, L. Bieker, J. Erdmann, et al. Sumo - simulation of urban mobility: An overview. In Proc. of the third international conference on advances in system simulation (SIMUL 2011), Barcelona, Spain, October 23, 2011, pp.63–68.

[30] P. Angelov. Autonomous Learning Systems: From Data Streams to Knowledge in Real-time. Wiley Press,2013.1.

[31] X. Zeng, G. Li. Incremental partial least squares analysis of big streaming data. Pattern Recognition, 47(11), 2014, pp. 3726-3735.

[32] T. Li. PICKT: A solution for big data analysis. Proc. of the 10th International Conference on Rough Sets and Knowledge Technology, Tianjin, China, 2015,pp.15-25.

[33] B. Yevgeniy, V. Olena, P. Iryna, et al. Fast learning algorithm for deep evolving GMDH-SVM neural network in data stream mining tasks. Proc. of the IEEE First International Conference on Data Stream Mining & Processing, Lviv, Ukraine, 2016, pp. 257-262.

[34] E. C. Hall, R. M. Willett. Online learning of neural network structure from spike trains. Proc. of the International IEEE/EMBS Conference on Neural Engineering, Montpellier, France, 2015, pp. 930-933.

[35] M. Valipour. Optimization of neural networks for precipitation analysis in a humid region to detect drought and wet year alarms. Meteorological Applications, 23(1), 2016, pp. 91–100.

[36] J. Zhao, Z. Xu, D. Zuo, et al. Temporal variations of reference evapotranspiration and its sensitivity to meteorological factors in Heihe River Basin, China. Water Science and Engineering 319(1), 2015, pp.1-8.