Intelligent Transportation Control based on Proactive Complex Event Processing

Yongheng Wang, Shaofeng Geng and Qian Li

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

Abstract. Complex Event Processing (CEP) has become the key part of Internet of Things (IoT). Proactive CEP can predict future system states and execute some actions to avoid unwanted states which brings new hope to intelligent transportation control. In this paper, we propose a proactive CEP architecture and method for intelligent transportation control. Based on basic CEP technology and predictive analytic technology, a networked distributed Markov decision processes model with predicting states is proposed as sequential decision model. A Q-learning method is proposed for this model. The experimental evaluations show that this method works well when used to control congestion in in intelligent transportation systems.

1 Introduction

Internet of Things (IoT) technology provides dynamic global network infrastructure where objects in real world can be connected into the internet based on various kinds of sensors. Recently, with the rapid development of information and communication technologies, building an IoT application is no longer difficult. The main problem that people concern now is how to process the huge events produced by IoT applications.

Complex Event Processing (CEP) [1] is the technology to accept as input a stream of primitive events that happen in the external environment, interpret and combine them to identify higher level composite events. The high level events can be sent to the controller of the system which can execute some actions to determine the overall system's behavior. CEP has been used in many areas, such as sensor networks for environmental monitoring [2], continuous analyzing of stocks to detect trends [3], etc.

The traditional kind of CEP is called reactive event processing which means agents are triggered by events and react to change states of the system. Another kind of method is called proactive event processing, which means 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 [4]. IoT with proactive CEP provides a new way to improve Intelligent Transportation Systems (ITS) or transportation IoT. For example, we can predict some future congestion states based on the current state and historical data, and then take some actions to avoid some future congestion states.

To support proactive CEP, besides the core CEP technology, we also need Predictive Analytics (PA) and sequential decision making technology. PA is the technology that can be used to predict future events or system states through the analysis of historical events or states. As an optimal sequential decision making process for stochastic dynamic systems, Markov Decision Processes (MDP) becomes a reasonable choice for sequential decision making in proactive event processing. However, currently there are still some challenges to support proactive CEP. In the PA part, we need to develop models and algorithms that have high accuracy and good performance for IoT applications. Sequential decision making with MDP for proactive CEP is a more challenging technology since we need to handle not only huge joint state and action space, but also predicting state which is not included in standard MDP.

In this paper, we propose a proactive CEP architecture and method (Pro-CEP) for intelligent transportation control. The proactive CEP architecture contains probabilistic event processing network, predictive analytic component and decision making component. An extended Networked Distributed Markov Decision Processes (ND-MDP) model with predicting states and corresponding Q-learning method are also proposed. The Pro-CEP method is evaluated in a simulated transportation IoT and the results show it works well when processing proactive congestion control in transportation IoT.

2 Related Works

CEP recognizes complex events from a set/sequence of raw events by continuously monitoring the data flow.

This project is sponsored by the National Natural Science Foundation of China, under grant 61371116.

Luckham and Etzion introduced the basic concepts and architecture of CEP [1,5]. Event Processing Agent (EPA) is a component that takes a set/sequence of primitive 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.

Some research work about proactive event based systems has been proposed, such as proactive management of transport processes and proactive application event notification in sensor network. Engel et al. proposed a proactive event processing framework based on CEP, PA and MDP [4]. In their framework, 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. We used the similar framework but we investigated novel predictive analytic and decision making method for transportation IoT.

MDP has been studied for many years and recently several variants of MDP emerged. If we use MDP in proactive CEP, one of the challenges is the NP-complete problem caused by large state space and action space. Basically there are two research directions on this problem. In the first direction the problem is simplified by using the information from the application domain. For example, Jia et al. used state aggregation methods [6]. It is difficult to find a common solution in this direction since these methods are related to specific application domain. In the second direction, approximate methods are used for large scale MDP. For example, Cao et al. used dynamic programming based on events [7]. The key idea in these methods is to approximate the value function during the optimization process.

In many IoT applications, the agent does not have complete model of the environment and does not know the exact reward function. Therefore reinforcement learning becomes a reasonable choice. Zhang et al. proposed a model-free, scalable learning approach that synthesizes multi-agent reinforcement learning and distributed constraint optimization (DCOP) [8]. Their method distributes the learning of the joint policy and employs DCOP techniques to coordinate distributed learning to ensure the global learning performance. In order to control traffic congestion, Fares et al. proposed a system that introduces a new microscopic framework at the network level based on collaborative MDP modeling and an associated cooperative Q-learning algorithm [9]. Compared to our work, the current MDP models and learning methods try to find optimal policy based on current state or history state but not predicted future state.

3 System Architecture

The goal of our system is to control traffic congestion intelligently using proactive CEP in transportation IoT. There are various sensors in the system to get the position of vehicles, such as induction loops, RFID, GPS, etc. There are also other sensors to get environment information, such as temperature and humidity sensor, wind sensor, etc. The traffic control agents try to reduce potential congestion through controlling the traffic light.

The system architecture is shown in figure 1. In order to process uncertain event, we extended the EPN framework into Probabilistic Event Processing Network (PEPN) which is composed of many connected Probabilistic Event Processing Agents (PEPA). Raw events are generated by sensors or other information system (such as weather broadcast system). The imprecise raw events are processed by PEPN to get probabilistic complex events, which represent some high level semantic such as the position or running path of a vehicle. Based on the run time prediction of congestion state by predictive analytic component, the decision maker selects optimal actions using MDP model and assigns corresponding proactive agents (PRA) to execute the actions.

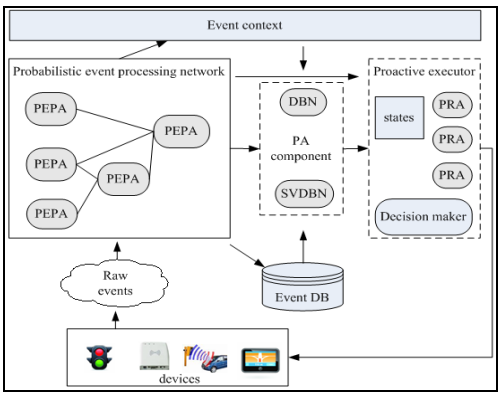

Figure 1. System architecture.

Definition 1 (primitive event): A primitive event means an atomic occurrence of interest in time which is represented as <A, T, P_r> where A is the set of attributes, T is the timestamp that the event occurs and P_r is the probability value of the event.

Definition 2 (complex event): Complex event is a combination of primitive events or complex events by some rule which is represented as <E, R, T_s, P_r> 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 and P_r is the probability value.

Event type is a specification for a set of events that have the same semantic intent and the same structure. Every event is an instance of an event type. An event type can represent either primitive events or complex events. The main complex event patterns include ALL, ANY, COUNT, SEQ, etc. The detailed meaning of the patterns is described in [5]. For example, in the area of transportation system, 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 running path of a vehicle.

We have developed a high-performance Distributed Probabilistic Complex Event Processing method (DPCEP). DPCEP extends SASE [10] which is 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.

In our work we only consider event matching EPA which is the most important one. PEPA is the extension of EPA that supports probabilistic event processing and we also call it probabilistic CEP engine. Multiple PEPAs are connected by event channel to create an EPN. An event channel is a processing element that receives events from one or more source processing elements, makes routing decisions, and sends the input events unchanged to one or more target processing elements in accordance with these routing decisions. The detailed information about DPCEP is described in [11]. The predictive analytic component is implemented using Adaptive Bayesian Network (ABN) based on the work of Pascale et al. [12].

4 Sequential Decision Making

Definition 3 (ND-MDP$^+$): A Networked Distributed Markov Decision Processes with future states (ND-MDP$^+$) is defines as $\langle I, S, \bar{S}, \vec{A}, P, R\rangle$, where I is a finite set of agents. S is a finite set of states, with distinguished initial state s_0. $\bar{S}$ is a finite set of future states which can be predicted from historical states. $\vec{A}=\times_{x\in I} A_i$ is a set of joint actions, where A_i is the set of actions for agent *i*. $P: S\times\vec{A}\to S'$ is the state transition function, defining the distributions of states that result from starting in a given state and a joint action by agents. $R: S\times\vec{A}\to\Re$ is the reward function for the set of agents for each set of joint actions and each state.

Being different from standard ND-MDP, ND-MDP$^+$ contains a future state set $\bar{S}$ which can be predicted using the predictive analytic method. The proactive event processing system starts from predicting future state and select a joint action to be executed by agents according to the joint policy. The execution of the joint action maximizes the total expected reward for both the future state and the current state.

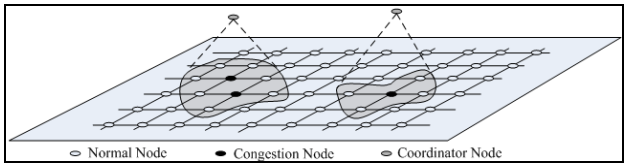

Figure 2. Transportation network partition.

In order to resolve the congestion control problem with ND-MDP$^+$, we use a dynamic network partition method to partition the traffic network as shown in Figure 2. We partition every predicted congestion node and it neighbor nodes into a group. If two groups are overlapped, we simply combine them into one group. Based on the assumption that congestion nodes are sparse, we can get a few separated groups. A coordinator node is selected for every group. We use *G* to represent the partitioned hyper graph and every $g\in G$ connects all the nodes in a group.

We use the reinforcement learning method Q-learning to learn the joint policy. The Q-function $Q(s,\bar{s},a)$ represents the reward of executing joint action a under state s and predicting state $\bar{s}$. The optimal joint policy π^* can be derived from $Q(s,\bar{s},a)$ by

$$\pi^*(s) = \arg\max_{a\in A} Q^*(s,\bar{s},a) \tag{1}$$

The extended Q-learning can be described by the following equation:

$$\begin{aligned} Q(s^t,\bar{s}^t,a^t) &= (1-\alpha)Q(s^t,\bar{s}^t,a^t) \\ &+ \alpha[(1-\beta)r_s^t + \beta r_{\bar{s}}^t \\ &+ \gamma \max_a Q(s^{t+1},\bar{s}^{t+1},a)] \end{aligned} \tag{2}$$

where $\alpha\in(0,1)$ is the learning rate, $\gamma\in[0,1]$ is the discount factor, r^t is the reward executing action a on state *s*, $Q(s^t,\bar{s}^t,a^t)$ represents the Q-value of executing action a for joint state $(s,\bar{s})$ at time t. The joint reward r^t contains two parts where r_s^t means the reward from the current state s to its next state and and $r_{\bar{s}}^t$ means the reward from predicting state $\bar{s}$ to its next predicting state. $\beta\in(0,1)$ is a weight factor.

The coordination graph (CG) framework [13] for collaborative multi-agent system assumes the action of an agent *i* only depends on a subset of the other agents $\Gamma(i)$. This assumption is consistent with our transportation scenario since the traffic flow of an intersection is affected by the nearby intersections. In this paper, $\Gamma(i)$ means the agents that in the same group with *i*. Based on this assumption, we use a decomposable Q-function $\hat{Q}(s,a)$ to approximate the global Q-function $Q(s,a)$. This $\hat{Q}(s,a)$ is defined as a sum of local Q-functions of the neighbor agents:

$$\hat{Q}(s,\bar{s},a) = \sum_{l\in G} Q(s_l,\bar{s}_l,a_l) \tag{3}$$

Where $Q(s_l,\bar{s}_l,a_l)$ is the Q-value for agent on node *l* by executing action a_l on joint state $(s_l,\bar{s}_l)$ and behaving globally optimally from then on in respect to maximizing $\hat{Q}(s,\bar{s},a)$. This approximation is reasonable because the global reward (congestion reduction) is the sum of local reward and $\hat{Q}(s,\bar{s},a)$ is closely related to $Q(s_l,\bar{s}_l,a_l)$. With the $\hat{Q}(s,\bar{s},a)$ approximation, the Q-learning update rule in equation (2) can be changed to

$$\begin{aligned} \sum_{l\in G} Q_l(s_l^t,\bar{s}_l^t,a_l^t) &= (1-\alpha)\sum_{l\in G} Q_l(s_l^t,\bar{s}_l^t,a_l^t) + \\ &\alpha\{\sum_{l\in G}[(1-\beta)r_{sl}^t + \beta r_{\bar{s}l}^t] + \gamma \max_a \hat{Q}(s^{t+1},\bar{s}^{t+1},a)\} \end{aligned} \tag{4}$$

In order to decompose $\max_a \hat{Q}(s^{t+1},\bar{s}^{t+1},a)$, we define $a^* = \arg\max_a \hat{Q}(s,\bar{s},a)$. Then we get

$$\begin{aligned} \max_a \hat{Q}(s^{t+1},\bar{s}^{t+1},a) &= \hat{Q}(s^{t+1},\bar{s}^{t+1},a^*) \\ &= \sum_{l\in G} Q_l(s_l^{t+1},\bar{s}_l^{t+1},a_l^*) \end{aligned} \tag{5}$$

Then the Q-learning update rule for each local group is

$$Q_1(s_1^t, \bar{s}_1^t, a_1^t) = (1-\alpha)Q_1(s_1^t, \bar{s}_1^t, a_1^t) + \alpha\{[(1-\beta)r_{s1}^t + \beta r_{\bar{s}1}^t] + \gamma Q_1(s_1^{t+1}, \bar{s}_1^{t+1}, a_1^*)\} \quad (6)$$

In order to find the joint action that maximizes the joint Q-value for both current state and predicting state, we use the max-sum algorithm proposed by Stranders et al. [14]

5 Experimental Evaluation

5.1 Simulation System

In order to evaluate our method, we created a transportation IoT simulation system which is shown in Figure 3. This system is based on the road traffic simulation package SUMO [15]. 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. The event manager is connected to SUMO through the TraCI (Traffic Control Interface) interface of SUMO to get the induction loop variables and vehicle location variables. The values of these variables are used to simulate RFID and GPS readers.

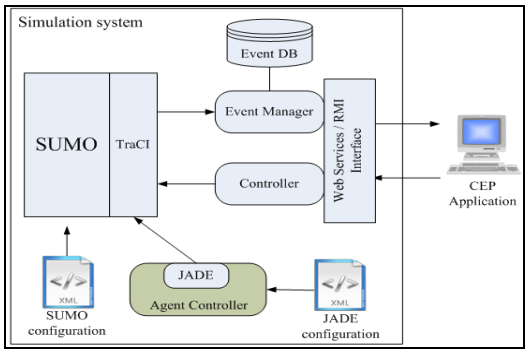

Figure 3. Simulation system architecture.

In the simulation system we set a road network of 15×15 intersections and set 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. In order to support more intelligent control of vehicles, we use the JADE (Java Agent DEvelopment Framework) framework which can support multi-agent system. Based on this framework, the agent controller can control the moving of vehicles in SUMO with intelligent algorithm which considers the environment.

Based on this simulation system we evaluated the precision and performance of our method. We used four servers with Xeon E3 processor and 16GB memory. The operating system is Ubuntu 14.04. Three of the servers are used to run Pro-CEP and the last server is used to run the simulation system.

In order to evaluate Pro-CEP method, we partitioned the congestion value (calculated based on the waiting queues of intersections) into 10 levels where "0" means no congestion and "9" means the highest congestion. Engel et al. proposed a framework and basic model without implementation detail. We have not found other work which has the same function with ours. Therefore our method is only compared with the default simulation system and the result is shown in Figure 4. In this experiment, only directed neighbors are partitioned into the group of predicted congestion nodes. From the figure we can see the average congestion level reduced obviously when Pro-CEP method is used. The reason is that Pro-CEP method can predict the congestion state and take some actions to avoid it proactively. The figure also shows that the reduction of congestion level becomes more obvious when the congestion level becomes high. We think the reason is that when the congestion is not serious, the proactive actions can't affect the states too much.

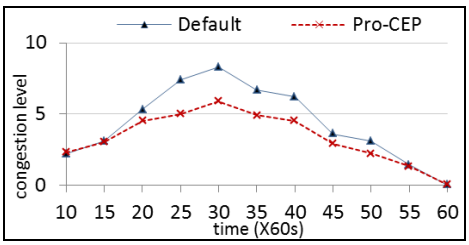

Figure 4. average congestion level over time

In the next experiment we evaluated the effect of Pro-CEP with different vehicle numbers and group partition policies. We use average reduction level to represent the effect of Pro-CEP which means the average reduced level when Pro-CEP is used comparing to default observed values. We use 3 group partition policy L_1, L_2 and L_3 where the number i in L_i means how many layers of neighbors are partitioned into the group of a predicted congestion node. The result is shown in Figure 5. As we can see, when vehicle number increases from 40 thousand, the average reduction level decreased. We think the reason is that when vehicle number is large, the density of congestion nodes becomes large. Some neighbor congestion nodes are partitioned into the same group which makes our method difficult to find optimal joint actions. We can also find that when the number i in L_i is increased, we get better result but the improvement is slight. From this result we believe in our experiment the reduction of congestion for a node is mainly related to its direct neighbor nodes.

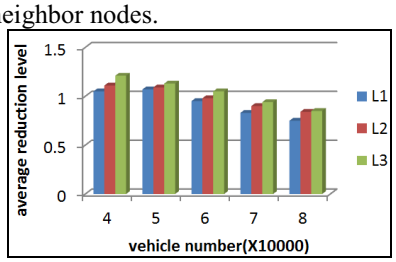

Figure 5. effect with different vehicle numbers and group partition policies

We also evaluated the performance of Pro-CEP. The average decision time with different vehicle number and

group partition policy is shown in Figure 6. We can find the average decision time increases when the vehicle number becomes large. The reason is that when vehicle number is large, more neighbor congestion nodes are partitioned into same group which makes the decision process more complex. We can also find when the number i in L_i is increased, the average decision time is increased obviously. The reason is that more neighbors need to be considered and the group becomes large. From Figure 5 and Figure 6 we conclude that we should not use large i for L_i.

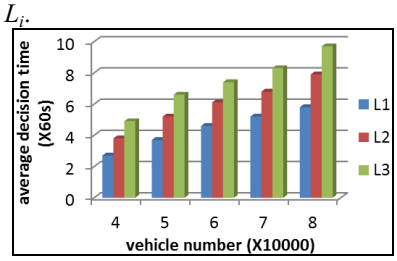

Figure 6. performance with different vehicle numbers and group partition policies

From all the experiments we can see Pro-CEP works well when processing proactive congestion control in transportation IoT. The ND-MDP$^+$ model has obvious effect and acceptable performance for congestion control in large transportation IoT.

6 Conclusion

In this paper we proposed a proactive complex event processing method for intelligent congestion control in transportation IoT. A networked distributed Markov decision processes model with predicting states is used as sequential decision model. Q-learning method is proposed to find optimal joint policy. The experimental evaluations show that this method works well for proactive congestion control in transportation IoT.

The performance and scalability of Pro-CEP still need to be improved. In the max-sum algorithm for ND-MDP$^+$, new optimization method needs to be find if we want to use this method for large scale applications.

References

1. D.C. Luckham. Event Processing for Business: Organizing the Real-Time Enterprise. Wiley Press, Dec. 2011.
2. K. Broda, K. Clark, R Miller, et al. SAGE: a logical agent-based environment monitoring and control system. Ambient intelligence, Lecture Notes in Computer Science, vol 5859. Springer, Berlin, Heidelberg, pp 112–117.
3. A.J. Demers, J. Gehrke, M. Hong, et al. Towards expressive publish/subscribe systems. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), v 3896 LNCS, 2006, pp. 627–644.
4. Y. Engel, O. Etzion. Towards proactive event-driven computing. In Proc. of Fifth ACM International Conference on Distributed Event-Based Systems, DEBS 2011, New York, pp.125-136.
5. O. Etzion, P. Niblett. Event Processing in Action. Manning Publications, 2010.
6. Q. S. Jia. On State Aggregation to Approximate Complex Value Functions in Large-Scale Markov Decision Processes. IEEE Transactions on Automatic Control, 56(2), pp. 333- 344.
7. Q. S. Jia and Q. C. Zhao. Strategy optimization for controlled Markov process with descriptive complexity constraint. Science China Series F: Inform. Sci., 52(11), pp. 1993–2005.
8. C. Zhang, V. Lesser. Coordinated Multi-Agent Reinforcement Learning in Networked Distributed POMDPs. In Proc. of the 25th AAAI Conference on Artificial Intelligence and the 23rd Innovative Applications of Artificial Intelligence Conference, August 7, 2011, San Francisco, CA, United states, pp. 764-770.
9. A. Fares, W. Gomaa. Multi-Agent Reinforcement Learning Control for Ramp Metering. Advances in Intelligent Systems and Computing, Vol. 1089, pp. 167-173.
10. E. Wu, Y. Diao and S. Rizvi. High-performance complex event processing over streams. In Proc. of 2006 ACM SIGMOD international conference on Management of data, June 27-29, 2006, Chicago, IL, USA.
11. Y.H Wang, K. Cao and X.M Zhang. Complex Event Processing over Distributed Probabilistic Event Streams. COMPUTERS & MATHEMATICS WITH APPLICATIONS, 66 (2013), pp1808–1821.
12. A. Pascale, M. Nicoli. Adaptive Bayesian network for traffic flow prediction. In Proc. of Statistical Signal Processing Workshop (SSP), 2011 IEEE, pp.177-180.
13. M. Papageorgiou, H. Hadj-Salem, J. M. Blosseville. Alinea: A local feedback control law for on-ramp metering. Transportation Research Record (1320) (1991).
14. R. Stranders, A. Farinelli, A. Rogers, et al. Decentralised coordination of mobile sensors using the max-sum algorithm. In IJCAI, issue 2009, pp.299–304.
15. M.Behrisch, L. Bieker, J. Erdmann, et al. Sumo - simulation of urban mobility: An overview. Proceedings of the third international conference on advances in system simulation, Barcelona, Spain, October, pp. 63–68.