Hindawi Publishing Corporation
International Journal of Distributed Sensor Networks
Volume 2014, Article ID 159052, 8 pages
http://dx.doi.org/10.1155/2014/159052

Research Article

A Proactive Complex Event Processing Method for Large-Scale Transportation Internet of Things

Yongheng Wang and Kening Cao

College of Information Science and Engineering, Hunan University, Changsha 410082, China

Correspondence should be addressed to Yongheng Wang; yh.wang.cn@gmail.com

Received 25 December 2013; Revised 23 January 2014; Accepted 18 February 2014; Published 23 March 2014

Academic Editor: Gelan Yang

The Internet of Things (IoT) provides a new way to improve the transportation system. The key issue is how to process the numerous events generated by IoT. In this paper, a proactive complex event processing method is proposed for large-scale transportation IoT. Based on a multilayered adaptive dynamic Bayesian model, a Bayesian network structure learning algorithm using search-and-score is proposed to support accurate predictive analytics. A parallel Markov decision processes model is designed to support proactive event processing. State partitioning and mean field based approximation are used to support large-scale application. The experimental evaluations show that this method can support proactive complex event processing well in large-scale transportation Internet of Things.

1. Introduction

The Internet of Things (IoT) bridges the gap between the physical world and its representation within the digital world. In recent years, with the rapid development of information and communication technologies, bandwidth and storage are no longer restricted in IoT applications. The key issue is how to process the massive events produced by large-scale IoT applications (an event means an atomic occurrence of interest in time).

In IoT applications, event processing engines need to process events that arrive from various kinds of sources such as sensors, RFID readers, and GPS. The events generated by devices directly are called primitive events. The semantic information inside primitive events is quite limited. In real application, users pay more attention to high level information such as business logic and rules. For example, each reading operation of the RFID reader at a garage generates a primitive event, but complex events like "the car leaves the garage" are the kind of events that users are really concerned with. To get complex events, many primitive events need to be combined based on some rules. In IoT application systems, business logic is converted into complex events and business logic is processed based on complex events detection. Complex event processing (CEP) [1] is used to process massive primitive events and get valuable high level information from them. As an example, in logistics industry, CEP is used to track the goods and trigger some actions when exception is found.

In many real-time IoT applications, events are uncertain due to some factors such as measuring inaccuracy, signal disturbance, or privacy protection. Usually, we use probabilities to process such uncertainty. Therefore, it is necessary to develop probabilistic event processing engine.

The traditional type of the event processing is reactive processing which means that actions are triggered by events or by the system states. A proactive event processing system is able to mitigate or eliminate undesired future events or states or to identify and take advantage of future opportunities, by using prediction and automated decision making methods [2]. For example, in a transportation system, we can predict some possible congestion states and take some actions to mitigate or eliminate the congestion states. The proactive event processing systems use predictive analytics (PA) technology which predicts future events or system states through analysis of historical events. The system also uses iterated decision processes (DP) technology which analyzes system states and selects appropriate actions to achieve expected

states. The CEP, PA, and DP technologies have been studied widely, but there are few papers about how to integrate them together to support proactive event-driven system. Proactive event processing in large-scale transportation IoT needs to process massive historical events and analyze complex states iteratively which makes most of the existing algorithms unable to be used directly. Furthermore, proactive event processing systems need high performance since they usually have to take actions in time.

In this paper, we propose a proactive complex event processing (Pro-CEP) method for large-scale transportation Internet of Things. We designed a multilayered adaptive dynamic Bayesian network (mADBN) model for predictive analytics. Based on probabilistic complex event processing, our method uses concurrent actions Markov decision processes (MDP) to integrate CEP and PA.

2. Related Works

2.1. CEP and Probabilistic CEP. Complex event processing detects complex events based on a set/sequence of occurrences of primitive events by monitoring the event flow continuously. Etzion and Niblett introduced the concept, architecture, and methods of complex event processing in their book [3]. Event processing agent (EPA) is a component that applies some rules to generate complex events as output based on a set of input events. Event processing network (EPN) is a network of a collection of EPAs, event producers, and event consumers linked by channels. The network in EPN is used to describe the event processing flow execution. Luckham first introduced the event processing network in the field of modeling [1].

Most of the CEP methods in active databases and RFID applications use fixed data structures such as tree, directed graph or Petri-Net. Those methods cannot extend event query language according to the change of requirement or optimize event query language flexibly. SASE [4] is a high performance complex event processing method which uses nondeterministic finite automaton (NFA). Recently, there is some work about improving the SASE method.

Most probabilistic CEP engines are based on sequential variants of probabilistic graphical models, such as hidden Markov models, dynamic Bayesian Networks, and conditional random fields. Recently, some probabilistic CEP methods based on NFA are proposed. In the work of [5], a data structure called chain instance queues is used to detect complex events with a single scanning of probabilistic event stream. In another paper [6], an optimized method based on SASE is used to calculate the probability of complex events and to obtain the confidence value of the complex patterns given by user against uncertain raw input event stream. Compared with our work, these CEP methods are not integrated with PA and DP to support proactive application.

2.2. Predictive Analytics with Bayesian Networks. Predictive analytics applies some statistical and data mining techniques such as clustering, classification, and regression. In the case of predictive analytics methods for complex event data, some attributes of the monitored system are predicted based on the previously monitored events.

Recently, Bayesian network (BN) has been used in predictive analytics. Castillo et al. used BN to predict the random character of the total mean flow level and the variability of origin-destination pair flows [7]. In the work of Pascale and Nicoli, an adaptive BN was proposed in which the network topology changes according to the nonstationary features of traffic [8]. Gaussian mixture model (GMM) was used to describe the joint probability distribution between the cause nodes and the effect node in a Bayesian network. In the work of Hofleitner et al., dynamic Bayesian network (DBN) is used in PA and a model based on hydrodynamic traffic theory is also proposed to learn the distribution of vehicles on arterial road segments [9]. Compared with our work, these methods are not designed for large-scale IoT applications and cannot process massive event data in proactive event-based system.

2.3. Proactive Event Processing and Markov Decision Processes. Recently, many proactive applications have been developed, for example, proactive security systems, proactive routing in mobile wireless networks, and proactive service level agreement negotiation in service oriented systems. In the work of Engel et al., a proactive event-driven computing framework based on CEP, PA, and Markov decision processes (MDP) is proposed [2, 10]. In this framework, the event processing model is extended and two more types of agent are included: predictive agents which can predict future uncertain events based on the prediction models and proactive agents which can select the best proactive action that should be taken.

MDP has been studied for many years and some variants emerged recently. In the area of large-scale IoT, the main challenge is the combination explosion problem caused by large state space. Mainly, there are two research directions on this problem. The first one tries to simplify the problem by using the information from the application domain, including state aggregation methods, time aggregation methods, and action elimination methods. These methods are related to specific application domain, and it is not easy to find a common solution. In the second direction, approximate methods are developed, including approximate dynamic programming, dynamic programming based on events, and selective approximation. The key issue in these methods is how to approximate the value function during the optimization process. When applied to large-scale IoT, MDP has larger size and some new properties. The model design and large-scale calculation become a new challenge. Compared with our work, current MDP methods are not combined with CEP and cannot process the large state space in large-scale proactive event-based system.

3. Backgrounds

3.1. System Architecture. The system architecture of our work is shown in Figure 1. Raw events are generated by devices such as RFID reader, radar, and GPS. We assume the events can be imprecise because of the limitation of measuring accuracy or signal disturbance. The probabilistic raw events

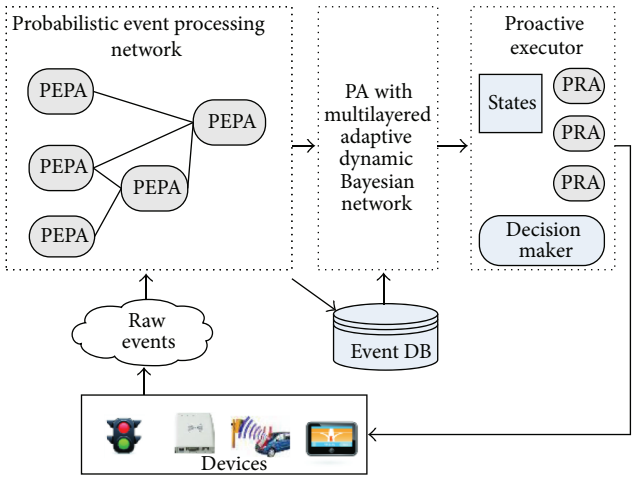

Figure 1: System architecture.

are processed by the probabilistic event processing network (PEPN) which is composed of many interconnected probabilistic event processing agents (PEPA). Different PEPAs can be connected since hierarchical complex event is supported. The PA component can predict the system states from the complex events based on the mADBN model. Complex events are saved to event database and the historical data can be used to train the models in PA components. The decision maker selects appropriate actions according to the predicted states and assigns some proactive agents (PRA) to execute the actions.

3.2. Event Model and Probabilistic CEP

Definition 1 (primitive event). A primitive event in a stream means an atomic occurrence of interest in time. A probabilistic primitive event is represented by $\langle A, T, \Pr\rangle$, where A is the set of attributes and T is the timestamp that the event occurs. Pr is the concrete probability value used to present the occurrence probability of the event.

The probability value represents the possibility that an event is converted accurately from truthful data of nature to digital data used for computing in electronic devices.

Definition 2 (complex event). A complex event is a combination of primitive events or complex events by some rule. A probabilistic complex event is represented by $\langle E, R, \mathrm{Ts}, \Pr\rangle$, where E is the set of elements that compose the complex event, R is the rule of the combination, Ts is the time span of the complex event, and Pr is the probability value.

The main complex event patterns in our work include ALL, ANY, COUNT, and SEQ. In this paper, the COUNT pattern can be used to count the number of vehicles in a specified area during specified time span. The SEQ pattern can be used to represent the moving path of a vehicle. The detailed meaning of the patterns can be found in [3]. Those patterns can be composed to create hierarchical complex patterns.

In a large-scale IoT application, we may need to support distributed event streams processing. In this paper, we assume the primitive events from different event streams are independent. In the same event stream, some primitive events have Markov property which means that the probability of next event is only related to the current event in the sequence but has nothing to do with previous events. Like the work of [6], we use condition probability table (CPT) to save and sort the condition probability.

We extended the SASE method to support probabilistic CEP by adding probability value for each event in the active instance stacks (AIS). Detailed method and algorithm can be found in another paper of the same author [11].

4. Proactive CEP Method

4.1. Predictive Analytics Model. The mADBN model is shown in Figure 2. In this model, there are a state plane and a set of location planes. Each plane is represented by an adaptive dynamic Bayesian network with two dimensions: time and space. In the state plane, nodes denote the states of the system observed at different time instants and/or spatial locations, edges being their probabilistic relations. In Figure 2, the directed edges mean that the state of (i, t) depends on a set of other states before time t. The term "dynamic" in mADBN means that we are modeling a dynamic system. The term "adaptive" means that the graph structure is created based on machine learning from historical data. Each location plane represents the location change of a vehicle (moving path). The directed edges represent the transaction probability from one location to another and the directed edges with solid line represent the real paths.

The graph structure of the state plane can be learned based on analysis of the vehicle location planes. Given a node set V and a constraint set C, the Bayesian network structure learning problem is to find an optimized set of edges and an optimized set of distribution parameters. Mainly, there are two types of methods for Bayesian network structure learning: constraint-based method and search-and-score method. Constraint-based method checks the conditional dependence in data systematically and creates the network structure based on the dependence. Search-and-score method creates reasonable network structure based on a score function. Search-and-score method gets better result but the performance is not good for massive data. In this paper, we combine the two methods together. Global CPT can be created using the historical data in object location planes. Candidates are selected based on global CPT using constraint-based method and then a search-and-score algorithm is used to learn the structure of Bayesian network. In our method, local CPT is first created for each object plane and then all local CPTs are summed up and normalized to get the global CPT. The method was implemented with a MapReduce algorithm to support parallel calculation.

The main idea of the structure learning algorithm using search-and-score is to maximize the score function. Like

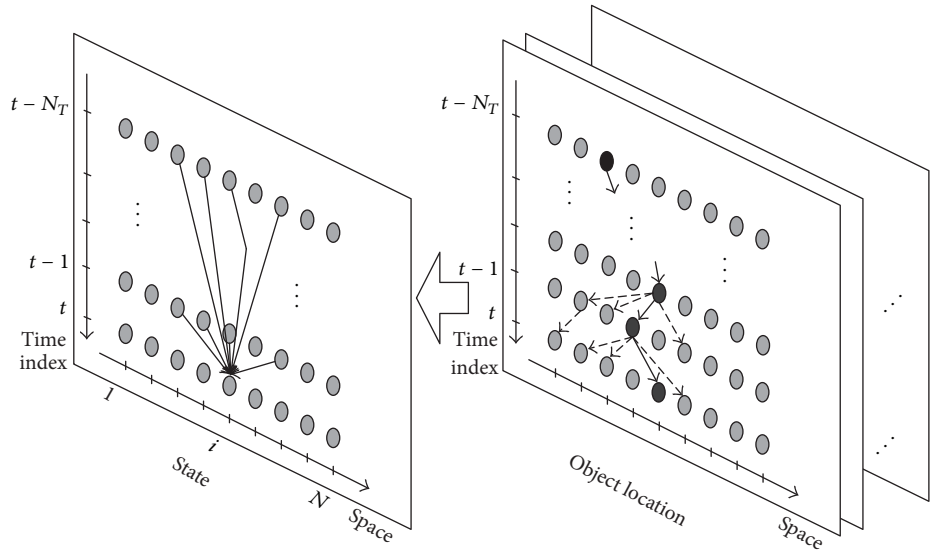

Figure 2: The mADBN model.

the work of [12], we use Bayesian information criterion (BIC) score function which is defined as follows:

$$\mathrm{SBIC}(D, G, \Theta) = \log P\left(D \mid \widehat{\Theta}, G\right) - \frac{d}{2}\log m + o(1), \quad (1)$$

where D is the data set, G is the graph structure, $\widehat{\Theta}$ is the maximum likelihood distribution parameters for D, d is the number of edges, and m is the sample size per vertex. The algorithm has an expansion stage and a contraction stage. During the expansion stage, edges that can increase the score function are added. During the contraction stage, edges are removed if that does not decrease the score function. The candidate parent vertexes for a specific vertex are selected according to the CPT.

4.2. Predictive Analytics Method. In the mADBN model of Figure 2, let $f_{i,t}$ represent the flow state of node (i,t) and let $\mathrm{pa}(i,t)$ represent all the parent nodes of (i,t). N_P is the number of nodes in $\mathrm{pa}(i,t)$. The set of flow states for $\mathrm{pa}(i,t)$ is $F_{\mathrm{pa}(i,t)} = \{f_{j,s} : (j,s) \in \mathrm{pa}(i,t)\}$. The joint distribution of all nodes in the state plane can be expressed as follows:

$$p(F) = \prod_{i,t} p\left(f_{i,t} \mid F_{\mathrm{pa}(i,t)}\right). \quad (2)$$

According to the Bayesian formula, the conditional probability $p(f_{i,t} \mid F_{\mathrm{pa}(i,t)})$ can be calculated by

$$p\left(f_{i,t} F_{\mathrm{pa}(i,t)}\right) = \frac{p\left(f_{i,t}, F_{\mathrm{pa}(i,t)}\right)}{F_{\mathrm{pa}(i,t)}}. \quad (3)$$

The joint distribution $p(f_{i,t}, F_{\mathrm{pa}(i,t)})$ is modeled with GMM as follows:

$$p\left(f_{i,t}, F_{\mathrm{pa}(i,t)}\right) = \sum_{m=1}^{M} \alpha_m g_m\left(f_{i,t}, F_{\mathrm{pa}(i,t)} \mid \mu_m, C_m\right), \quad (4)$$

where M is the number of nodes and $g_m(\cdot \mid \mu_m, C_m)$ is the mth Gaussian distribution which has a $(N_P + 1) \times 1$ vector of mean values μ_m and a $(N_P + 1) \times (N_P + 1)$ covariance matrix C_m. Like the work of [13, 14], we use the EM algorithm to infer the parameters $\{\alpha_m, \mu_m, C_m\}_{m=1}^{M}$ from the historical data. The conditional distribution $p(f_{i,t} \mid F_{\mathrm{pa}(i,t)})$ can be derived from $p(f_{i,t} \mid F_{\mathrm{pa}(i,t)})$ and the estimate $\widehat{f}_{i,t}$ can be calculated from $F_{\mathrm{pa}(i,t)}$ using the minimum mean square error (MMSE) method.

4.3. Decision Making with MDP. According to the properties of proactive complex event processing in IoT application, we extend the traditional MDP as follows.

Definition 3 (parallel MDP for proactive complex event processing). A parallel MDP for proactive complex event processing is represented as $\langle I, S, \vec{A}, P, R, E_c, C, T_c \rangle$, where I denotes the finite set of agents that process actions and S denotes the set of states with a special initial state S_0. $\vec{A} = \times_{i \in I} A_i$ denotes the set of actions, where A_i is the action from the ith agent. $P : S \times A \times S \to [0,1]$ denotes the set of state transformation functions, where $P(s, \alpha, s')$ represents the probability that state s transforms to s' with the execution of action α. $R : S \to \mathrm{Re}$ denotes the set of reward functions (Re means the set of real numbers). Every $(s, \alpha) \in S \times A$ satisfies $\sum_{s' \in S} p(s, \alpha, s') = 1$. E_s is the set of complex events in IoT which affects the system states. C is the context of event and agents act according to different context. $T_s : S \times E_s \times S \to [0,1]$ denotes the transforming probability which represents the affection of complex events to the system states.

The proactive event processing system starts from an original state and selects one or more actions executed by agents according to the policy. The execution of actions

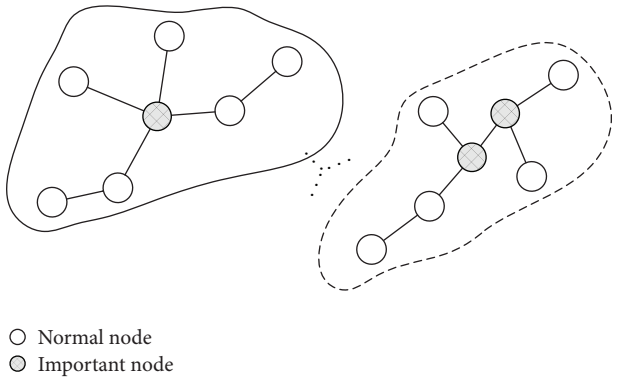

Figure 3: Network partition based on important nodes.

changes the primitive events and the transforming probability values from old states to new states are calculated based on the analysis of the complex events. The key issue of MDP is to find a policy $\pi : S \rightarrow A$ which reflects a state to a set of actions. The best policy maximizes the total rewards. The selection of policy is based on the following equation (value function):

$$v_\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p\left(s' \mid s, \pi(s)\right) v_\pi\left(s'\right). \quad (5)$$

The update of policy is based on the following equation:

$$\pi^*(s) = \arg\max_{a \in A}\left(r(s,a) + \gamma \sum_{s' \in S} P\left(s' \mid s, a\right) v_\pi\left(s'\right)\right), \quad (6)$$

where γ is the decay factor. In traditional MDP, actions are executed in order. Recently, some work of parallel MDP is proposed in which actions are partitioned into groups and group of actions executed parallel without affecting each other. In this paper, the MDP model executes actions parallel to the same state and all actions work together to transform the state into an expected one. Our MDP method is based on the following assumptions.

Assumption 4. The vehicles in large-scale IoT can be partitioned based on context and each group is related to a substate of the system. We need to consider the substates of the system but not the state of each vehicle.

An event context is a specification of conditions that groups event instances so that these instances can be processed in a related manner. For example, when processing the traffic congestion problem, the traffic flows at roads or junctions are substates and the system global states are composed of all substates.

Assumption 5. In large-scale transportation IoT, the state of a node depends on the states of its neighbors (the neighbors here mean the nodes that are linked with the current node within some distance).

Assumption 6. In large-scale transportation IoT, nodes can be partitioned into two classes: important and normal. The nodes whose states we want to control proactively are important nodes. The network can be partitioned based on important nodes and the state of important nodes can be changed by changing the state of their neighbors as shown in Figure 3.

According to Assumptions 4, 5, and 6, a new variable $G = (V, E)$ which represents the structure of subgraphs is added to the parallel MDP model, where V is the set of vertexes and E is the set of edges. An edge (i, j) exists which means that the state of node i affects the state of node j.

Definition 7 (neighbor state nodes). For any state node $i \in V$, the neighbor state nodes are $N(i) = \{j \in V \mid d_{i,j} \leq k\}$, where $d_{i,j}$ is the distance of nodes i and j and k is a threshold value.

Definition 8 (state transformation decomposition). Assume there are n substate nodes and n corresponding actions; according to the assumptions, we get

$$p\left(s' \mid s, a\right) = \prod_{i=1}^{n} P_i\left(s_i' \mid s_{N(i)}, a_i\right), \quad (7)$$

where $s_{N(i)} = \{s_j \mid j \in N(i)\}$.

Definition 9 (reward decomposition). Assume there are n substate nodes and n corresponding actions; the total reward can be written as follows:

$$r(s,a) = \sum_{i=1}^{n} r_i\left(s_{N(i)}, a_i\right). \quad (8)$$

A policy π can also be decomposed as $(\pi_1, \pi_2, \ldots, \pi_n)$.

Definition 10 (occupation measure). Let $\mathbf{S}$ be the state space and let $x^0 \in \mathbf{S}$ be the original state of MDP. The occupation measure $P_{x^0,\pi,\gamma} : \mathbf{S} \rightarrow [0,1]$ is defined as follows:

$$P_{x^0,\pi,\gamma}(y) = (1-\gamma) \sum_{t=0}^{\infty} \gamma^t P_\pi\left(S^t = y \mid S^0 = x^0\right), \quad (9)$$

where $y \in \mathbf{S}$. With occupation measure, the value function can be rewritten as follows:

$$V_\pi\left(x^0\right) = \frac{1}{1-\gamma} \cdot \sum_{y \in \mathbf{S}} P_{x^0,\pi,\gamma}(y) \cdot r(y, \pi(y)). \quad (10)$$

The key issue is to find a solution to calculate $P_\pi(S^t = y \mid S^0 = x^0)$ approximately. According to the decomposition we have defined, it can be inferred from $p_i(x_i^t \mid x_{N(i)}^{t-1}, \pi(x_{N(i)}^{t-1}))$. Now, the problem is to find an approximation $\widehat{G}_\pi^{t,i}(x_i^t \mid x_i^{t-1})$ which satisfies

$$\begin{aligned} \widehat{C}_\pi^t\left(y \mid x^0\right) &= \sum_{s^{t-1}} \widehat{G}_\pi^t\left(y \mid x^{t-1}\right) \cdot \widehat{C}_\pi^{t-1}\left(x^{t-1} \mid x^0\right) \\ &= \prod_{i=1}^{n} \sum_{x_i^{t-1}} \widehat{G}_\pi^{t,i}\left(y_i \mid x_i^{t-1}\right) \cdot \widehat{C}_\pi^{t-1,i}\left(x_i^{t-1} \mid x_i^0\right), \end{aligned} \quad (11)$$

where $\widehat{C}$ is the approximation of the conditional probability $P_\pi(S^t = y \mid S^0 = x^0)$ defined by $\widehat{C}^1_\pi(y \mid x^0) = \prod_i \widehat{G}^{1,i}_\pi(y_i \mid x^0_i)$ and $\forall t, \widehat{C}^t_\pi(y \mid x^0) = \prod_{i=1}^n \widehat{C}^{t,i}_\pi(y_i \mid x^0_i)$.

As an effective method for analyzing large and complex stochastic models, mean field theory is widely used in many areas. Using mean field theory, the effect of all other individuals on any given individual is approximated by a single averaged effect. Like the work of Sabbadin et al. [15], we applied mean field approximation which approximates a multidimensional distribution $P(u_1, u_2, \ldots, u_m)$ as the joint distribution G of m independent variables $u_1, \ldots, u_m$. G should be as close to P as possible according to Kullback-Leibler divergence:

$$\mathrm{KL}(GP) = E_Q\left[\log\left(\frac{G}{P}\right)\right]. \tag{12}$$

For state 1 (1 is the first time point and 0 is the start point), we get the following approximation using Kullback-Leibler divergence:

$$\widehat{G}^1_\pi = \arg\min_{P^1\pi} \mathrm{KL}\left(G^1_\pi\left(S^1 \mid S^0\right) \cdot P^0\left(S^0\right) \mid P_\pi\left(S^1 S^0\right) \cdot P^0\left(S^0\right)\right). \tag{13}$$

After state transformation decomposition, the approximation can be written as

$$\widehat{G}^{1,i}_\pi\left(x^1_i \mid x^0_i\right) \propto \exp E_{p^0}\left[\log p_i\left(x^1_i \mid x^0_i, S^0_{N(i)\setminus\{i\}}, \pi_i\left(x^0_i, S^0_{N(i)\setminus\{i\}}\right)\right)\right]. \tag{14}$$

The rest of the iteration is similar to (14). The data processing servers include a master node and N slave nodes. The main iteration is executed at the master node and the calculation of substates is partitioned into slave nodes and executes parallel.

5. Experimental Evaluations

We developed a transportation IoT simulation system based on the road traffic simulation package SUMO [16]. The system supports mobility trace of vehicles by using an OpenStreetMap [17] road map of Beijing. On the roads, we set many virtual induction loops which can detect vehicles that pass corresponding areas. The virtual RFID or GPS readers are implemented using the TraCI interface of SUMO to get the induction loop variables. Each virtual induction loop covers a region. We selected 114 junctions from the map and set 160 thousand vehicles in the system. In order to simulate real traffic system, a series of rules are defined. Each 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 and hospital, with corresponding probabilities. Based on this simulation system, we evaluated the precision and performance of Pro-CEP. We used 5 Lenovo ThinkServer RD series servers with 4 GB memory as a cluster and the operating system is Ubuntu 12.

Table 1: Deviation of two methods. The average percent is calculated by ((average deviation)/(average observed vale)) ∗ 100%.

	Deviation	
	ABN	Pro-CEP
Max	286	122
Min	16	12
Average	126.33	71.5
Average percent	16.44%	9.3%

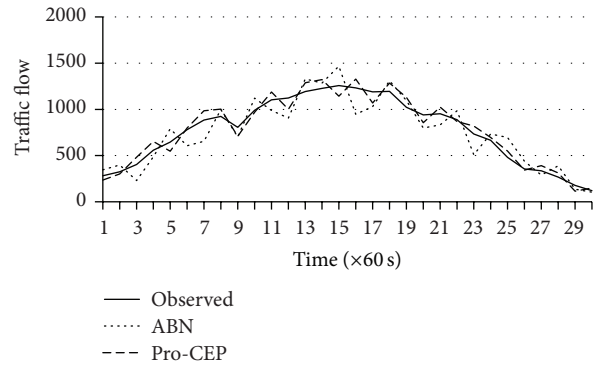

Figure 4: PA accuracy for a typical node.

We first run the simulation for many times to get the historical data of the vehicle paths. In the first experiment, the accuracy of our PA method is compared with ABN [8] which uses Bayesian network but has no vehicle location planes like our model. The result is shown in Figure 4 and Table 1. From the figure, we can see that the accuracy of Pro-CEP is better than traditional method ABN. The reason is that Pro-CEP use the object location plane to create a more precise Bayesian network structure.

In the next experiment, we compared the mean congestion of the system with and without Pro-CEP. Since we have not found other methods that have the same function as Pro-CEP, no other method is compared. The congestion is partitioned into 10 levels, where "0" means no congestion and "9" means the highest congestion. The result is shown in Figure 5. We can see the mean congestion level reduced obviously when Pro-CEP is used. The reason is that Pro-CEP can predict the congestion and take some actions to avoid it. The reduction of congestion level is more obvious when the congestion level becomes high because the system can redirect more vehicles to light loaded nodes in such circumstance.

In the next experiment, we evaluated the performance of Pro-CEP with different server number and data size. The important node number is fixed at 30 and Bayesian model constructing time is not included. The result is shown in Figure 6. As we can see, the running time increases when the vehicle numbers become larger. The reason is that more vehicles need to be rerouted which increases the calculation of the algorithm. The performance for 5 servers is higher than

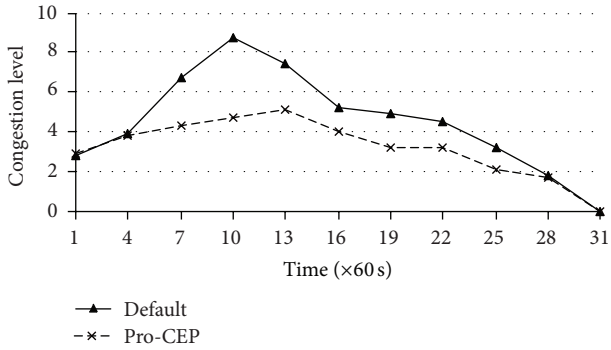

Figure 5: Mean congestion level over time.

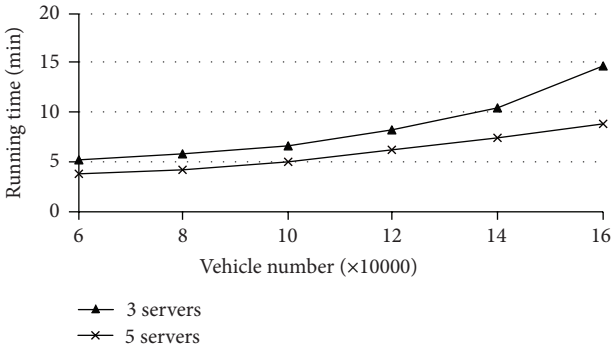

Figure 6: Performance of Pro-CEP with different server number and data size.

3 servers and the running time of the former increases slowly because the parallel method can take advantage of multiple servers.

We also evaluated the performance of Pro-CEP with different server number and important node number. The result is shown in Figure 7. The vehicle number is fixed at 160 thousand. The result shows that the running time increases obviously when the important node number becomes larger. The reason is that more important nodes means more substates in the parallel MDP which increases the calculation of the algorithm.

From all the experiments, we can see that Pro-CEP can support proactive complex event processing in large-scale transportation Internet of Things. The PA method gets high accuracy based on the mADBN model and Bayesian network. The parallel MDP model has obvious effect for congestion control in large transportation IoT. The performance of Pro-CEP decreases when the vehicle number and important node number become large, but we can get better performance with more servers.

6. Discussions and Conclusion

In this paper, we proposed a proactive complex event processing method for large-scale transportation Internet of Things based on a novel mADBN model and parallel MDP. A structure learning algorithm using search-and-score is proposed

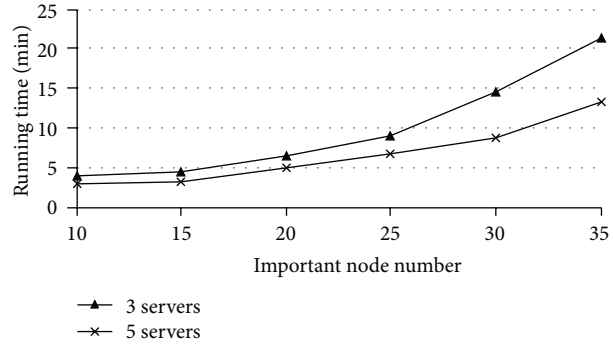

Figure 7: Performance of Pro-CEP with different server number and important node number.

to support accurate PA. A GMM based approximation for Bayesian network, a state partition method, and a mean field based approximation method of parallel MDP are proposed to support large-scale transportation IoT. The experimental evaluations show that this method can support proactive complex event processing well in large-scale transportation Internet of Things.

The performance and scalability of Pro-CEP still need to be improved. In the PA method, the EM algorithm needs to be parallelized to improve scalability. The master node in parallel MDP is the bottleneck and the iteration algorithm still needs to be parallelized further.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (no. 61371116) and the Hunan Provincial Natural Science Foundation (no. 13JJ3046).

References

[1] D. C. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*, Addison Wesley, Boston, Mass, USA, 2002.

[2] Y. Engel and O. Etzion, "Towards proactive event-driven computing," in *Proceedings of the 5th ACM International Conference on Distributed Event-Based Systems (DEBS '11)*, pp. 125–136, New York, NY, USA, July 2011.

[3] O. Etzion and P. Niblett, *Event Processing in Action*, Manning Publications, Greenwich, Conn, USA, 2010.

[4] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 407–418, Chicago, Il, USA, June 2006.

[5] X. Chuanfei, L. Shukuan, W. Lei, and Q. Jianzhong, "Complex event detection in probabilistic stream," in *Proceedings of the 12th International Asia Pacific Web Conference (APWeb '10)*, pp. 361–363, April 2010.

[6] H. Kawashima, H. Kitagawa, and X. Li, "Complex event processing over uncertain data streams," in *Proceedings of the 5th international conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pp. 521–526, 2010.

[7] E. Castillo, J. M. Menéndez, and S. Sánchez-Cambronero, "Predicting traffic flow using Bayesian networks," *Transportation Research B*, vol. 42, no. 5, pp. 482–509, 2008.

[8] A. Pascale and M. Nicoli, "Adaptive Bayesian network for traffic flow prediction," in *Proceedings of the IEEE Statistical Signal Processing Workshop (SSP '11)*, pp. 177–180, June 2011.

[9] A. Hofleitner, R. Herring, and P. Abbeel, "Learning the dynamics of arterial traffic from probe data using a dynamic Bayesian network," *IEEE Intelligent Transportation Systems Society*, vol. 13, no. 4, pp. 1679–1693, 2012.

[10] Y. Engel, O. Etzion, and Z. Feldman, "A basic model for proactive event-driven computing," in *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pp. 107–118, Busan, Republic of Korea, 2012.

[11] Y. H. Wang and X. M. Zhang, "Complex event processing over distributed probabilistic event streams," in *Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD '12)*, pp. 1489–1493, 2012.

[12] S. Samaranayake, S. Blandin, and A. M. Bayen, "Learning the dependency structure of highway networks for traffic forecast," in *Proceedings of the 50th IEEE Conference on Decision and Control and European Control Conference*, pp. 5983–5988, 2011.

[13] J. J. Verbeek, N. Vlassis, and B. Kröse, "Efficient greedy learning of gaussian mixture models," *Neural Computation*, vol. 15, no. 2, pp. 469–485, 2003.

[14] N. Kumar, S. Satoor, and I. Buck, "Fast parallel expectation maximization for gaussian mixture models on GPUs using CUDA," in *Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications (HPCC '09)*, pp. 103–109, June 2009.

[15] R. Sabbadin, N. Peyrard, and N. Forsell, "A framework and a mean-field algorithm for the local control of spatial processes," *International Journal of Approximate Reasoning*, vol. 53, no. 1, pp. 66–86, 2012.

[16] M. Behrisch, L. Bieker, and J. Erdmann, "Sumo—simulation of urban mobility: an overview," in *Proceedings of the 3rd International Conference on Advances in System Simulation*, pp. 63–68, Barcelona, Spain, October 2011.

[17] M. Haklay and P. Weber, "Openstreet map: user-generated street maps," *IEEE Pervasive Computing*, vol. 7, no. 4, pp. 12–18, 2008.