Research Article

Modeling and Analysis of Connected Traffic Intersections Based on Modified Binary Petri Nets

Omar Yaqub^{1,2} and Lingxi Li^{1,2}

¹ Department of Electrical and Computer Engineering, Indiana University-Purdue University Indianapolis (IUPUI), Indianapolis, IN 46202, USA

² Transportation Active Safety Institute (TASI), Indiana University-Purdue University Indianapolis (IUPUI), Indianapolis, IN 46202, USA

Correspondence should be addressed to Lingxi Li; ll7@iupui.edu

Received 30 October 2012; Revised 18 February 2013; Accepted 19 February 2013

Academic Editor: Aboelmagd Noureldin

Copyright © 2013 O. Yaqub and L. Li. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

We propose an approach for the modeling and analysis of two connected traffic intersections based on Petri nets (PNs). We first use a PN to model an isolated four-way signalized intersection; then we extend it to model two successive signalized intersections. We find that this model has unbounded places, which in turn results in some confliction problems. Hence, we introduce the concept of modified binary petri nets (MBPNs) to overcome the limitation and resolve the confliction problem when we design our model and its controller. This MBPN model is a powerful tool and can be useful for the modeling and analysis of many other traffic applications.

1. Introduction

As a powerful tool that consists of a combined graphical and mathematical representation, Petri nets have been used for the modeling, control, and analysis in different applications including sensor networks [1], power systems [2, 3], manufacturing systems [4–7], and many other practical systems. In traffic management problems, Petri nets have been used to model the traffic network in different ways for a variety of purposes. It can be concluded that when vehicle flow has been studied, hybrid Petri nets (HPNs) are a suitable modeling tool because they consist of both continuous and discrete nodes that work together to reflect the dynamics of the overall traffic system. Continuous nodes are suitable for modeling continuous events such as vehicle flow while discrete nodes are used to represent discrete events such as phase change in traffic signal and enabling/disabling vehicle movement because of occurrences of emergent events such as accidents and the blocking of the road. In [8], a general HPN model for transportation system was developed. Traffic flow was described by continuous nodes and the events that affect the traffic dynamics were modeled through discrete nodes. In [9], a simple HPN was used to model the intersection of two one-way streets, while in [10] a continuous Petri net was

used to model a nonsignalized intersection, and then were added discrete nodes that are essential to represent a four-way intersection with two-phase traffic light through an HPN. In [11], the authors developed an HPN model to improve the performance of special and emergency vehicles.

In the aforementioned works on traffic network modeling and management, HPN models were adopted because they are more accurate to reflect the dynamics of the entire traffic network for certain applications. On the other hand, however, in some traffic network applications such as control and monitoring, only events are critical to be studied and analyzed. For these problems, discrete Petri nets as well as timed/colored Petri nets are the suitable modeling tools. In [12], the authors proposed a discrete Petri net model which describes the phase change of traffic light signal for an intersection. In [13], a discrete Petri net model for a small transportation system was developed to estimate the optimal travel route between the starting and destination points. In [14, 15], timed colored Petri nets were used to represent urban traffic light control systems and then a real-world supervisor of the modeled urban traffic light system was implemented. Deterministic and stochastic Petri nets were used in [16] to model railroad level crossing traffic control systems to identify and avoid the critical scenarios.

Compared with HPN models, discrete Petri nets are much simpler in terms of modeling and analysis. Therefore, for applications that do not need to capture the detailed vehicle flow dynamics, discrete Petri nets are a good tool to use. In this paper, we introduce a discrete Petri net model for two connected traffic intersections. This model is based mainly on higher-level event occurrences without considering details of lower-level continuous dynamics. For instance, it can tell us that the event of vehicles crossing an intersection during a specific traffic light phase from a specific entrance to a known destination has occurred, but without considering information such as how many vehicles have crossed the intersection or how much time this process has taken. The advantage is that this model can give us an abstract view of the event correlations about the entire complex traffic network. Therefore, it can be used to analyze the effect of the occurrence of a specific event at a specific node on other parts of the network. In addition, we propose a modified binary Petri net (MBPN) model that is suitable for the analysis and control of these connected intersections.

The contributions of this paper are two-fold: (1) we develop a comprehensive discrete Petri net model for two connected intersections by considering both signal phase change and vehicle flow directions; and (2) we propose a modified binary Petri net model to resolve the potential confliction problem for the two-intersection PN.

This paper is organized as follows. In Section 2, the basics of discrete Petri nets are briefly reviewed. A discrete Petri net model for a single signalized intersection is described in detail in Section 3. In Section 4, two connected intersections are modeled by extending the single-intersection model and are analyzed in detail. In Section 5, we propose the modified binary Petri net for the purpose of monitoring and control. Conclusions and directions for future work are presented in Section 6.

2. Petri Nets Basics

In this section, the basic principles of (discrete) Petri net are briefly reviewed. The reader can find more details about Petri nets and their extensions in [17–19].

A Petri net is a bipartite directed graph comprising of two types of nodes, called *places* (drawn as circles) and *transitions* (drawn as bars), which are connected through arrows that are called *arcs*. A Petri net is called a marked Petri net if a nonnegative integer number of tokens (drawn as black dots) is assigned to each place.

Thus, a marked Petri net is defined as

$$\text{PN} = (P, T, B^-, B^+, m_0), \quad (1)$$

where $P = \{p_1, p_2, \dots, p_n\}$ is a finite set of places consisting of n elements; $T = \{t_1, t_2, \dots, t_m\}$ is a finite set of transitions consisting of m elements; B^- is the input incident matrix, which captures the arc weights directed from places to transitions. For instance, $B^-(p_i, t_j)$ is the weight of the arc directed from the place p_i to transition t_j . If there is no such arc, then the element of $B^-(p_i, t_j)$ is set to be zero; B^+ is the output incident matrix, which captures the arc weights directed from

transitions to places. For instance, $B^+(p_i, t_j)$ is the weight of the arc directed from the transition t_j to place p_i . If there is no such arc, then the element of $B^+(p_i, t_j)$ is set to be zero; m_0 is the initial marking (denotes the number of tokens in each place initially) of the Petri net.

It is not difficult to see that the sizes of the input and output incident matrices are $n \times m$.

A transition is said to be enabled if each of its input places contains a number of tokens that is greater than or equal to the arc weight that connects that specific place to the transition. In other words, transition t_j is enabled at a specific marking m_k if and only if

$$m_k(p_i) \geq B^-(p_i, t_j), \quad (2)$$

for all input places p_i to transition t_j . The inequality is taken elementwise. An enabled transition may fire. The firing of a transition is the mechanism of removing a specific number of tokens from each input place (which is equal to the weight of the arc directing from this specific input place to the transition) and depositing a specific number of tokens to each output place (which is equal to the weight of the arc directing the transition to this specific output place).

Let S be a transition firing sequence; then we define s as the firing vector of sequence S in such a way that each element s_j in s captures the number of times that transition t_j fires in S . For instance, if $S = t_2 t_3 t_1 t_1$ and the Petri net has four transitions, then $s = [2 \ 1 \ 1 \ 0]^T$. For a given Petri net marked with a specific marking m_k , we say that the firing sequence S is enabled if and only if a specific marking $m_{k'}$ exists in such a way that when the firing sequence S is applied to the given Petri net marked with m_k it will produce the marking $m_{k'}$. Mathematically, transition firings can be captured by state equation given as follows:

$$m_{k'} = m_k + B \cdot s, \quad (3)$$

where B is the incident matrix of the Petri net and it is defined as $B = B^+ - B^-$ (B^- is the input incident matrix and B^+ is the output incident matrix of the Petri net as defined earlier).

Example 1. Consider the Petri net shown in Figure 1. The net consists of four places: p_1, p_2, p_3 , and p_4 and four transitions: t_1, t_2, t_3 , and t_4 . Since p_1 is the only place that has a token, the initial marking is written as $m_0 = [1 \ 0 \ 0 \ 0]^T$. The input and output incident matrices (B^-, B^+) are given by

$$B^- = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B^+ = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{bmatrix}. \quad (4)$$

So the incident matrix B can be calculated as

$$B = B^+ - B^- = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 \end{bmatrix}. \quad (5)$$

According to (2), it is not difficult to see that the only enabled transition is t_1 at the initial marking. When t_1 fires, one token

FIGURE 1: A simple Petri net model.

is removed from place p_1 and one token is added to p_2 and p_3 , respectively, to result in the new marking $m = [0 \ 1 \ 1 \ 0]^T$. Mathematically, the new marking can also be obtained by using (3), where m_k is the initial marking m_0 , $m_{k'}$ is the new marking, and the firing vector s is given by $[1 \ 0 \ 0 \ 0]^T$ since we are looking for the result of firing t_1 only once. Therefore, we have

$$m_{k'} = m_k + B \cdot s$$

$$= \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}. \quad (6)$$

3. Petri Net Model for a Single Intersection

3.1. Structure of a Single Intersection. In Figure 2, a four-bidirectional-road signalized intersection is shown with its accessible roads. Direction symbols {n, s, e, w} are used to define the four bidirectional roads. They represent the four directions north, south, east, and west, respectively. The set $\{R_w^{\text{in}}, R_e^{\text{in}}, R_s^{\text{in}}, R_n^{\text{in}}, R_w^{\text{out}}, R_e^{\text{out}}, R_s^{\text{out}}, R_n^{\text{out}}\}$ represents the roads connecting to the intersection; the subset $\{R_w^{\text{in}}, R_e^{\text{in}}, R_s^{\text{in}}, R_n^{\text{in}}\}$ represents the incoming roads while the subset $\{R_w^{\text{out}}, R_e^{\text{out}}, R_s^{\text{out}}, R_n^{\text{out}}\}$ represents the outgoing roads.

The Petri net model, which will be introduced in the following subsection, is designed based on the assumption that this intersection's traffic light signals have four phases as a case of study as shown in Table 1. However, note that the design concept can also be applied to other phase plans. In general, the number of phases for a four-bidirectional-signalized intersection is no more than eight, as shown in [12]. In particular, two- and six-phase plans [12, 20] are also applicable for such an intersection. Our PN model here can also be modified to account for two-phase and six-phase signalized traffic intersection. A phase is defined as an event of giving particular permissions to vehicles coming from specific directions to cross the physical area of the intersection towards specific destinations.

TABLE 1: Four phases of the signalized intersection considered in this paper.

Phase	Enabled movements
1	
2	
3	
4

FIGURE 2: The structure of a single intersection.

As shown in Table 1, during Phase 1, vehicles flowing from east to south and from west to north are permitted to cross the intersection; during Phase 2, vehicles flowing from east to west (and north) and from west to east (and south) are permitted to cross the intersection; during Phase 3, vehicles flowing from north to east and from south to west are permitted to cross the intersection; and during Phase 4, vehicles flowing from north to south (and west) and from south to north (and east) are permitted to cross the intersection. It is not difficult to check that these signal phases do not conflict with each other and they comprise of a complete signal cycle.

3.2. Petri Net Model of the Single Intersection. The discrete Petri net model for the four-bidirectional road (shown in Figure 2) is depicted in Figure 3. The overall PN model is divided into 13 sub-PNs (modules), where each one is bounded by a dashed rectangular box. Eight of these modules represent the movement of vehicles while entering and crossing the intersection from left ($N_l^{\text{in}}, S_l^{\text{in}}, E_l^{\text{in}},$ and W_l^{in}) or from forward/right ($N_f^{\text{in}}, S_f^{\text{in}}, E_f^{\text{in}},$ and W_f^{in}). On the other hand, the four sub-PNs ($N^{\text{out}}, S^{\text{out}}, E^{\text{out}},$ and W^{out}) represent movement of vehicles while leaving the intersection toward

the four directions. Finally, the sub-PN (T) models the phase change of the traffic light signal.

The sub-PN (T) consists mainly of four places (p_{p1} , p_{p2} , p_{p3} , and p_{p4}) and four transitions (t_{p1}^s , t_{p2}^s , t_{p3}^s , and t_{p4}^s) that represent the four different phases mentioned in Table 1. The other nodes in module T exist only to make sure that the model offers a safe operation (no directional conflicts between vehicles while crossing the intersection). Having a token at the place p_{pk} indicates that the k th phase is taking place while losing this token means the ending of this phase. Furthermore, firing of transition t_{pk}^s reflects the starting of the k th phase. It is easy to note that the condition $p_{p1} + p_{p2} + p_{p3} + p_{p4} = 1$ has to be satisfied all the time.

Events related to vehicle movement through the intersection are described by other 12 sub-PNs; eight of them are used to model vehicles entering and crossing the intersection as I_j^{in} where I takes the notations (N, S, E, and W) and j takes either l (left) or f (forward and right). The subnet I_j^{in} consists of five places and four transitions; having a token in place p_{ij}^{in} implies that vehicles are entering the intersection from the i th incoming direction with the intention to take the direction of j . Similarly, having a token in place p_{ij}^q indicates the event that there is a queue of vehicles waiting in the incoming direction i with the intention to cross the intersection directed j (left or forward/right). Furthermore, a token at the places p_{ij}^p means that the permission is given to those vehicles in the queue to cross the intersection while a token existing in p_{ij}^c represents the event that those vehicles are crossing the intersection.

Finally, the other four sub-PNs (N^{out} , S^{out} , E^{out} , and W^{out}) are quite simple. Each of them consists of a place p_i^{out} and a transition t_i^{out} . A token in the aforementioned places describes the event that vehicles are leaving the intersection at the outgoing direction i .

3.3. Dynamics of the Petri Net Model. The state evolution of the PN model shown in Figure 3 is performed based mainly on the state equation given in (3) and on the assumption that whenever a transition is enabled it fires immediately. The fact that permission is given to vehicles entering the intersection from a specific direction towards a specific destination during a specific phase results in the consideration that this permission has to be canceled at the end of this phase no matter some vehicles cross the intersection or not. Consequently, eight conflicts might appear in the model between each coupled transitions (t_{ij}^p , t_{ij}^s). The proposed algorithm specifies a priority execution for each conflict. If the initial marking of the Petri net model is the one given in Figure 3, the only enabled transition is t_{p1}^s . Therefore, it fires immediately through removing a token from each input place to this transition (p_{p4} , p_{p4}^2) and adding a token to each output place of this transition (p_{p1} , p_{p1}^1 , p_{w1}^p , and p_{cl}^p). It simply means that the first phase starts (indicated via the token in place p_{p1}^{in}) and permission is given to vehicles to cross the intersection.

To clarify the firing sequence, let us stick only to those vehicles that enter from west towards north. As shown in Figure 3, initially a token exists in place p_{w1}^q , so vehicles are

waiting in the west entrances to take left with the intersection. However, a conflict appears at this step between transitions (t_{w1}^p , t_{w1}^s) since both of them are enabled and only one token is marked in their common input place p_{w1}^p . At this point, the algorithm will give the priority to transition t_{cl}^p , whose firing will result in removing tokens from the permission and queuing places (p_{w1}^p , p_{w1}^q) and adding a token to the place p_{w1}^c , which indicates that the event of vehicles crossing the intersection from west to north is taking place simultaneously with the occurrence of the event that vehicles are crossing the intersection from east to south.

In the following step, three transitions are fired together, named (t_{w1}^c , t_{cl}^c , and t_{p2}^s). Firing the transitions t_{w1}^c , t_{cl}^c implies that vehicles are leaving the intersection towards south and north (through removing tokens from p_{w1}^c , p_{cl}^c) and adding a token to each one of p_s^{out} , p_n^{out} indicates the event that vehicles are leaving intersection towards south and north directions. Simultaneously, in the sub-PN, the transition t_{p2}^s fires will indicate the starting of the second phase.

The necessity of having the sink transition t_{w1}^s is to make sure that vehicles cross the intersection safely; if we assume that no vehicles are waiting in the queue to cross the intersection from west to north during the first phase, the permission will still be available through the existence of a token in p_{w1}^p , which will lead to unsafe operation that vehicles coming from the west cross the intersection even after the ending of the first phase. Therefore, the role of the sink transition t_{w1}^s is to take away the permission if it has not been used during the first phase.

We developed an algorithm to identify the enabled transitions at each time step and fire the enabled transitions immediately based on the state equation and the aforementioned priority rule. Given the initial marking, input and output incident matrices, the algorithm will firstly construct the column matrix A (which initially is equal to the initial marking); then it will calculate the subsequent markings by state equation given in (3) and compare them with the existing marking; if it is not equal to any of them, it will be saved as a new column in matrix A and will be used to find the subsequent markings. Otherwise, if the newly calculated marking is equal to any previous ones (any column in A), the algorithm will stop. Matrix A at this point consists of all possible markings in their order of occurrences. The block diagram of the algorithm (Algorithm 1) for capturing the state evolution of the PN is shown in Figure 4.

In Table 2, we capture the state (marking) evolution of the Petri net model shown in Figure 3 related to each step of transition firing. Note that this state evolution is partial since we only show the number of tokens in some important places. The first column in the table represents the initial marking of the Petri net. When a specific event occurs at a specific firing step, it will be represented through the value “1” in the table. Similarly, the change of value from “1” to “0” in a specific table cell implies the ending of that specific event. Since the model consists of 56 places, it is not possible to include all of them in the table. We focus only on the places

FIGURE 4: Block diagram of Algorithm 1 for capturing the state evolution of the Petri net shown in Figure 3.

with essential meanings. It is to be noted that at each firing step, one and only one of the four phase places (p_{p1} , p_{p2} , p_{p3} , and p_{p4}) will take the value “1” (which is expected as previously mentioned). It is also not difficult to notice that during the 10th step, the current marking goes back to the second generated marking in Table 2, which indicates that the proposed PN model is live and reversible.

If we use a different initial marking, such as a marking that represents the case where no vehicles enter the intersection from north, south, and east, the value “1” will never exist at the place p_w^{out} during any firing step, which means that the

event vehicles leave the intersection towards west will not take place. This is an expected logical result to avoid the conflicts of events.

4. Petri Net Model for Two Connected Intersections

In this section, we generalize the PN model obtained from the previous section to two connected intersections. The structure of two connected intersections is shown in Figure 5; if we call the intersection on the left side *Intersection-1* and

TABLE 2: Partial state evolution of the single-intersection Petri net model shown in Figure 3.

	m_0	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
p_{p1}	0	1	1	0	0	0	0	0	0	1	1
p_{p2}	0	0	0	1	1	0	0	0	0	0	0
p_{p3}	0	0	0	0	0	1	1	0	0	0	0
p_{p4}	1	0	0	0	0	0	0	1	1	0	0
p_{ij}^{in}	1	1	1	1	1	1	1	1	1	1	1
p_{wl}^c, p_{el}^c	0	0	1	0	0	0	0	0	0	0	1
p_{wf}^c, p_{ef}^c	0	0	0	0	1	0	0	0	0	0	0
p_{nl}^c, p_{sl}^c	0	0	0	0	0	0	1	0	0	0	0
p_{nf}^c, p_{sf}^c	0	0	0	0	0	0	0	0	1	0	0
p_w^{out}	0	0	0	0	0	1	0	1	0	1	0
p_e^{out}	0	0	0	0	0	1	0	1	0	1	0
p_n^{out}	0	0	0	1	0	1	0	0	0	1	0
p_s^{out}	0	0	0	1	0	1	0	0	0	1	0

the intersection on the right side *Intersection-2*, then we can say that the east output of *Intersection-1* is the west input to *Intersection-2* while the west output of *Intersection-2* is the east input to *Intersection-1*.

For this case, the Petri net model for traffic network shown in Figure 5 consists of two-single intersection PN models connected to each other. However, minor modifications are done for each model. For *Intersection-1* model, the arcs connecting t_{el}^{in} to p_{el}^{in} and t_{ef}^{in} to p_{ef}^{in} have been removed; while for *Intersection-2* model, arcs connecting t_{wl}^{in} to p_{wl}^{in} and t_{wf}^{in} to p_{wf}^{in} have been removed. To distinguish between the PN nodes belonging to each intersection, we added (2) as the subscript for each node symbol in *Intersection-1*, similarly adding (3) for *Intersection-2*. The two-single intersection models are connected to each other through their east and west inputs/outputs nodes, the PN model for the connection between two intersections is shown in Figure 6.

During the event that vehicles are leaving *Intersection-1*, a token existing at $p_{e(1)}^{\text{out}}$ will result in the firing of transition $t_{e(1)}^{\text{out}}$, which removes the token from $p_{e(1)}^{\text{out}}$ and adds a token to both places $p_{wf(2)}^{\text{in}}$ and $p_{wl(2)}^{\text{in}}$ that represents the occurrence of the event that vehicles are entering *Intersection-2* from the west. However, this step does not perform very smoothly when we execute the two-intersection model using Algorithm 1.

The case is that, during one-phase cycle (the four phases took place once successively), transitions $t_{e(1)}^{\text{out}}$ and $t_{w(2)}^{\text{out}}$ fire three times because places $p_{e(1)}^{\text{out}}$ and $p_{w(2)}^{\text{out}}$ are charged three times per cycle since there are vehicles leaving *Intersection-1* east directed and *Intersection-2* west directed during three phases in a cycle. On the other hand, the four places, $p_{ef(1)}^{\text{in}}$, $p_{el(1)}^{\text{in}}$, $p_{wf(2)}^{\text{in}}$, and $p_{wl(2)}^{\text{in}}$, gain three tokens each per cycle and release only one of these tokens since the transitions downstream fire only once per cycle. It is obvious that this firing mechanism will lead the Petri net to be unbounded. Physically, the model will not be considered as a good representation for event occurrences because at some point if no vehicle enters the network through one of the three entrance of *Intersection-1*: west, north, and south, there will

be still vehicles traveling from *Intersection-1* to *Intersection-2* which is not a real case. That is because places $p_{ef(1)}^{\text{in}}$, $p_{el(1)}^{\text{in}}$ will have tokens to feed the net of *Intersection-2*. The same case of unreal representation will exist for vehicles traveling from *Intersection-2* towards *Intersection-1*.

To tackle this modeling issue, we have tried several approaches. Firstly, a Petri net controller was designed based on [21] to keep the number of tokens in each of the four places not exceeding one. This controller worked for these four places but it transferred the unbounded problem to the upstream places $p_{e(1)}^{\text{out}}$ and $p_{w(2)}^{\text{out}}$. Adding another controller places to control these two nodes will not solve the problem because it causes conflicts which stop the execution of Algorithm 1 when it is used for the new model (with the controller). Designing a controller for the six places in one step will cause similar problem. Therefore, we propose the concept of the modified binary Petri net (MBPN), which will be presented in detail in the next section.

5. Modified Binary Petri Nets

5.1. Introduction to Modified Binary Petri Nets. As it can be seen from the previous discussions, we use PNs to determine whether an event has occurred or not based on the occurrence of other events. Modeling in such a way can be easily done through representing the occurrence of a specific event by associating a token in a specified place that describes this event. However, modeling in this traditional way will lead to problems in our models as discussed above.

Thus, we propose the concept of the modified binary Petri net (MBPN). The idea is to follow the traditional Petri net firing mechanisms, but with a restriction that no places can be marked with more than one token. Thus, the MBPN can be defined as follows.

- (i) The weights of all arcs in the MBPN are ones, which indicates that the net is ordinary and all elements in the input and output incidents matrices can only be zeros or ones.

FIGURE 5: The structure of two connected intersections.

FIGURE 6: Petri net model for the connections between two intersections.

- (ii) The initial marking of the MBPN can only be zero or one for each place.
- (iii) State evolution from one marking to another is performed through the following two steps:

- (1) we calculate marking $m_{k'}$ according to traditional state equation of Petri nets:

$$m_{k'} = m_k + B \cdot s; \quad (7)$$

- (2) we update the marking in places according to the number of tokens as follows:

$$m_{k'}(p) = \begin{cases} 0 & \text{for places } p \text{ such that } m_{k'}(p) = 0, \\ 1 & \text{for places } p \text{ such that } m_{k'}(p) \geq 1. \end{cases} \quad (8)$$

Based on this definition, it is easy to see that for MBPN we have the following:

- (i) elements of incident matrices take only three values $-1, 0$, or 1 ;
- (ii) any modified binary Petri net is bounded since the number of states is less than or equal to 2^n , where n is the number of places.

To illustrate the differences between the traditional PN and the MBPN, we can use the example provided in Figure 1 for illustration. If Algorithm 1 is applied for traditional PN, the marking evolution will be given by

$$\begin{aligned} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} &\xrightarrow{t_1} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{t_2, t_3} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 2 \end{bmatrix} \xrightarrow{t_4} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{t_1, t_4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ &\xrightarrow{t_1, t_2, t_3} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 2 \end{bmatrix} \xrightarrow{t_2, t_3, t_4} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 3 \end{bmatrix} \xrightarrow{t_1, t_4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 2 \end{bmatrix} \\ &\xrightarrow{t_1, t_2, t_3, t_4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 3 \end{bmatrix} \xrightarrow{t_1, t_2, t_3, t_4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 4 \end{bmatrix} \dots \begin{bmatrix} 0 \\ 1 \\ 1 \\ r-1 \end{bmatrix} \\ &\xrightarrow{t_1, t_2, t_3, t_4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ r \end{bmatrix} \xrightarrow{t_1, t_2, t_3, t_4} \begin{bmatrix} 1 \\ 1 \\ 1 \\ r+1 \end{bmatrix} \dots \end{aligned} \quad (9)$$

As it is shown above, the PN is unbounded due to place p_4 , in which the number of tokens will keep increasing after the marking $[1 \ 1 \ 1 \ 2]^T$.

Now we consider the MBPN. It is not difficult to see that the marking evolution is given by

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \xrightarrow{t_1} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \xrightarrow{t_2, t_3} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \xrightarrow{t_4} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \quad (10)$$

Thus, MBPN has bounded markings and avoids the potential problems in creating extra tokens for our connected traffic intersection model. In general, for traffic applications that can be represented through binary-state nodes (0 or 1), MBPN is a suitable modeling tool and can guarantee the smooth representation for event occurrences in these applications.

5.2. Using MBPN Firing Mechanism for Studying the Two Connected Intersections. Now we investigate the PN model

TABLE 3: A partial set of marking evolution of MBPN model for two connected intersections.

	m_0	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th	11th	12th	13th	14th	15th
$P_{p1(1)}, P_{p1(2)}$	0	1	1	0	0	0	0	0	0	1	1	0	0	0	0	0
$P_{p2(1)}, P_{p2(2)}$	0	0	0	1	1	0	0	0	0	0	0	1	1	0	0	0
$P_{p3(1)}, P_{p3(2)}$	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	0
$P_{p4(1)}, P_{p4(2)}$	1	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1
$P_{e(1)}^{\text{out}}$	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1
$P_{wl(2)}^{\text{in}}$	0	0	0	0	0	0	1	0	1	1	1	0	0	0	1	1
$P_{wf(2)}^{\text{in}}$	0	0	0	0	0	0	1	0	1	1	1	1	1	0	1	1
$P_{w(2)}^{\text{out}}$	0	0	0	0	0	1	0	1	0	1	0	0	0	1	0	1
$P_{el(1)}^{\text{in}}$	0	0	0	0	0	0	1	0	1	1	1	0	0	0	1	1
$P_{ef(1)}^{\text{in}}$	0	0	0	0	0	0	1	0	1	1	1	1	1	0	1	1

shown in Figure 6. We use Algorithm 1 with a small modification that takes into account (8) mentioned above by MBPN. The algorithm performed 15 steps before it stopped due to the condition of having repeated markings. Because of the size of the PN model ($P = 112, T = 88$), we do not show markings for all places in the net. Instead, in Table 3, we show the markings of the places which represent the connections between two intersections. It is easy to note that the MBPN framework resolves the issue of unbounded states and is a more realistic tool for the modeling and analysis of certain traffic applications other than traditional PNs.

5.3. Discussions on Potential Extensions to Traffic Networks. Starting from two connected intersections model, we can expand the approach proposed in this paper to study traffic network with much more complex dynamics. To see this implication, assuming that we have a complex traffic network consisting of n intersections, we can derive its input incident matrix B^- composed from input incident block matrices as:

$$B^- = \begin{bmatrix} B_1^- & 0 & \cdots & 0 \\ \vdots & B_2^- & 0 & \vdots \\ 0 & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & B_n^- \end{bmatrix}, \quad (11)$$

where $B_1^-, B_2^-, \dots, B_n^-$ are the input incident matrices for intersections $I_1, I_2, \dots, I_n$, respectively. The sizes of these block matrices may be different because they depend on the intersection specifications such as the number of incoming and outgoing roads connected to them and the number of phases applied to them. Note that the off-diagonal entries are zeros because intersections are connected through transitions to places.

For the output incident matrix, the diagonal block matrices will be the output incident matrices $B_1^+, B_2^+, \dots, B_n^+$ for the intersections $I_1, I_2, \dots, I_n$. Off-diagonal block matrices represent the connection between two intersections. For instance, B_{1-2}^+ represents the connections from intersection I_1 to intersection I_2 , B_{2-1}^+ represents the connection from I_2

to I_1 , and so forth. The general form of the output incident matrix for the overall traffic network can be written as

$$B^+ = \begin{bmatrix} B_1^+ & B_{2-1}^+ & \cdots & B_{n-1}^+ \\ B_{1-2}^+ & B_2^+ & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ B_{1-n}^+ & B_{2-n}^+ & \cdots & B_n^+ \end{bmatrix}. \quad (12)$$

From the discussions above, we observe that we can easily expand the two-intersection model proposed in this paper to a large traffic network with much more intersections. In addition, the MBPN model proposed can guarantee the boundedness of places in the net; thus, it is suitable for capturing the event occurrences of the overall traffic network.

6. Conclusions and Discussions

In this paper, we developed a discrete Petri net model for a single traffic intersection and then extended it to represent a traffic network consisting of two connected intersections. This model was based on traditional Petri nets in order to analyze the effect of occurrence/nonoccurrence of specific events on other network nodes without considering the detailed dynamics. An algorithm was designed to capture transition firings based on the state equation and the priority assignment. We noticed that the traditional Petri net model for two connected intersections will lead to unbounded places in the model, which are not realistic for the monitoring and control. Thus, we introduced a modification to the traditional Petri nets to resolve this issue. We called this new type of Petri nets the *modified binary Petri nets*, which have a good representation for traffic network in terms of events occurrences and traffic dynamic correlations.

One direction for our future work is to enhance the model with timing information by using timed transitions. By incorporating with time information, the new model will be suitable for event simulations in large traffic networks. Another interesting research direction is to identify a wider range of applications (e.g., routing, scheduling, etc.) of modified binary Petri nets in traffic systems and networks.

References

- [1] L. Li and D. S. Kim, "Least-cost path estimation in wireless ad hoc sensor networks using Petri nets," in *Proceedings of the 5th ACM International Conference on Ubiquitous Information Management and Communication (ICUIMC '11)*, Seoul, South Korea, February 2011.
- [2] C. N. Hadjicostis and G. C. Verghese, "Power system monitoring using Petri net embeddings," *IEEE Proceedings C*, vol. 147, no. 5, pp. 299–303, 2000.
- [3] N. Lu, *Power system modeling using Petri nets [Ph.D. thesis]*, 2002, <http://www.ima.umn.edu/~mali/thesis.pdf>.
- [4] F. S. Hsieh and J. B. Lin, "Context-aware workflow management for virtual enterprises based on coordination of agents," *Journal of Intelligent Manufacturing*, 2012.
- [5] F. S. Hsieh and C. Y. Chiang, "Collaborative composition of processes in holonic manufacturing systems," *Computers in Industry*, vol. 62, no. 1, pp. 51–64, 2011.
- [6] F. S. Hsieh, "Design of reconfiguration mechanism for holonic manufacturing systems based on formal models," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 7, pp. 1187–1199, 2010.
- [7] L. Li, C. N. Hadjicostis, and R. S. Sreenivas, "Designs of bisimilar Petri net controllers with fault tolerance capabilities," *IEEE Transactions on Systems, Man, and Cybernetics A*, vol. 38, no. 1, pp. 207–217, 2008.
- [8] A. di Febbraro and S. Saccone, "Hybrid modelling of transportation systems by means of Petri nets," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 131–135, October 1998.
- [9] A. di Febbraro, D. Giglio, and N. Sacco, "Modular representation of urban traffic systems based on hybrid petri nets," in *Proceedings of the IEEE International Conferences on Intelligent Transportation Systems*, pp. 866–871, August 2001.
- [10] C. R. Vázquez, H. Y. Sutarto, R. Boel, and M. Silva, "Hybrid Petri net model of a traffic intersection in an urban network," in *Proceedings of the IEEE International Conference on Control Applications (CCA '10)*, pp. 658–664, Yokohama, Japan, September 2010.
- [11] A. di Febbraro, D. Giglio, and N. Sacco, "Urban traffic control structure based on hybrid petri nets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 4, pp. 224–237, 2004.
- [12] G. F. List and M. Cetin, "Modeling traffic signal control using Petri nets," *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, no. 3, pp. 177–187, 2004.
- [13] Y. Qu, L. Li, Y. Liu, Y. Chen, and Y. Dai, "Travel routes estimation in transportation systems modeled by Petri nets," in *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety*, pp. 73–77, Qingdao, China, July 2010.
- [14] Y. S. Huang and T. H. Chung, "Modeling and analysis of urban traffic lights control systems using timed CP-nets," *Journal of Information Science and Engineering*, vol. 24, no. 3, pp. 875–890, 2008.
- [15] Y. S. Huang and P. J. Su, "Modelling and analysis of traffic light control systems," *IET Control Theory and Applications*, vol. 3, no. 3, pp. 340–350, 2009.
- [16] Y. S. Huang, Y. S. Weng, and M. Zhou, "Critical scenarios and their identification in parallel railroad level crossing traffic control systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 4, pp. 968–977, 2010.
- [17] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*, Springer, New York, NY, USA, 2008.
- [18] T. Murata, "Petri nets: properties, analysis and applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541–580, 1989.
- [19] R. David and H. Alla, *Discrete, Continuous, and Hybrid Petri Nets*, Springer, New York, NY, USA, 2005.
- [20] Y. S. Huang, "Design of traffic light control systems using statecharts," *The Computer Journal*, vol. 49, no. 6, pp. 634–649, 2006.
- [21] J. Moody, K. Yamalidou, M. Lemmon, and P. J. Antsaklis, "Feedback control of petri nets based on place invariants," *Automatica*, vol. 32, no. 1, pp. 15–28, 1996.