International Journal of Multimedia and Ubiquitous Engineering
Vol. 9, No. 11 (2014), pp. 111-122
http://dx.doi.org/10.14257/ijmue.2014.9.11.11

# A Proactive Parallel Complex Event Processing Method for Large-Scale Intelligent Transportation Systems

Yongheng Wang and Xiaoming Zhang

*College of Computer Science and Electronics Engineering, Hunan University*
*yh.wang.cn@gmail.com, zhangxm19712003@yahoo.com.cn*

***Abstract***

*Intelligent Transportation Systems (ITS) is one of the important application areas of the Internet of Things (IoT). The key issue is how to process the huge events generated by IoT system to support ITS. In this paper a proactive parallel complex event processing method is proposed for congestion control in large-scale ITS. A Bayesian model averaging method is used to obtain accurate predictions under different event context. Based on the predictive analysis, a parallel Markov decision processes model is designed to support decision making for large-scale ITS. An optimized parallel policy iteration algorithm is proposed based on state partition and policy decomposition. The experimental evaluations show that this method has good accuracy and scalability when used to process congestion control in large-scale ITS.*

***Keywords:*** *Internet of Things; Complex Event Processing; Predictive Analytics; Markov Decision Processes*

## 1. Introduction

Internet of Things (IoT) can be defined as a dynamic global network infrastructure where objects in real world are connected into the internet through standard and interoperable communication protocols based on various kinds of sensors such as RFID and GPS. The events that generated by devices directly are called primitive events. Usually there is limited semantic information inside primitive events. In real application, people mainly concern about 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 only complex events like "the car leaves the garage" are events that people really concern. We can get these complex events by combining many primitive events according to some patterns. Complex Event Processing (CEP) [1] is used to process huge primitive events and get valuable high-level information from them.

In some IoT application, actions can be executed on some devices to change the state of the system. The traditional event processing methods are called reactive methods since the actions are triggered by the state change of the system. A proactive event processing system has the ability to mitigate or eliminate undesired future events, or to identify and take advantage of future opportunities, by applying prediction and automated decision making technologies [2]. For example, in ITS we can predict some congestion states according to the current state and historical data and then take some actions to avoid some future congestion states. Predictive Analytics (PA) is the technology that predicts future events through the analysis of historical events. CEP and PA is studied widely but currently there are few papers about how to integrate these two technologies to develop proactive event-driven system. As an optimal decision-making process for stochastic dynamic systems, Markov Decision Processes (MDP) is a reasonable choice for proactive event processing but there are few

ISSN: 1975-0080 IJMUE

papers about how to integrate MDP into proactive event processing and how to process the huge state space of MDP in this area.

In this paper, we propose a proactive parallel complex event processing architecture and method (PPCEP) for large-scale ITS. Based on probabilistic complex event processing, this method uses a Bayesian model averaging model to predict future events and system states. A novel parallel MDP method based on state partition and concurrent actions is proposed to support proactive event processing. An optimized parallel policy iteration algorithm is proposed. The method is evaluated for a simulated intelligent transportation system.

## 2. Related Works

Complex event processing recognizes complex events from a set/sequence of raw events by continuously monitoring the message flow. Etzion *et al.*, described the basic concept and architecture of complex event processing in their book [3]. Event Processing Agent (EPA) is a component that takes a set of input events as input and output a set of complex events according to some logic. Event Processing Network (EPN) is a network composed of EPAs, event producers, and event consumers linked by channels. Luckman first introduced the event processing network in the field of modeling [1]. Based on his idea, the conceptual model of EPN was further elaborated by Sharon and Etzion [4]. Most of the current CEP methods use fixed data structure such as tree, directed graph and Petri-Net. SASE [5] is a high performance query-plan based complex event processing method that uses Nondeterministic Finite Automaton (NFA) and Active Instance Stacks (AIS). Recently some improved versions of SASE are proposed [6].

In order to support imprecise event, many CEP engines use variants of probabilistic graphical models. In the work of Richardson *et al.*, [7], Markov Logic Networks (MLNs) have been used to handle uncertainty in complex event processing. Recently some work about processing complex events in probabilistic event streams based on NFA is proposed [8]. Compared with our work, the current CEP methods are reactive and are not integrated with PA to support proactive applications.

The predictive analytics methods based on complex events can predict the future values of some attributes of the monitored system based on the historical data. As a valid model for uncertain knowledge representation and inference, Bayesian Network (BN) is widely used in predictive analytics. Castillo et al. proposed a Dynamic Bayesian Network (DBN) model that can predict spatio-temporal events in sensor networks [9]. In the work proposed by Pascale *et al.*, an adaptive Bayesian network was used in traffic flow prediction [10]. In the work of Sun *et al.*, channel quality prediction in cognitive radio networks is supported using Bayesian Networks [11]. By averaging over many different competing models, Bayesian model averaging (BMA) incorporates model uncertainty into conclusions about parameters and prediction. Recently there are some work about BMA and model selection [12, 13], but there are few papers about using BMA in CEP applications.

Proactive event based systems have been studied by many researchers recently. Some examples include proactive management of transport processes [14] and proactive application event notification in sensor network [15]. Engel *et al.*, proposed a proactive event processing framework based on CEP, PA and MDP [2, 16]. In their work, two new types of agents are added: predictive agents which can derive future uncertain events based on prediction models, and proactive agents which can choose the best proactive action that should be taken. Compared with our work, the work of Engel *et al.*, lacks of implementation detail and is not optimized for large scale IoT application.

 

MDP has been studied for many years and recently some variants of MDP emerged. When using in large-scale IoT application, the main challenge is the combination explosion problem caused by large state space and action space. The current research on this problem can be classified into two directions. In the first direction the problem is simplified by using the information from the application domain [17]. In the second direction, approximate methods are used for large scale MDP [18]. When applying to large-scale IoT, MDP has larger state space and some new properties which bring new challenges for model design and calculation. Compared with our work, the current MDP methods are not optimized for large state space in large-scale proactive event-based system.

## 3. Proactive Complex Event Processing Method

### 3.1. System Architecture

The architecture of our work is shown in Figure 1. We extended the EPN framework into Probabilistic Event Processing Network (PEPN), which can process probabilistic raw events to get probabilistic complex events, such as the running path of a vehicle. Based on the probabilistic complex events, the PA component with BMA can predict the congestion states of the system. Complex events are saved into event database and the historical events can be used by the machine learning methods in PA for more accurate predict. In the proactive executor component, the decision maker with PPMDP selects appropriate actions according to the predicted states and assigns corresponding proactive agents (PRA) to execute the actions.

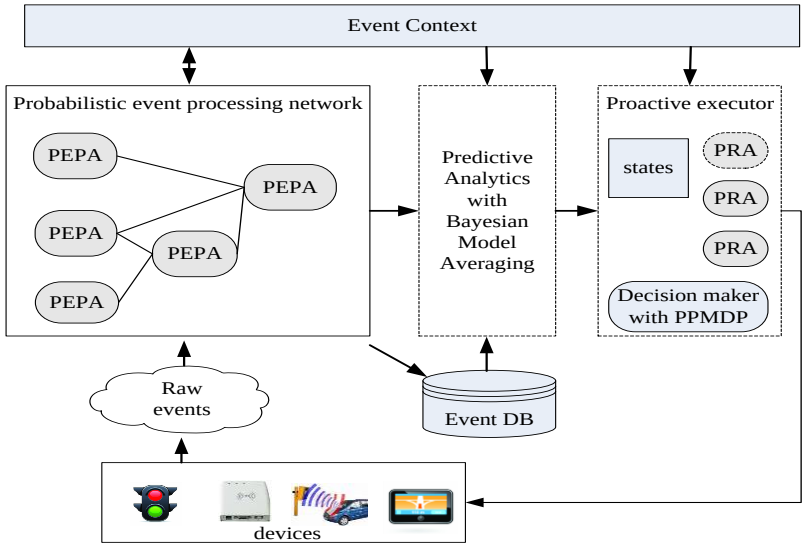


**Figure 1. System Architecture**

### 3.2. Predictive Analytics using Bayesian Model Averaging

#### 3.2.1. Basic Predictive Analytic Method

An event context is a specification of conditions that groups event instances so that these instances can be processed in a related manner. The event context assigns each event instance to one or more context partitions. Our basic PA model is a context-aware model since it acts differently under different context.

We consider a traffic network as a Bayesian network in which the state of a node i at time t is affected by many other nodes before time t. Let $f_{i,t}$ represent the flow state of (i,t) and pa(i,t) represent the parent nodes of (i,t) in the Bayesian network. $N_P$ denotes the number of nodes in pa(i,t). The set of flow states for pa(i,t) is $F_{pa(i,t)}=\{f_{j,s} : (j,s) \in pa(i,t)\}$. According to the BN theory, the joint distribution of all nodes in the flow states network can be expressed as:

$$p(F) = \prod_{i,t} p(f_{i,t} \mid F_{pa(i,t)}, C_{i,t}) \quad (1)$$

where $C_{i,t}$ is the context of (i,t). The conditional probability $p(f_{i,t}| F_{pa(i,t)},C_{i,t})$ can be calculates as:

$$p(f_{i,t} \mid F_{pa(i,t)}, C_{i,t}) = p(f_{i,t}, F_{pa(i,t)}, C_{i,t}) / p(F_{pa(i,t)}, C_{i,t}) \quad (2)$$

The $p(f_{i,t}, F_{pa(i,t)}, C_{i,t})$ and $p(F_{pa(i,t)}, C_{i,t})$ are difficult to calculate. Based on the fact that many nodes share the same context and there are only few context types, we partition the events according to context and train a model for every context. During real time predictive analysis, if the nodes share the same context, we use the corresponding model. Otherwise, we use Bayesian model averaging. For a given context, we model the joint distribution $p(f_{i,t} , F_{pa(i,t)})$ with GMM like the work of [10]:

$$p(f_{i,t}, F_{pa(i,t)}) = \sum_{m=1}^{M} \alpha_m g_m(f_{i,t}, F_{pa(i,t)} | \mu_m, C_m) \quad (3)$$

where M is the nodes number and $g_m(\cdot|\mu_m,C_m)$ is the m-th Gaussian distribution with $(N_P + 1) \times 1$ vector of mean values $\mu_m$ and $(N_P + 1) \times (N_P + 1)$ covariance matrix $C_m$. EM algorithm is used to infer parameters $\{\alpha_m, \mu_m, C_m\}_{m=1}^{M}$ from the historical data. Once $p(f_{i,t} , F_{pa(i,t)})$ has been obtained, the conditional distribution $p(f_{i,t}| F_{pa(i,t)})$ can be derived and the estimate $\hat{f}_{i,t}$ can be calculated from $F_{pa(i,t)}$ using the minimum mean square error (MMSE) method.

### 3.2.2. Bayesian Model Averaging

Assume there are K context types in data D where each context has a model, the model ensemble posterior distribution of a quantity Q is:

$$p(Q \mid D) = \sum_{k=1}^{K} p(Q \mid M_k, D) p(M_k \mid D) \quad (4)$$

where $p(Q, M_k, D)$ is the posterior distribution of quantity Q under model $M_k$ and data D. $p(M_k|D)$ is the posterior model probability (model weight). This is a linear combination of the basic models where the model weight parameters are decided by the posterior model probabilities. The $m_k$ parameters of model $M_k$ are represented by a vector $\Theta_k=(\theta_{1k}, \theta_{2k},.., \theta_{mk})$. D is the observation on the $m_k$ parameters which is represented by a vector $D=(d_1,d_2,\ldots,d_n)$. According to Bayesian theory we get:

$$p(\Theta_k \mid D, M_k) = \frac{p(D \mid \Theta_k, M_k) p(\Theta_k \mid M_k)}{p(D \mid M_k)} \quad (5)$$

The marginal likelihood $p(D|M_k)$ is called model evidence:

$$p(D \mid M_k) = \int p(D \mid \theta_k, M_k) p(\theta_k \mid M_k) d\theta_k \quad (6)$$

The posterior probability $p(M_k,D)$ can be calculated by:

$$p(M_k \mid D) = \frac{p(D \mid M_k) p(M_k)}{p(D)} \quad (7)$$

 

where $p(M_k)$ is the prior distribution of model k. Usually we assume all models have the same prior probability if we have no preference for them. The quality of different model is mainly determined by $p(D|M_k)$ since data distribution p(D) is not related to models. Therefore Equation (7) can be transformed to:

$$p(M_k \mid D) = \frac{p(D \mid M_k)}{\sum_{i=1}^{K} p(D \mid M_i)} \tag{8}$$

The integration in equation (6) is not easy to be calculated directly. We use cross validation distribution to estimate the model evidence $p(D|M_k)$:

$$\hat{p}(D \mid M_k) = \prod_{i=1}^{n} p(y_i \mid M_k), (x_i, y_i) \in D \tag{9}$$

where $p(y_i|M_k)$ can be calculate by:

$$p(y_i \mid M_k) = \int p(y_i \mid \theta_k, M_k)\, p(\theta_k \mid M_k) d\theta_k \tag{10}$$

If the model is relatively complex, equation (10) is still difficult to be calculated directly. We use MCMC method for approximate calculation. A series of independent samples $\theta_k^{(t)}$ : t=1,…,T of $\theta_k$ are sampled from the distribution $p(\theta_k |M_k)$. Then equation (10) can be approximated by:

$$\hat{p}(y_i \mid M_k) = \frac{1}{T} \sum_{t=1}^{T} p(y_i \mid \theta^{(t)}{}_k, M_k) \tag{11}$$

In order to find independent series of samples for MCMC, we use Markov chain $\theta_0,\theta_1,\ldots$, in which every $\theta_i$ depends on $\theta_{i-1}$ only. When some condition is satisfied, the series will converge to static distribution $p(\theta)$ after m iterations despite the original value of the series. Then the samples generated by iterations since m+1 can be used as independent samples for MCMC.

### 3.3. Decision Making with Proactive Parallel MDP

**Definition 1 (Proactive Parallel MDP)**: A proactive parallel MDP (PPMDP) is represented by $\langle I, S, \vec{A}, P, R, S_p, C \rangle$, where *I* is the set of agents that process actions proactively, and *S* denotes the set of system states with a special initial state $S_0$. $\vec{A} = \times_{i \in I} A_i$ denotes the set of actions in which $A_i$ is the action from the i-th agent. *P*: $S \times A \times S \rightarrow [0, 1]$ is the set of state transformation function, where P(s,α,s') denotes the probability that state s transforms to state s' with the execution of action α. *R*: $S \rightarrow Re$ is the set of reward functions (*Re* means the set of real numbers). Every $(s,\alpha) \in S \times A$ satisfies $\sum_{s' \in S} p(s, \alpha, s') = 1$. $S_p$ is the set of future states predicted by PA components and C is the context of events.

The system starts from an original state and recursively selects a set of actions executed by agents parallel according to the predictive analysis result. The key issue of PPMDP is to find a policy $\pi: S \times S_p \rightarrow \vec{A}$ which reflect the current state and predictive state to a set of actions.

In traditional MDP, the selection of policy is based on the following equations (value function and policy update equation):

$$V_\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s' \mid s, \pi(s)) V_\pi(s') \tag{12}$$

$$\pi^*(s) = \arg\max_{a \in A}(r(s,a) + \gamma \sum_{s' \in S} P(s' \mid s,a) V_\pi(s')) \quad (13)$$

where γ is the decay factor. This method is inefficient for large-scale intelligent transportation system because there are too many possible states and actions. PPMDP uses an optimized policy iteration algorithm and executes actions parallel based on the following observations:

**Observation 1**: The state in large-scale transportation IoT can be partitioned and each group is related to a sub state of the system. We need to consider the sub states of the system but not the state of every vehicle.

**Observation 2**: In large-scale transportation IoT, the state of a node depends on the states of its neighbors (the neighbors here means the nodes that are linked with the current node within certain distance).

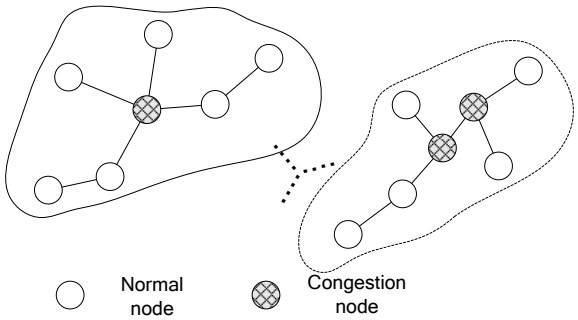


**Figure 2. Network Partition based on Congestion Nodes**

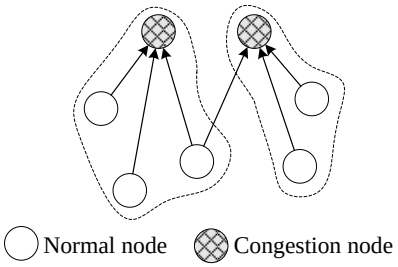


**Figure 3. Bayesian Network Partition**

With the result of predictive analytic, the network can be partitioned based on the congestion nodes as shown in figure 2. One or more nodes which have high congestion level (estimated by PA component) are selected as the center of the group. The nodes have distance smaller than a threshold value D from the center are partitioned into the group. Assume there are N nodes: $J=\{j_1,j_2,\ldots,j_N\}$. We partition the vehicle flow value into K levels and represent it as $CS=\{c_1,c_2,\ldots,c_K\}$. Then the state in PPMDP can be represented as $s_t=<s_{t1},s_{t2},\ldots,s_{tN}>$ where $s_{ti} \in S$ is the congestion state of node $j_i$ at time t and $s_i \in$ CS. A policy $\pi_{it}=\{a_{it1},a_{it2},\ldots,a_{itM}\}$ where $a_{itk}$ is an action aim to reduce the congestion of group k at time t by guiding some vehicles to change their paths. In this paper we assume the distance between any two group centers is large enough which means the sub-actions are independent and they can be executed parallel. Let G=(V,E) represent the set of sub graphs where V is the set of vertexes for a sub graph and E is the set of edges.

**Definition 2 (congestion state decomposition)**: The congestion state of the system C(*S*) can be decomposed by:

$$C(S) = \sum_{i=1}^{|G_s|} C(S_i) = \sum_{i=1}^{|G_s|} \sum_{j=1}^{|C_i|} C(S_{i,j}) \quad (14)$$

where $G_s$ is the set of state partitions and $C_i$ is the set of predicted congestion nodes in partition i.

**Definition 3 (neighbor nodes)**: For any node $i \in V$, the neighbor nodes are $N(i)=\{j \in V \mid d_{i,j} \leq k\}$, where $d_{i,j}$ is the distance of node i and j, and k is a threshold value.

**Definition 4 (policy decomposition)**: A policy π can be decomposed by:

 

$$\pi = \sum_{i=1}^{|G_S|} \pi_i = \sum_{i=1}^{|G_S|} \sum_{j=1}^{|C_i|} \pi_{N(j)} \quad (15)$$

**Definition 5 (state transformation decomposition)**: The state transformation possibility can be decomposed by:

$$P(s' \mid S, \pi) = \sum_{i=1}^{|G_S|} P(s_i' \mid S_i, \pi_i) = \sum_{i=1}^{|G_S|} P(s_i' \mid S_i, \{\pi_{N(j)} \mid j \in C_i\}) \quad (16)$$

**Definition 6 (reward function)**: The reward function of state s can is defined as:

$$R(s) = \alpha C(s) + (1 - \alpha) L(s) \quad (17)$$

where L(s) denotes the total path length of all vehicles and α is a weight factor.

Definition 6 means we want to minimum the total congestion and total path length. Now the key issue is how to find the optimized policy $\pi^*$ for each sub-state. We partition the Bayesian network in the PA model according to congestion nodes as shown in figure 3. All nodes that affect the congestion node are partitioned into the same group. Overlapped nodes are partitioned into groups on average. In order to reduce the congestion, in each sub-action the normal nodes in each group are redirected to other nodes outside the groups. The reward of subsequent sub-states can be calculated using the PA method. The policy iteration algorithm is shown in Figure 4.

```
function policy_iteration(ppmdp) returns a policy
  Gs←state_partition(S)
  for each si in Gs do parallel
    πsi←sub_policy_iteration(ppmdp, si)
  π←combine_policy(set of πsi)
return π
function sub_policy_iteration(ppmdp, s) returns a sub policy
  repeat
    unchanged? ←true
    for each state si in s
      BGsi←BN_partition(si)
      for each gj in BGsi do parallel
        ak←find_optimized_sub_actions(gj)
      as←find_ optimized _actions(set of ak)
      if Σ_{si'} P(si' | si, as)U[si'] > Σ_{si'} P(si' | si, πs[si])U[si'] then
        πs[si]←as
        unchanged? ←false
  until unchanged?
  Return πs
```

**Figure 4. Policy Iteration Algorithm**

The state_partition function partition the traffic network according to Figure 2 and sub-policies can be found based on this partition. The BN_partition function partition the Bayesian network according to Figure 3 to get the original states of sub-networks. The find_optimized_sub_actions function finds a series of local optimized actions using

stochastic gradient descent method. If there is no overlap in the partitions of BN_partition, the local optimized actions can be composed directly to get the total optimized policy in find_ optimized _actions. Otherwise, the find_ optimized _actions iterate select the most congested node according to PA and find an optimized sub-policy to reduce the congestion level using stochastic gradient descent method until no congested node can be found or no optimized sub-policy can be found.

## 4. Experimental Evaluations

### 4.1. System Implementation

Our method is implemented based on the Berkeley Data Analytics Stack (BDAS) which is an open source software stack that integrates software components to make sense of big data. The system implementation architecture is shown in Figure 5. The historical event data is stored in the HDFS. The PCEP (Probabilistic CEP) engine is created based on Spark Streaming. The filter and window function of Spark Streaming can help to process complex event patterns. The PA engine is created based on Spark and Spark Streaming. This architecture can support large-scale historical data processing and with the help of in-memory file system we can get high performance.

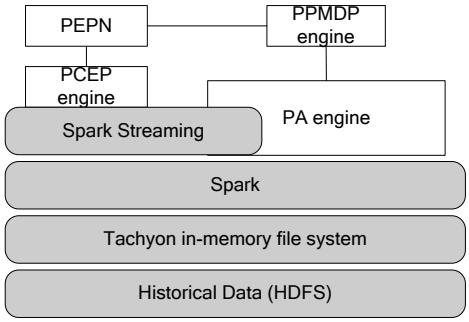


**Figure 5. System Implementation Architecture**

**Table 1. Deviation of two Methods. The Average Percent is Calculated by (average deviation)/(average observed vale)*100%**

| | *deviation* | |
|---|---|---|
| | *ABN* | *PPCEP* |
| ***Max*** | *286* | *109* |
| ***Min*** | *7* | *11* |
| ***Average*** | *111.6* | *56.4* |
| ***Average percent*** | *14.53%* | *7.34%* |

We developed a transportation IoT simulation system based on the road traffic simulation package SUMO. SUMO supports "induction loops" which can detect vehicles that pass corresponding areas. Getting the location of each vehicle is also supported. We use the TraCI interface of SUMO to get the induction loop variables and vehicle location variables, and then use these variables to simulate virtual RFID and GPS readers. In the experiment we selected 84 junctions from the map and set 80 thousand vehicles. In order to simulate real traffic system, a set 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, hospital, etc., with corresponding probabilities. Based on this simulation system we evaluated the precision and performance of our method. We used 4 servers with Xeon E3 processor and 16GB memory as data processing servers and the operating system is Ubuntu 12. Another PC with 4GB memory is used to run SUMO.

### 4.2. Experimental Evaluations for Proactive Complex Event Processing

We first run the simulation for many times to get the historical data of the vehicle paths. The conditional probability table is created from the historical data. In the first experiment we compared the accuracy of our PA method with the work of Xing *et al.*, [12] which uses

 

Adaptive Bayesian Network (ABN). The result is shown in Figure 6 and Table 1. From the figure we can see the accuracy of our PA method is better than traditional method ABN. The reason is that we use BMA to support better PA under different event context.

In the next experiment, we evaluated the mean congestion level of the system and the result is shown in Figure 7. We partitioned the congestion value into 10 levels where "0" means no congestion and "9" means the highest congestion. Since we have not found other methods have the same function with ours, our method is only compared with the default simulation system. From the figure we can see the mean congestion level reduced obviously when our method is used. The reason is that our method can predict the congestion state and take some actions to avoid it proactively. We can also find the reduction of congestion level is more obvious when the congestion level becomes high. The reason is that the system can redirect more vehicles to light loaded nodes in such circumstance.

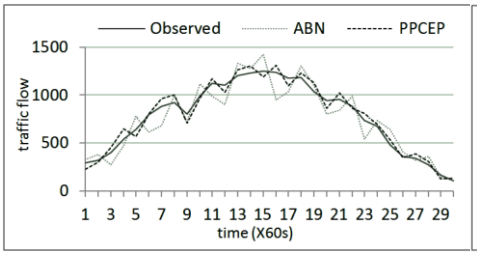


**Figure 6. PA Accuracy for a Typical Node**

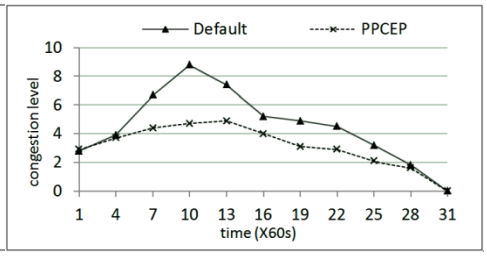


**Figure 7. Mean Congestion Level Over Time**

In the next experiment, we evaluated the performance of our method with different server number and data size. The number of important node is fixed at 25 and the constructing time of Bayesian model is not included. The result is shown in Figure 8. From the figure we can find the average running time for decisions increases when the vehicle numbers becomes larger. The reason is that more vehicles need more sub-actions which increase the complexity of the algorithm. We can also find the performance for 4 servers is higher than 2 servers and the running time of the former increases more slowly. The reason is that the parallel method in our system can take advantage of multiple servers.

We also evaluated the performance of our method with different server number and average congestion node number. The result is shown in Figure 9. The vehicle number is fixed at 80 thousand. We can find the running time increases obviously when the important node number becomes larger. The reason is that more important nodes means more sub-states in the parallel MDP which increases the complexity of the algorithm.

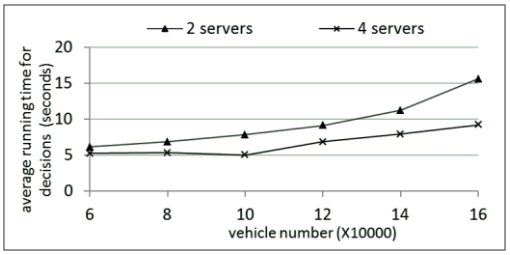


**Figure 8. Performance for Different Vehicle nNumber**

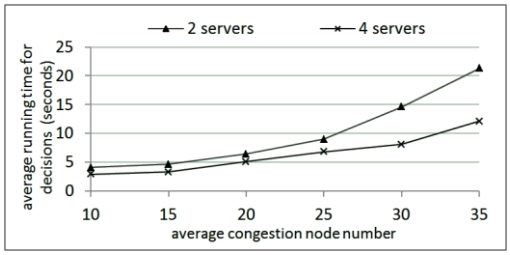


**Figure 9. Performance for Different Congestion Node Number**

From all the experiments we can see PPCEP can support proactive complex event processing well in large-scale ITS applications. Based on BMA, the PA method can get higher accuracy than traditional methods. The PPMDP model has obvious effect for congestion control in large transportation IoT. The performance of the PPMDP decreases when the vehicle number and important node number becomes large but we can get better performance with more servers.

## 5. Discussions and Conclusion

In this paper we proposed a proactive complex event processing method using parallel MDP for large-scale ITS. A Bayesian model averaging method is proposed to support accurate PA for different event context. Based on state partition and policy decomposition, an optimized parallel policy iteration algorithm for MDP is proposed to support large-scale IoT. The experimental evaluations show that this method has good accuracy and scalability when used for large-scale ITS.

The performance and scalability of PPCEP still need to be improved. The PA method is not efficient enough if there are too many nodes. In the parallel MDP, the sub-states and sub-action space can also be very large which makes the algorithm inefficient. We need to develop new parallel or approximation algorithms to improve the performance.

## Acknowledgements

This work is supported by the "Study of Proactive Complex Event Processing for Large-scale Internet of Things (61371116)" project of National Natural Science Foundation of China and the "Context-aware and proactive complex event processing for large scale internet of things (13JJ3046)" project of Hunan Provincial Natural Science Foundation of China.

## References

[1] D. C. Luckham, "The power of events: an introduction to complex event processing in distributed enterprise systems", **(2002)**; Addison Wesley, Boston.
[2] Y. Engel and O. Etzion. "Towards proactive event-driven computing. Proceedings of Fifth ACM International Conference on Distributed Event-Based Systems", **(2011)**; New York.
[3] O. Etzion and P. Niblett, "Event Processing in Action, Manning Publications", **(2010)**.
[4] G. Sharon and O. Etzion, "Event-processing network model and implementation", IBM System Journal, vol. 47, no. 2, **(2008)**, pp.321-344.
[5] E. Wu, Y. Diao and S. Rizvi, "High-performance complex event processing over streams", Proceedings of ACM SIGMOD international conference on Management of data, **(2006)**; Chicago IL, USA.
[6] H. Zhang, Y. Diao and N. Immerman, "Recognizing Patterns in Streams with Imprecise Timestamps", PVLDB, vol. 3, no. 1, **(2010)**, pp. 244-255.
[7] M. Richardson and P. Domingos, "Markov logic networks, Machine Learning, vol. 62, no. 1-2 **(2006)**, pp. 107-136.
[8] C. Xu, S. Lin and W. Lei, "Complex Event Detection in Probabilistic Stream", Proceedings of the 12th International Asia-Pacific Web Conference, **(2010)**, pp. 361-363.
[9] E. Castillo, J. M. Menéndez and S. Sánchez-Cambronero, "Predicting traffic flow using Bayesian networks", Transportation Research Part B: Methodological, vol. 42, **(2008)**, pp. 482-509.
[10] A. Pascale and M. Nicoli, "Adaptive Bayesian network for traffic flow prediction", Proceedings of the Statistical Signal Processing Workshop (SSP), IEEE, **(2011)**, pp.177-180.
[11] S. Sun, C. Zhang and G. Yu, "A Bayesian Network Approach to Traffic Flow Forecasting, Intelligent Transportation Systems", vol. 7, **(2006)**, pp. 124-132.
[12] N. Tawara, T. Ogawa, S. Watanabe and T. Kobayashi, "Fully Bayesian inference of multi-mixture Gaussian model and its evaluation using speaker clustering", Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, **(2012)**, pp. 5253-5256.
[13] Y. Zhou, A. M. Johansen and J. A. D. Aston, "Bayesian model comparison via path-sampling sequential Monte Carlo", Proceedings of IEEE Workshop on Statistical Signal Processing, **(2012)**, pp. 245-248.

 

[14] S. Dolev, M. Kopeetsky and A. Shamir, "RFID authentication efficient proactive information security within computational security", Theory of Computing Systems, **(2011)**, pp. 1–18.
[15] T. Kunz and R. Alhalimi, "Energy-efficient proactive routing in MANET: Energy metrics accuracy", Ad Hoc Networks, vol. 8, no. 7, **(2010)**, pp. 755–766.
[16] Y. Engel, O. Etzion and Z. Feldman, "A Basic Model for Proactive Event-Driven Computing", Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, **(2012)**; Berlin, Germany.
[17] Q. S. Jia, "On State Aggregation to Approximate Complex Value Functions in Large-Scale Markov Decision Processes", IEEE Transactions on Automatic Control, vol. 56, no. 2, **(2011)**, pp. 333- 344.
[18] M. H. Veatch, "Approximate linear programming for average cost MDPs", Mathematics of Operations Research, vol. 38, no. 3, **(2013)**, pp. 535-544.

## Authors

**Yongheng Wang**, was graduated from National University of Defense Technology. He is now a lecturer in College of Computer Science and Electronics Engineering, Hunan University. His main research area is complex event processing and data mining.

为方便组委会联系，请提供 2 位作者信息。

| | | | |
|---|---|---|---|
| 论文题目 | A Proactive Parallel Complex Event Processing Method for Large-Scale Intelligent Transportation Systems | | |
| 所属主题 | 智慧城市 | | |
| **第一作者** | | | |
| 姓名 | 王永恒 | 职称/学位 | 讲师/博士 |
| 单位 | 湖南大学信息科学与工程学院 | 邮编 | 410082 |
| 地址 | 湖南长沙岳麓山 | | |
| 电话 | | 手机 | 13975813465 |
| Email | Yh.wang.cn@gmail.com | | |
| **第二作者** | | | |
| 姓名 | 张小明 | 职称/学位 | 副教授/博士 |
| 单位 | 湖南大学信息科学与工程学院 | 邮编 | 410082 |
| 地址 | 湖南长沙岳麓山 | | |
| 电话 | | 手机 | 13687391318 |
| Email | zhangxm19712003@yahoo.com.cn | | |