MODELING, ESTIMATION, AND CONTROL IN HIGHWAY TRAFFIC BASED ON DISCRETE EVENT DYNAMIC SYSTEMS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Keyu Ruan

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2020

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL

STATEMENT OF DISSERTATION APPROVAL

Dr. Lingxi Li, Co-Chair

Department of Electrical and Computer Engineering

Dr. Jianghai Hu, Co-Chair

Department of Electrical and Computer Engineering

Dr. Yaobin Chen

Department of Electrical and Computer Engineering

Dr. Shreyas Sundaram

Department of Electrical and Computer Engineering

Approved by:

Michael A. Capano

Head of the Graduate Program

ACKNOWLEDGMENTS

I would like to first thank my advisors Dr. Lingxi Li and Dr. Jianghai Hu for the help in the work of this dissertation and also in the study and daily life in the past several years at Purdue. I also would like to thank Dr. Yaobin Chen and Dr. Shreyas Sundaram, who are my final defense committee members as well as great instructors in my study.

Besides, I would like to thank Dr. Stanley Chien, Dr. Renran Tian, and Dr. Qiang Yi for the help in doing the projects in the Transportation Active Safety Institute (TASI). The experiences I gained in the past years will help a lot in the future. I also would like to thank Dr. Brian King, Miss Sherrie Tucker, and Mr. Matt Golden in the Engineering school for the help in my academic study and the advice for my academic plan.

At last, I would like to give the most special thanks to my parents, Shengrong Ruan and Liumei Di. Thank them for the support of my Ph.D. study and the instructions in my life, my thoughts, and my views of the world. I cannot make it without their help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Ruan, Keyu. Ph.D., Purdue University, December 2020. Modeling, Estimation, and Control in Highway Traffic Based on Discrete Event Dynamic Systems. Major Professors: Lingxi Li & Jianghai Hu.

Petri net (PN) is a useful tool for the modeling and analysis of complex systems and has been widely used in a variety of practical systems. This dissertation aims at studying highway transportation systems using Petri nets and investigating several fundamental problems related to the modeling, state/structure estimation, and control of highway traffic.

This dissertation starts with two kinds of modeling schemes. The first one uses the Probabilistic Petri net to model a highway segment. The traffic movement probabilities have also been shown. The second scheme uses the traditional Petri net structure to model the traffic network around a city's metropolitan area, where places represent the destinations of interests and tokens represent time units.

After that, two estimation algorithms and one control algorithm have been proposed, respectively, based on external observations. The first algorithm deals with labeled Petri nets and the objective is to estimate the minimum initial marking that has (have) the smallest token sum. The second algorithm estimates the Petri net structures from the observations of finite token change sequences in terms of the minimum number of transitions and connections. At last, the traffic volume control algorithm is to keep the traffic volume within capacity. The controller will be applied in each evolution step depending on observation.

Since we have been focusing on the optimization problems of the structure and markings of the Petri net, it is directly related to the optimal route planning problems in highway traffic scenarios. Thus, we can obtain optimized traveling routes by applying proposed algorithms to the traffic systems.

1. INTRODUCTION

1.1 Background and Motivation

With the fast development of autonomous vehicles, the optimization and planning of highway traffic become more and more important in the aspects of enhancing driving safety and saving traveling time. The various advanced sensing techniques and control units that have been utilized make the basis of the highway traffic studies. For instance, in [1], a technique was introduced for guidance control of the parallel parking, where unskilled drivers can benefit from it in parking tasks. In [2], the authors studied the strategies of decision-making and driving state control of unmanned vehicles. The objectives were achieved with interval type-2 fuzzy sets, fuzzy comprehensive evaluation, and fuzzy control rules. Authors in [3] surveyed the topics on traffic signal management and control using Model Predictive Control (MPC) algorithms. Driver behavior data were collected and analyzed in [4] for the evaluation of in-vehicle camera-based driver state sensing system that can be an objective and reliable source for improving driving safety. Other than these systems, intelligent vehicles can also provide braking assistance to ensure occupant safety. For example, the Automatic Emergency Braking (AEB) systems, such as Crash Imminent Braking (CIB) system and Dynamic brake support (DBS) system, are believed to be technologies representing the significant advances in vehicle safety. The evaluation of the CIB system [5] showed that the system can apply automatic braking to mitigate damages and avoid potential collisions. It is proved that these systems operate very well within a certain speed range towards pedestrians and bicyclists, in particular, the braking behavior of pedestrian AEB systems was analyzed in [6]. The authors of [7] evaluated Road Departure Prevention systems (RDPSs), which can potentially reduce the crash risk by issuing visual/audible warning signals to the driver and engage automatic steering adjustment when the vehicle is detected to deviate from the current lane or the road edge.

In order to evaluate the performance of these techniques and devices, other researches have been done to develop the standard procedure of performance evaluation. For instance, in [8], a testing protocol has been proposed with the scoring criteria to evaluate the performance of Crash Imminent Braking (CIB) systems. The same group of authors extended the work in [9], which combined the warning system into the performance evaluation. In [10]-[14], a large number of vehicle tests have been done in simulated scenarios with different collision targets, i.e., a pedestrian and a bicyclist. The testing results showed that Automatic Emergency Braking (AEB) systems have ideal performance in pedestrian detection and crash avoidance within a certain speed interval. Authors in [15] filled the vacancy of the emergency braking system performance evaluation in the vehicle to vehicle scenarios by proposing their approach. In [16], the AEB system has been further studied to find the limitations in its performance with multiple complex scenarios. In the evaluation methods mentioned above, the researchers have used the percentage of kinetic energy reduction as the main criterion. There are also many other studies that are based on other parameters. The authors in [17] assessed the AEB system based on the vehicle speed change. A weighting method called the ”analytic hierarchy process” has been utilized for the data analysis. The speed reduction was used as the main basis in [18] for the assessment of the behavior of the AEB system. In the study in [19], the performance of the AEB system was evaluated with the initiating distance of the auto-braking. The AEB system has been compared with the human driver in emergency situations in [20], which proved that the AEB system had much better performance in aspects like the onset distance and braking distance.

Besides the aforementioned works, the high-definition (HD) map is also an essential technique and is widely used for vehicle localization and path planning based on pre-obtained environment information. The authors in [21] presented a method for HD map updating with the cellular network that meets the requirements for highly automated driving. This work is claimed to be the first one that correlates the data requirements with the network infrastructure. In [22], the authors provided a control scheme for predictive cruise control, which utilized the HD map information. The proposed predictive cruise control system helped achieve a higher rate of fuel-saving. Other than the highway and urban driving, the HD map technique has also been used in many other situations. The authors in [23] used HD maps in transfer vehicles in smart factories, in order to perform better interactions with the environment. They proposed an HD map update strategy to solve the open problem for the sustainability of the performance. The HD map technique has also been used for underwater vehicles in [24], where the authors tried to analyze water situations and find solutions for water resources management.

Although the HD map technique has been well received in the research community, its accuracy remains to be evaluated. The authors in [25] used a framework that contains an automated ground-truth process, which provides the user with a unique visualization of the captured video and quickly generates the ground-truth information. They have also used this framework in a night-time lane detection system presented in [26]. The authors in [27] presented an efficient solution in evaluating the road marking detection algorithms. They have validated this evaluation process with a virtual database.

Furthermore, a variety of algorithms and technologies that can provide vehicles with more advanced capabilities were developed. The authors in [28] have proposed an approach for estimating the rotated angle of the steering wheel. A particle filtering algorithm was used for the estimation. By applying a deep learning-based regression framework, the patterns of normal driving and driving toward an obstacle have been evaluated. The work in [29] focused on obtaining a control structure that is used for stabilizing the vehicle when facing emergency scenarios. The structure prioritizes collision avoidance among conflicting objectives, which is implemented using model predictive and feedback controllers. In [30], another group of researchers has proposed a system using Unified Map built with various onboard sensors to detect collision risks. This system can have the information of all nearby obstacles, which makes it an efficient planner for collision-free paths.

The other researches paid their attention to the traffic emergencies on the highway. The most representative case is traffic congestion, which is extremely dangerous for the coming vehicles from downstream. In [31], the authors investigated the statistic data of the fatal traffic accidents on the highway and found that in congested driving conditions, there was a much higher crash rate, where the back-of-queue crash was about 13% of all highway fatal accidents. Because of that, the studies in [32] [33] made contributions to smoothing the driving from free flow to congestion. At the same time, works have also been done to improve the awareness of the drivers while approaching congestion. In [34], a queue warning system has been implemented, which could alert the motorists with the variable message signs. A specially designed communication system has been utilized to transmit the warning for a distance that was over 100 km. Based on the data from the authors, they have successfully reduced the rate of rear-end collision. The author in [35] developed an end-of-queue warning system based on an artificial neural network model-based algorithm. The portable variable message signs have been used for warning delivery. An end-of-queue warning system has been proposed by Y. Liu. et al. in [36]. Different traffic data like the real-time traffic velocity helped the establishment of the system. The influence of other factors, such as the road parameters, communication range, and penetration rate have also been considered. Besides, many researchers were interested in improving the performance of congestion detection methods. In [37], in order to measure the length of the queue during a certain time period, a method has been proposed, which could estimate the time needed for a certain amount of vehicles to leave the queue along with the average delay caused by the queue. The authors in [38] proposed another algorithm to estimate the queue length, which could be used in an intersection scenario. The scheme of stochastic gradient descent was utilized in this study. The study in [39] was done to evaluate the effects of the enforcement vehicles at upstream of work zones. Based on the collected data of the traffic speed, deceleration, and traffic volumes, it has been proved that the officers with their enforcement vehicles could make a reasonable solution for the congested traffic with the assistant of certain analytic approaches.

Thanks to contributions done by the researches above, it is clear that driving safety and efficiency are critical for model life. However, there still much other work that could be done. In this dissertation, we would like to focus on the optimization, estimation, and control of the highway traffic, which is indispensable for further scheduling and planning of the traffic. In order to better analyze these problems in highway traffic, discretization of the highway traffic network could be considered as a necessary preliminary step. Suppose there is a situation that a passenger wants to travel from the starting place to the finishing place. During this path, this passenger also would like to visit some other destinations, such as a supermarket. For different research purposes, we could focus on different sizes of the traffic network. When doing discretization, each discretized part of the network could have different practical meaning, such as a block of the road or a destination of interest. We could apply the algorithms to optimize, estimate, or control the structure and states of the network and obtain the best solutions.

Before we get any deeper into the content of this dissertation, we would like to first introduce the Discrete Event Dynamic System (DEDS), which is the modeling tool we would like to use in this dissertation. Discrete Event Dynamic System (DEDS) is a class of asynchronous dynamic systems that have event-driven state evolution. DEDSs have been developed in the 1940s but became widely used in researches and applications since the 1980s. A lot of theoretical models and analysis schemes from different aspects have been developed. In normal cases, only a finite number of discrete values will be taken as the state of these systems, which corresponds to possible practical conditions like the status of the system components, the number of parts waiting to be dealt with, or the indicators of the tasks like planning and scheduling. The changes of the states are because of the occurrence of certain events, such as the change of certain environmental conditions and the start or finish of system operation. DEDSs have been widely used in practical applications and are important in research areas like computer networks, transportation systems, and intelligent vehicles, which makes it an ideal tool for theoretical analysis and practical applications in large and complex processing and systems. In order to make the full use of the robustness of the DEDSs and provide the optimal solutions to the practical problems, the research targets have been focusing on integrating the various models and theoretical methods to form a multi-level, multi-model theory system in the recent studies.

In this dissertation, one of the DEDS models, the Petri nets [40], [41] will be the main tool for our study. A Petri net is a kind of mathematical and graphical language that describes the distributed dynamic systems. The powerfulness of the Petri net makes it very suitable for the simulation of the dynamic features of the asynchronous concurrent systems.

Besides, the Petri net is also ideal for the modeling and analysis of complex systems. With the tool of Petri net, in this dissertation, firstly, we use two schemes to model the traffic network. The first one uses *Probabilistic Petri net* to model the traffic in a highway segment. The movement of the vehicles has been assigned with different probabilities based on traffic data from a database. The second one models the traffic network with a larger scale using the traditional Petri net. Important destinations of interest are marked as the places in the modeled Petri net and the paths between these destinations are denoted with transitions and arcs. Weights on the arcs depend on the distances and traffic loads.

After that, an approach has been proposed to estimate the Petri net structures with the minimum scale from asynchronous observations of token change sequences [42]. An algorithm that can deal with the observed sequences with finite length has been proposed, which is able to estimate the structure of the underlying Petri net with a polynomial computational complexity in terms of the number of transitions. Note that there is implicit information contained in the observed token change sequences, which includes the initial marking, final marking, and the number of places. The solution of the proposed algorithm should be consistent with the given observations and be the optimal structure(s) that has(have) the least transitions and most sparse incident matrix, i.e., the most zero entries in the incident matrix. Besides that, another algorithm has been developed to check the consistency of the solution obtained from the first algorithm, which is to check whether the estimated Petri net structure can match future observations.

In the next part, based on the state-of-art studies, we extend the results of minimum initial marking estimates problem in labeled Petri nets only observable transitions into the ones with *unobservable* transitions (with certain special structure). Several algorithms for the minimum initial marking estimation (*MIM-UT*) [43] have been proposed. The minimum initial marking estimation problem is originally to model the industrial production process, which obtains the best production sequence with minimum source cost that can produce the same output that satisfies some certain processes. In particular, it is assumed that we have full knowledge of the structure of the target Petri net and the unobservable subnet has a special structure called *contact-free*. Given the observed label sequence, the goal is to obtain the minimum initial marking estimate set, in which each marking has the smallest token sum, and is able to fire a sequence that is consistent with the observation. We develop an algorithm to obtain the minimum initial marking estimate set with the complexity that is polynomial in terms of the length of the observed label sequence. Besides, two heuristic algorithms have been proposed to further reduce the computational complexity. We also provide an illustrative example to evaluate the performance of the proposed algorithms and compare their outputs.

After that, an algorithm has been proposed for the traffic volume control, where the traffic network of interest is represented with the Petri net structure. Each place represents one road segment and the transitions denote intersections and junctions. The transitions are assigned with an alphabet or empty labels depending on their observability, which makes the Petri net a labeled Petri net. Therefore, based on a given observed label sequence, we would like to control the network to avoid the traffic volume to be exceeding the capacity of each road segment. A controller has been given during each observation to control each transition, representing the traffic control in the intersections and junctions.

After the brief introduction of the content of this dissertation, before we can get any deeper of the details, we would like to talk about some related studies that are closely related to our topics and consider their contributions and extendable future work.

1.2 Related Work

1.2.1 Traffic Modeling

The Petri net structure is widely used in traffic modeling. Authors in [44] modeled the railway traffic with three Petri net models and did further simulation and analysis. Based on these, they built a modular approach to do traffic coordination. In [45], the timed Petri net has been used to design a traffic-signal emergency control policy. This policy could improve traffic incident response and control. X. Zhu in [46] applied the Petri net to the dynamic modeling for airport apron. The apron traffic system was divided into disjoint zones to fit the requirement of the Petri net and a method for apron conflict control is proposed. U. Bhanja [47] did optimization of traffic flow in an urban area with different techniques. The work mainly focuses on communication between roadside units and vehicles. The Petri net plays an important role in the author's work as the validation of the proposed techniques. Authors in [48] used the Petri net structure to model the urban traffic network either. The traffic network has been divided into intersections and roads, which have limited capacities. The Petri nets play a key role to represent the dynamic of traffic.

The first step to model traffic is to discretize the traffic into segments. With the developed advanced sensing and driver-assistance systems, it is feasible for intelligent vehicles to make decisions to reduce the risk of crashes based on the traffic and road conditions. Studies have been done with many different schemes. For instance, in [49], R. Aziz et. al. proposed an algorithm for segmenting the highway network. Their method of the segmenting was based on the transmitted GPS data interval. Some further work has been done to identify the break-points for the trips within each interval. These segments have been used for trip planning, infrastructure management, and other decision-making tasks. Y. Guan et. al. in [50] studied the problem of decision making of self-driving cars. Several highway scenarios are designed, which are discretized in both space and speed. Markov decision process has been applied to the discretized system in order to deal with the decision marking problem. Another approach has been proposed in [51], which is used for the optimization of the sensor placement on the highway for a better observation of traffic conditions. In [52], J. Brown et. al. proposed a method to discretize the road network in intersections in order to study the safety impact on short segments. With the discretized system, the authors have shown the differences between rural signalized and unsignalized intersections. G. Gentile et. al. in [53] developed a traffic discretization method to study dynamic traffic assignment problems. The method includes three different levels of discretization: flow, space, and time. The authors have implemented in existing commercial software for transport analysis. These methods for traffic discretization made it possible for further traffic studies of the motion planning and decision making based on discretized structures.

After the discretized model has been obtained, many researchers have modeled traffic flow with DEDS and done some optimization/minimization work. In [54], T. Nishi et. al. proposed an approach with the Petri net to optimize route planning problems for automated guided vehicles. Their goal is to minimize total transportation time. The effectiveness of the proposed method is demonstrated with a practical-sized problem. C. Tolba et. al. proposed an approach in [55] that represents all the variables of traffic flow with continuous Petri nets with variable speed. The proposed Petri net model is suggested for the analysis of both motorway and road junctions. L. Zhang et. al. [56] used hybrid Petri nets in their work to specify and control traffic at an intersection. With the assistance of some other models, the system is proved to be suitable to perform traffic optimization between two adjacent intersections. In [57], B. Huang et. al. proposed a hybrid model of extended fluid stochastic Petri net for urban traffic systems. The discrete part of the system is used to model the decision making and traffic lights control part. The illustrative example presented showed that the model is effective.

1.2.2 Petri Net Estimation and Optimization

The applications of Petri net mentioned above show that the Petri net is a well-accepted tool in practical systems, such as the microgrid systems and traffic systems. In our research, the estimation and optimization algorithms are also related to these fields. The objective of the optimization of the Petri nets could be structure-wise, such as the least transitions and connections, which can imply the smallest number of devices and parts in a practical system; or marking-wise, such as the minimum number of initial tokens, which could represent the cost of input resources in industrial production. These optimization operations will significantly save time and resource costs in transportation or manufacturing. In literature, researchers have been studying the Petri net structure and marking optimization problem based on external observation [58]– [62]. Before we could study the optimization of Petri nets in detail, some more similar studies that relate to our topics will be introduced and we would like to point out the similarities and differences.

Reconstruction of Unknown Petri Net Structures from Asynchronous Observations of Token Change Sequences

Because the behaviors in the modern systems become increasingly complex, researchers in recent studies paid significant attention to the problem of system identification based on external observation of underlying system behavior, which includes the transition firing sequences, the label sequences, the token changes, etc. For example, in [63] approaches based on integer programming have been proposed in order to identify Petri net structure and the initial marking from the observed finite language, which assumes the full knowledge of the transitions and the places. Authors in [64] have extended the previous approaches by using the linear programming approach. Based on the observation of an event sequence that generates corresponding output vectors, the studies in [65] and [66] discussed the problem of identification of the Petri net structure, where the integer linear programming was utilized. We can also obtain a summary of the research related to this direction in [67] and [68].

In this part of the dissertation, a similar problem will be considered, which aims at system estimation and reconstruction but with a different observation, assumptions. In particular, the objective is to estimate the Petri net structure(s) based on the observed sequences of token changes at each place. More specifically, we assume that the information of token changes of each place in the Petri net can be provided individually. After receiving information regarding the token changes, the task is to estimate the Petri net structure(s) that is(are) consistent with the observed token changes sequences. However, since the sequences are asynchronous, we are unaware of neither the order of the token change nor the transition firing sequence. Therefore, the external observation can only provide partial information about the state evolution of this Petri net. The study in [69] used a similar setup but has a different research purpose, which is to reconstruct all possible transition firing sequences with the full knowledge of the Petri net structure. The proposed algorithm in [69] could reconstruct all valid transition firing sequences, with relatively high computational complexity. Authors in [70] extended the previous work, which slightly reduced the complexity by adding the *counting places*.

Minimum Initial Marking Estimation in Labeled Petri Nets with Unobservable Transitions

When studying the Petri net, the estimation of the markings or states is an important and fundamental problem, which has the objective to estimate the state of the system modeled by a Petri net structure with external observations. In [71], an application has been proposed related to Petri net deadlock prevention, where the Petri net structure contains unobservable transitions. The authors in [72] studied a marking estimation problem in an unlabeled P-time Petri net. The unobservable transitions were also involved. Only partial observation from the state observer could be obtained in the proposed approach. The study in [73] also applied the scheme of the observer for the estimation of both the markings and the firing vectors. The authors in [74], [75], [76], and [77] introduced several techniques of the Petri net marking estimation based on external label observations. With proper assumptions, these techniques were not affected by the length of the observation. Authors in [78], [79] utilized a probabilistic setting and proposed an marking estimation approach. A finite set of initial markings were assumed to be known with priori probabilities. Given the label observations, the goal was to obtain the marking estimates with conditional probabilities.

Although the studies mentioned above have a significant contribution to the Petri net marking estimation area, most of them assumed that they have full or partial knowledge of the initial marking of the system. On the other hand, in the minimum initial marking estimation problem that we would like to study in this dissertation, the objective is different, which is to estimate the initial marking set based on the observed label sequence that is caused by the system's underlying transition firing activities. Besides, in our study, some transitions in the Petri net are unobservable, which means that the firing of these transitions can occur without being observed. The study of this problem is inspired by the problem of initial resource allocation during the process of product manufacturing, where the proper planning and scheduling should be done with the least resources to complete a set of necessary pre-set processes or tasks [80].

There are also studies related to similar topics. For example, in [81], the authors proposed a heuristic method for the minimum initial marking estimation. In the study, it was assumed that we have full knowledge of the structure and transition firings, but each transition could only be fired for limited times. The objective was also to obtain the initial marking with the least token sum. Authors in [82], [83], and [84] studied the minimum initial marking estimation problem in labeled Petri nets, which have observable transitions only, and proposed algorithms to estimate the minimum initial markings.

1.2.3 Other Practical Use of Petri Net

There are also other practical applications of the Petri net, which are also important topics that many researchers have been studying. In [85], the authors used a device called middleware in the sensor network, where the structure of the network could be represented by the Petri net. The devices in this network could be denoted by places and transitions, while the connections of the network could be represented by the arcs. In [86], the authors talked about the application of the Petri net in the area of the process flow. In this study, the process of the system has been modeled with the Petri net, where places, transitions, arcs represent the states of the system, the actions that cause state evolution, and the connections between states and actions, respectively. The initial state of the Petri net had one token in the place named *Ready*. During the work process of the system, the token will be moved from state to state and reach the final place called *End Session* at the end, which marked the end of the process flow. The study in [87] applied the Petri net to detect the failure components in a system. The places and transitions were used to represent the failure modes and the conditions that cause the failure modes, respectively. Since the failure modes and the causes were undetectable, the authors also did structure identifications in their study. D. Li et al. [88] modeled the reversion losses and shoot-through currents in switched-capacitor DC-DC converter with Petri net structure. The Petri net models are proved to be reliable for their work. In [89], the complex flexible manufacturing systems were modeled with the Petri net. The author has used the traditional Petri net for the simulation with Pinvariants to be the supervisor and used the timed Petri net for the performance evaluation. In [90], the authors applied the Petri net into the networked control system, which is used to communicate and exchange data via a network. A modified Petri net structure called Colored Petri net is used in their work. Z. Ding et al. in [91] also used a modified Petri net structure called stochastic-differential Petri net to model the switched stochastic system, which is used to describe hybrid systems with randomness. The proposed model has been examined with a checking technique and proved to be correct and meet the requirements.

The previous work and studies of the practical applications of the Petri net could be valuable references for our study in this dissertation, where we could still help to improve their results and extend the conclusion based on our researches. In later chapters of this dissertation, we will talk about the algorithms that are used for the estimation, optimization, and control of certain systems, which could be modeled with the Petri net structures. In the next part of the introduction, we will briefly talk about the major contributions of each chapter.

1.3 Major Contribution

Traffic Modeling

In this dissertation, our first objective is to discretize the traffic network with *DEDS* (mainly Petri net). The idea of road discretization is not new. However, different approaches have been used for different applications. There is still much room to model and simulate the traffic system in a more complete way. In this dissertation, we proposed two different ways of traffic modeling with the Petri net. Firstly, we propose a probabilistic Petri net model for studying the driving behavior of vehicles on each lane of a certain highway segment. This method is quite general and can be applied to different highways with different geometries and traffic volume. In particular, the contributions of this part are summarized as follows: 1) The highway segment is discretized into blocks that are homogeneously distributed in terms of distance. 2) Driving behavior data of vehicles are classified in terms of time. The original driving dataset is studied with a reasonable sampling rate. 3) The Probabilistic Petri net model is developed by fitting the driving behavior data into the discretized highway blocks. The movement of vehicles will follow a certain probability assigned by the model according to the traffic data available.

Following that, in the second way, we are modeling a relatively larger traffic network around a city rather than only a segment on the highway with the traditional Petri net structure. The discretization techniques that we have been using are different from common ways, which discretize the traffic network uniformly and study the traffic dynamic within a small area. We study the route planning problem on a large scale of traffic networks around a major U.S. city. The scenario design is inspired by the daily driving case, which starts at home and ends at the workplace. During the way, some destinations of interest, such as a supermarket or a gas station will be visited. Many research directions could be yielded from this scenario. For example, how to find the best alternative way, which passes by the least destinations of interest; or how to minimize the traveling time if we want to start from home and end at the workplace and visit several certain destinations of interest. Thus, the study aims at saving travel time and improve driving safety. It is a very practical and useful topic in our daily life.

When converting the traffic network into a Petri net structure, each place in the Petri net represents a place of interest (*POIs*). Thus, the time cost of traveling between places will be different. Each token in this Petri net represents one unit time. Hence, the time cost of traveling between places will be reflected by the reduction of different amounts of tokens when firing a transition. The arc weights will show how many tokens will be consumed, which depends on both the distance between two places and the traffic condition. Note that in order to simplify the problem, only one direction of the roads is considered, which aims to remove the loops in the Petri net. The direction of the roads depends on the location of the starting place and the finishing place. Besides, since only highway/freeway and major local roads are considered when modeling the traffic network, it is possible that there exist alternative routes that are better in some sense, which makes the optimization of the Petri net structure a reasonable topic.

Minimum Initial Marking Estimation in Labeled Petri Nets with Unobservable Transitions

In this part of the dissertation, the objective is to extend the work in [84] to the labeled Petri net structures that have unobservable transitions, which means to deal with the problem of minimum initial marking estimation in labeled Petri nets with unobservable transitions (*MIM-UT*). The *MIM* estimation could be used to simulate the industrial production, which follows a sequence of certain production processes and to minimize the required input resources. In previous work in [84], the MIM estimation problem with only observable transitions has been solved. By applying some techniques, the redundant markings are removed so that the algorithm could reach complexity that is polynomial in terms of the length of label observations.

However, the problem would become more challenging with unobservable transitions. Before each observation label, there could be an unobservable transition fired. In our work, we could show that with some proper assumptions of the structure of an unobservable subnet, it is possible to find the set of *MIM-UT* estimates through an algorithm that has a complexity that is polynomial in terms of the length of the observed label sequence. We provided an illustrative example in a later chapter to show that this algorithm can work properly and generate the exact solution set.

Moreover, we have proposed two heuristic algorithms that could reduce the computational complexity. Although the heuristic algorithms cannot give the most accurate and comprehensive solutions, it is a trade-off between resource consumption/time cost and accuracy. However, for practical use, most of the time we don't need the exact solution, which needs more time to find. A sub-optimal solution is also acceptable when it cost much less computation resource. The provided illustrative example is also used to evaluate and compare the performance of the heuristic algorithms with the main algorithm.

Reconstruction of Unknown Petri Net Structures from Asynchronous Observations of Token Change Sequences

Our objective in this part is quite different from the other researches and is to estimate the Petri net structure based on the observations of the token change sequence in each place. As mentioned above, we could obtain information including the initial marking, the final marking, and the number of places. An algorithm has been developed, which is able to estimate the optimal structure of the Petri net structure that has the least transitions and the most sparse incident matrix, based on the token change observations. The length of the sequences is assumed to be continuously growing. However, when the observation sequence is extracted at a certain time stamp and to be used it as the input of an algorithm, the length of the extracted observation snippet is fixed, regardless of what will be generated later. This feature could be used so that only the current observation snippet is focused to save the computation resource.

Since the input observation sequences are with constant length, the work of scanning markings, firing vectors, and solving state equations have become simpler. With the constant length of given sequences, A finite number of markings could be obtained from the observed token change sequences and could have a finite number of transitions and the upper and lower bounds of the number of transition firings. Then we need to know the bounds of markings and firing vectors. They will be used to determine the breaking conditions of the scanning loops. Both of them could be found with mathematical methods.

Furthermore, another algorithm has been developed for consistency check, which is to examine whether the obtained Petri net structure from the previous algorithm could still be consistent with the future observation sequences with finite-length. As mentioned above, with the growth of the observation sequences, the estimated Petri net structure from the previous algorithm could be potentially not consistent with the observation. Because of that, we first do the consistency check to the estimated Petri net with the new observation. If the consistency check fails, a new estimation is needed to be calculated. It could be shown that the consistency check algorithm can run with constant complexity.

Traffic Volume Control based on Observed Label Sequences

A traffic network in a certain area is modeled with the labeled Petri net and then control the traffic flow in order to control the traffic volume and reduce the rate of collision. Although there exists some similar work that also used the Petri net as the modeling tool and applied control method, they either used the Petri net structure that was too simple or didn't consider enough elements that could occur in real traffic conditions. In my work, the traffic network in a certain area is modeled with the Petri net that contains both observable and unobservable transitions, where observable transitions represent traffic intersections on main roads that normally have traffic controllers assigned, like traffic lights and signs, and unobservable transitions represent junctions that connect small roads with the main traffic network, which possibly don't have any control method to guild the drivers. Because of the existence of the unobservable transitions, it will be more challenging to design a controller to achieve our objective, which also makes our work valuable and be an extension and supplement of the state-of-art studies.

1.4 Organization

This dissertation is organized as follows: After the introduction in Chapter 1, Chapter 2 introduces necessary mathematical notations and background knowledge of the Petri net structure; Chapter 3 shows two methods of traffic modeling based on two different Petri net structures for different network scales and research purposes; Chapter 4 talks about the algorithm of minimum initial marking estimation in labeled Petri nets with unobservable transitions with its application in traffic planning; Chapter 5 introduces the second algorithm that is used to estimate and reconstruct the Petri net structure based on the observed token change sequences in the places; Chapter 6 proposes a control algorithm that is used to limit the traffic volume based on labeled Petri net with unobservable transitions, aiming to reduce congestion and collision. The algorithm is also based on external label sequence observation. At last, the summary and conclusion are given in Chapter 7.

2. PETRI NET PRELIMINARIES

The notation and preliminary definitions that will be used in this dissertation will be introduced in this chapter. In the 1960s, Carl Adam proposed the Petri net, which models and describes the distributed systems with a mathematical language. Petri net has the advantages of intuitive graphical expression and rigorous mathematical formulation. Both of the advantages make the Petri net an excellent tool for solving practical problems. The target systems could be expressed with intuitive mathematical formulation, which would be more convenient for problem-solving. [40] and [41] provide more specific information about the Petri nets.

2.1 Basic Knowledge of the Petri Net

2.1.1 Structure

A Petri net structure is a weighted bipartite graph $Ne = (Pl,\ Tr,\ Ar,\ We)$. The finite set of *places* is denoted by $Pl = \{p_1,\ p_2,\ \ldots,\ p_n\}$, where the places are drawn as circles. The finite set of *transitions* is denoted by $Tr = \{t_1,\ t_2,\ \ldots,\ t_m\}$, where the transitions are drawn as bars. The set of arcs between places and transitions is denoted by $Ar \subseteq (Pl \times Tr) \cup (Tr \times Pl)$ and the *weight function* on the arcs is denoted by $We : Ar \rightarrow \{1,\ 2,\ 3,\ \ldots\}$. Fig. 2.1 shows an example of a simple Petri net:

We could represent the relationship between places and transitions by notations. The input transition set for a p_i or input place set for a t_i could be represented with ${}^{\rightarrow}p_i$ or ${}^{\rightarrow}t_i$, where $p_i \in Pl$ and $t_i \in Tr$. The set of output transition set for a p_i or output place set for a t_i could be represented with $p_i^{\rightarrow}$ or $t_i^{\rightarrow}$. Also, ${}^{\rightarrow}p_i^{\rightarrow}$=${}^{\rightarrow}p_i \cup p_i^{\rightarrow}$ (${}^{\rightarrow}t_i^{\rightarrow}$=${}^{\rightarrow}t_i \cup t_i^{\rightarrow}$) could represent both input and output transitions or places for p_i or t_i.

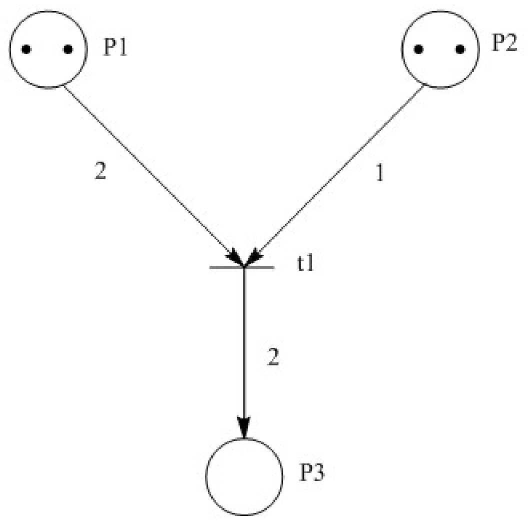

Fig. 2.1. A Simple Petri Net.

Example 1: Consider the Petri net in Fig. 2.1. Place set, transition set, arcs and weights are denoted by $Pl = \{p_1, p_2, p_3\}$, $Tr = \{t_1\}$, $Ar = \{(p_1 \times t_1), (p_2 \times t_1), (t_1 \times p_3)\}$ and $We(p_1,t_1) = 2$, $We(p_2,t_1) = 1$, $We(t_1,p_3) = 1$, respectively. The relationship between places and transitions are represented by $p_1^{\rightarrow} = \{t_1\}$, $p_2^{\rightarrow} = \{t_1\}$,$^{\rightarrow}p_3 = \{t_1\}$, $^{\rightarrow}t_1 = \{p_1, p_2\}$, $t_1^{\rightarrow} = \{p_3\}$.

2.1.2 Marking

The vector $Mk : Pl \rightarrow Z^+$ denotes the *marking*, which assigns a non-negative integer number of tokens to each place of the Petri net. The tokens are drawn as black dots. The initial marking of the Petri net is denoted by Mk_0. Thus, a Petri net system is denoted by $< Ne, Mk_0 >$. The number of tokens in place p, i.e., the marking of place p, is denoted with $Mk(p)$.

Example 2: Consider the Petri net shown in Fig. 2.1. The number of tokens that are assigned to place p_1, p_2, and p_3 are two, two, and zero. Hence, the marking of this Petri net is $Mk_0 = [2,\ 2,\ 0]^T$.

2.1.3 Incident Matrices

The *input incident matrix* $In^- = [in^-_{ij}]$ (respectively the *output incident matrix* $In^+ = [in^+_{ij}]$) is defined to be a matrix with dimension n by m and in^-_{ij} (respectively in^+_{ij}) at its (i, j) position. Here in^-_{ij} denote the integer weight of the arc from place p_i to transition t_j, and in^+_{ij} denote the integer weight of the arc from transition t_j to place p_i ($1 \leq i \leq n$, $1 \leq j \leq m$). The difference $In \equiv In^+ - In^-$ is defined to be the *incident matrix* of the Petri net. Note that if there is no arc connecting place p_i and transition t_j, the value of in^-_{ij} or in^+_{ij} is set to be zero.

Example 3: The incident matrices of the Petri net in Fig. 2.1 are shown as follows:

$$\begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix} \tag{2.1}$$

From top to bottom, the three matrices are input incident matrix In^-, output incident matrix In^+, and incident matrix In for the Petri net shown in Fig. 2.1, respectively.

2.1.4 Enabling and Firing Transitions

In order to *enabled* a transition t, each of its input places p should have at least $In^-(p,t)$ tokens. Notation $Mk[t\rangle$ is used to denote that t is enabled at marking Mk. If t is enabled, it may *fire*. When it fires, $In^-(p,t)$ tokens are removed from each of its input place and $In^+(p,t)$ tokens are deposited to each of its output place, which generates a new marking $Mk' = Mk + In(:,t)$. Here $In(:,t)$ denotes the column of In that corresponds to t. This firing action could also be denoted by $Mk[t\rangle Mk'$, which also means that marking Mk' is reachable from initial marking Mk by firing transition t.

Example 4: Consider the Petri net in Fig. 2.1. We know the initial marking and the input incident matrix are $Mk_0 = [2,\ 2,\ 0]^T$ and $In^- = [2,\ 1,\ 0]^T$from the previous sections. Because we have $Mk_0 \geq In^-(:,t_1)$, transition t_1 could be enabled.

Example 5: Consider the Petri net in Fig. 2.1. Since t_1 is enabled by the marking $[2,\ 2,\ 0]^T$, we could fire it. During the firing, two tokens will be removed from place p_1 and one token will be removed from place p_2 according to $In^-(:,t_j) = [2,\ 1,\ 0]^T$. After that, two tokens will be deposited to place p_3 according to $In^+(:,t_j) = [0,\ 0,\ 2]^T$. So after t_1 fired, $Mk_1 = [0,\ 1,\ 2]^T$. This process is denoted by $Mk_0[t_1\rangle Mk_1$.

2.1.5 Firing Sequence, Firing Vector, and State Equation

The transition firing sequence is denoted by $\delta = t_{i1}t_{i2}\ldots t_{ik}$ $(t_{ij} \in Tr)$. The sequence δ is said to be enabled with respect to Mk if $Mk[t_{i1}\rangle Mk_1[t_{i2}\rangle \ldots Mk_{k-1}[t_{ik}\rangle$ where $Mk_j \geq 0$ $(j \in \{1,2,\ldots,k-1\})$ denote a set of markings in the net. This process is denoted by $Mk[\delta\rangle$. Notation $Mk[\delta\rangle Mk'$ is used to denote the firing of δ from Mk yields Mk'. Meanwhile, $|\delta(t)|$ is used to represent the total number of occurrences of transition t in δ. More specifically, the *firing vector* that corresponds to δ is denoted by $vf = [|\delta(t_1)|\ \ |\delta(t_2)|\ \ \ldots\ \ |\delta(t_m)|]^T$. Note that after firing an enabled sequence δ from marking Mk, the new marking Mk' can also be computed from the state equation as follows.

$$Mk' = Mk + B \cdot vf(\sigma). \tag{2.2}$$

Example 6: Consider the Petri net in Fig. 2.2.

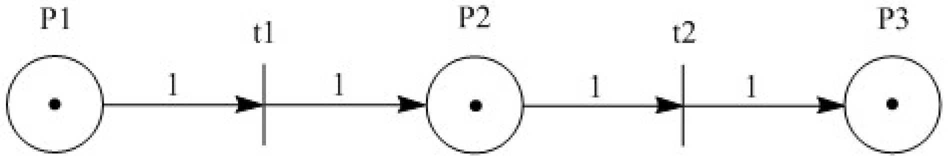

Fig. 2.2. The Petri Net Used in Example 6.

The incident matrix is:

$$\begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}$$

At stage 0, the initial marking is $Mk_0 = [1\ 1\ 1]^T$. Therefore, transitions t_1 and t_2 could be fired. Suppose $Mk_1 = [0\ 0\ 3]^T$ is the marking at stage 1, which means that t_1 has been fired once and t_2 has been fired twice. The corresponding firing vector $vf = [1\ 2]^T$. Thus, the state equation of the process should be:

$$\begin{bmatrix} -1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix}$$

From the example above, we could see that the entries of the firing vector vf are actually the number of firing times of every transition, which means that the vf vector could also be written as the sum of all v_is (i=1,2,....), where v_i is firing vector of single firing action with only one nonzero entry. In example 6, $vf = v_1 + v_2 + v_2 = [1\ 0]^T + [0\ 1]^T + [0\ 1]^T$, where v_1 shows that one firing of t_1 and v_2 shows one firing of t_2.

2.1.6 Reachability Graph

There existing other Petri nets with more complex structures, which would have many more firing options that are hard to be analyzed. In that case, we use a scheme called the *reachability graph* to enumerate all the possible firing action branches.

The reachability graph consists of the reachable markings from the initial marking Mk_0 and the corresponding fired transition sequences. Because of that, to study a finite Petri net, we only need to know its reachability graph instead of it structure. Following that, other mathematical methods like the state equation would be applied for problem-solving.

Example 7: Consider the Petri net in Fig. 2.3. Its reachability graph or reachability tree has been given in the following Fig. 2.3:

The graph in Fig. 2.3 consists of the reachable markings, which are directed with arcs marked with the fired transitions, where each arc shows a potential firing branch. For instance, from the initial marking $Mk_0 = [1,\ 1,\ 1]^T$to the marking $Mk_1 = [0,\ 2,\ 1]^T$, transition t_1 need to be fired. Alternatively, from the initial marking $Mk_0 = [1,\ 1,\ 1]^T$ to the

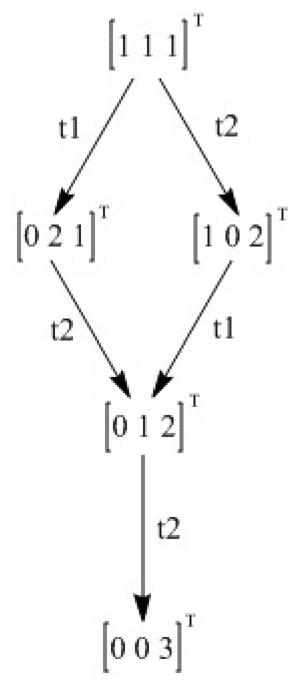

Fig. 2.3. The Reachability Graph for the Petri Net in Fig. 2.2.

marking $Mk_2 = [0,\ 0,\ 3]^T$, the fired transitions could form a firing sequence, which could be either $t_1t_2t_2$ or $t_2t_1t_2$.

As we can see, in a reachability graph of a Petri net, sometimes we could have more than one firing sequence from initial marking Mk_0 to another marking Mk_i. Although we have multiple firing sequences, the corresponding firing vector vf will be the same, since we have the same incident matrix In and the same value of $Mk_i - Mk_0$. Hence, the state equation will generate the same solution for vf. This feature could be very important in practical problem-solving in this dissertation.

2.1.7 External Observation

Besides the structural parameters of the Petri net mentioned in the previous sections, several external observed terms that are also important to our research in this dissertation. There are two kinds of external observations used in this dissertation: token change observation and label observation.

Token Change Observation

First, we introduce the token change observation sequences, which are defined as:

Definition 1: [69] Let $\mathscr{S}^n$ denote the space of sequences of markings in $(Z^+)^n$ and define $\Gamma_Q : \mathscr{S}^n \to \mathscr{S}^Q$ be the *projection* that focuses on the sequence of marking changes at places indexed by the set Q *and* also removes repeated elements in the sequence.

For example, the projection Γ_{p_i} of the sequence of markings $Mk[0], Mk[1], \ldots, Mk[k]$ at the i^{th} place is given by:

$$\Gamma_{p_i}(Mk[0] \to Mk[1] \to \ldots \to Mk[j_1] \to \ldots \to Mk[j_k] \to \ldots \to Mk[k]) =$$
$$Mk(p_i)[0] \to Mk(p_i)[j_1] \to Mk(p_i)[j_2] \to \ldots \to Mk(p_i)[j_k],$$

where $\{j_1, j_2, \ldots j_k\}$ gives the time epoch set, where at each time epoch the the number of tokens in place p_i changes. More specifically, $\{j_1, j_2, \ldots j_k\}$ satisfies

$$Mk(p_i)[0] = Mk(p_i)[1] = \ldots = Mk(p_i)[j_1 - 1] \neq Mk(p_i)[j_1]$$

$$Mk(p_i)[j_1] = Mk(p_i)[j_1 + 1] = \ldots = Mk(p_i)[j_2 - 1] \neq Mk(p_i)[j_2]$$

, and so forth.

Suppose that the token change in each place is reported separately. The observed token change sequence at each place p_i ($i \in \{1, 2, \ldots, n\}$) is denoted by $s_i = Mk(p_i)[0] \to Mk(p_i)[j_1] \to Mk(p_i)[j_2] \to \ldots \to Mk(p_i)[j_k]$ ($0 < j_1 < j_2 < \ldots < j_k$). The initial (final) number of tokens in place p_i is denoted by $Mk(p_i)[0]$ ($Mk(p_i)[j_k]$). The symbol S is used to denote the set of observed sequences of token changes, i.e., $S = \{s_1, s_2, \ldots, s_n\}$.

Definition 2: [69] Suppose $\delta = t_{i1}t_{i2}\ldots t_{ik}$ is a transition firing sequence and we have $Mk_0[t_{i1}\rangle Mk_1[t_{i2}\rangle \ldots [t_{ik}\rangle Mk_k$, where $Mk_j \geq 0$ ($j \in \{1, 2, \ldots, k\}$) represents a marking sequence. δ is said to be a *consistent transition firing sequence* with respect to the observed token change sequence s_i of place p_i if it satisfies $\Gamma_{p_i}(Mk_0 \to Mk_1 \to Mk_2 \to \ldots \to Mk_k) = s_i$. Similarly, suppose we have the observed token change sequence set $S = \{s_1, s_2 \ldots s_n\}$, δ is also defined as a *consistent transition firing sequence* with respect to S if it is consistent with each sequence s_i ($i \in \{1, 2, \ldots, n\}$).

Example 8: Consider the Petri net shown in Fig. 2.4 with place set $Pl = \{p_1, p_2, p_3\}$, transition set $Tr = \{t_1, t_2\}$. The initial marking is set to be $Mk_0 = [1,\ 1,\ 1]^T$.

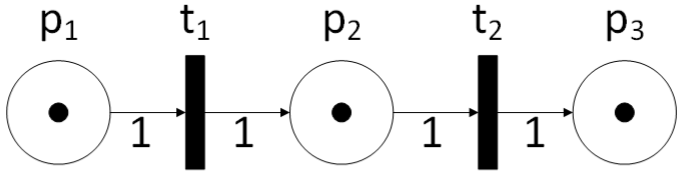

Fig. 2.4. The Petri Net Used in Example 8.

We are given the set of observed token change sequence set $S = \{s_1, s_2, s_3\}$ as follows:

$$\begin{aligned} s_1 &: \quad 1 \to 0, \\ s_2 &: \quad 1 \to 2 \to 1 \to 0, \\ s_3 &: \quad 1 \to 2 \to 3. \end{aligned}$$

In this case, it is clear that $t_1t_2t_2$ is the only legal transition sequence that is consistent with the observations in S. The marking evolution under $t_1t_2t_2$ is given by Eqn. 2.3. It can also be verified that the number of token changes at each place (after projection Γ_{p_i} in each place p_i) satisfies the conditions in Eqn. 2.4 to Eqn. 2.6.

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \xrightarrow{t_1} \begin{bmatrix} 0 \\ 2 \\ 1 \end{bmatrix} \xrightarrow{t_2} \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix} \xrightarrow{t_2} \begin{bmatrix} 0 \\ 0 \\ 3 \end{bmatrix} \tag{2.3}$$

$$\Gamma_{p_1}(Mk_0 \to Mk_1 \to Mk_2 \to Mk_3) = s_1, \tag{2.4}$$

$$\Gamma_{p_2}(Mk_0 \to Mk_1 \to Mk_2 \to Mk_3) = s_2, \tag{2.5}$$

$$\Gamma_{p_3}(Mk_0 \to Mk_1 \to Mk_2 \to Mk_3) = s_3. \tag{2.6}$$

Observation of the Label Sequence

The other external observation used in this dissertation is the observation of the label sequence. First, we would like to introduce the labeled Petri net structure. A labeled Petri net structure is denoted by $Ne_L = (Pl, Tr, Ar, We, Labels, \Delta)$, where $N = (Pl, Tr, Ar, We)$ is a Petri net structure and the *labeling function* $Labels : Tr \to \Delta \cup \{\varepsilon\}$ assigns each transition a given alphabet label from the set Δ, or the empty label ε if the transition is unobservable. Note that a label could be shared between two or more transitions. For a label $l \in \Delta$, we use Tr_l to denote the set of transitions with label l, and $|Tr_l|$ to denote the total number of transitions associated with label l.

In this dissertation, we will consider the labeled Petri net structure with unobservable transitions. Therefore, the transition set Tr is divided into two sets: observable subset Tr_o and unobservable subset Tr_u, where $Tr = Tr_o \cup Tr_u$ and $Tr_o \cap Tr_u = \emptyset$. When plotted, Tr_o is represented with empty bars and Tr_u is represented with solid bars.

Definition 3: Given a labeled Petri net with unobservable subset $Tr_u \subseteq Tr$. The *unobservable subnet* of the given labeled Petri net is defined as the net $Ne_{Pu} = (Pl,\ Tr_u,\ In_u^-,\ In_u^+)$, where In_u^- and In_u^+ consist of the columns that correspond to unobservable transitions in the input and output incident matrices In^- and In^+.

Definition 4: [92] Two transitions t_i and t_j are said to be contact-free, if they satisfy the following conditions:

$$^{\to}t_i^{\to} \overset{\to}{\bigcap} t_j^{\to} = \emptyset,\ ^{\to}t_i \bigcap t_i^{\to} = \emptyset,\ and\ ^{\to}t_j \cap t_j^{\to} = \emptyset$$

i.e., the two transitions do not share any input or output places and don't have self-loops.

In this dissertation, all unobservable transitions in the Petri net are assumed to be *contact-free*. The Petri net in Fig. 2.5 with its unobservable subset in Fig. 2.5.(b) has been provided. Note that the unobservable transitions t_5 and t_6 are contact-free.

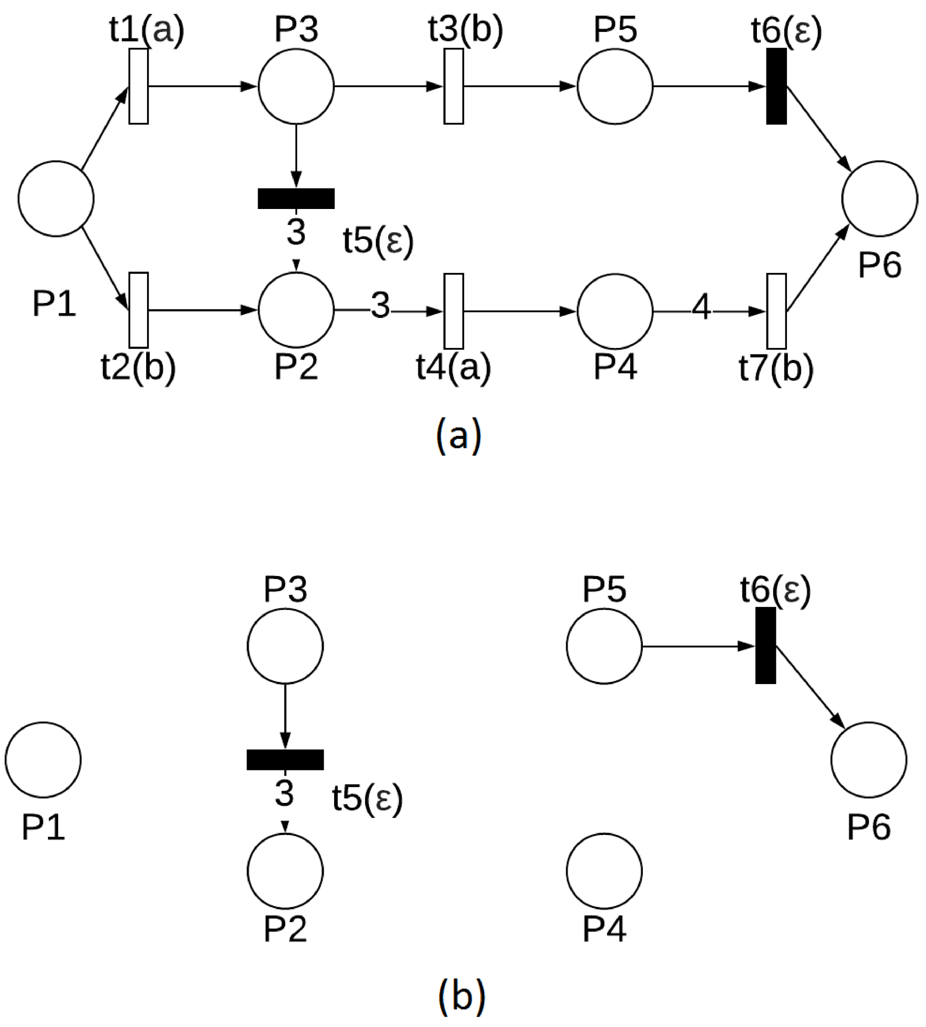

Fig. 2.5. A Labeled Petri Net (a), and its Unobservable Subnet (b).

2.1.8 Probabilistic Petri Net

Based on the study in [93], the probabilistic Petri nets have similar structures with regular Petri nets except the firing of each transition is associated with a probability, which captures its likelihood to occur. In particular, a probabilistic Petri net is a five-tuple $N_{pp} = (Pl, Tr, Ar, We, prob)$, where Pl, Tr, Ar, and We are the same as regular Petri nets, the probability function $prob$ is defined as $prob: \ Tr \rightarrow [0,\ 1]$, which assigns each transition a number between 0 and 1 for its likelihood to occur if it is enabled.

Fig. 2.6 shows an example of a probabilistic Petri net structure. Note that if all transitions share the same input place, the sum of probabilities of all these transitions is one. In Fig. 2.6, for instance, $prob_1 + prob_2 + prob_3 = 1$.

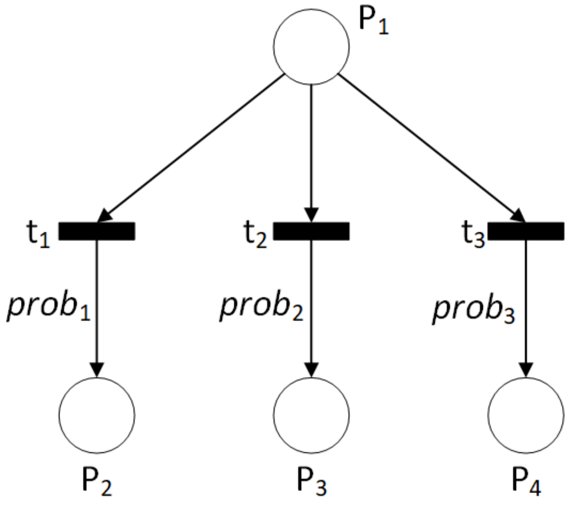

Fig. 2.6. An Example of a Probabilistic Petri Net.

3. HIGHWAY TRAFFIC MODELING WITH THE PETRI NET

In this chapter, two different traffic modeling schemes have been introduced. The first one applies the Probabilistic Petri net structure to a highway period, which discretizes the road into blocks, and movement of the traffic has been assigned with probabilities. The second one models the traffic in a larger area with the traditional Petri net structure, which focuses on the destinations of interest (*DOI*) during the traveling path. These modeling schemes make the first step of the study in this dissertation.

The first section of this chapter will introduce the modeling scheme with the probabilistic Petri net while the second section will be focused on the modeling scheme based on the traditional Petri net. Detailed modeling procedures and necessary parameters will also be discussed in each section.

3.1 Highway Traffic Modeling with the Probabilistic Petri Net

In this section, a novel method for modeling the highway traffic using Probabilistic Petri nets (PPNs) is proposed. More specifically, the highway has been partitioned into discrete segments and probabilistic measures are derived based on the traffic data considering the vehicle movements. The proposed model is validated through the study of a publicly available dataset called Active Transportation Demand Management (ATDM) Trajectory Level Validation, which provides the real traffic data from different driving scenarios. The proposed method will generate a graphical structure of the PPN as well as all the important attributes related to the real traffic data. The output model can be used for path planning and collision avoidance for highway traffic.

3.1.1 Data Filtering

The dataset we study is called ATDM Trajectory Level Validation [95] by the U.S. Department of Transportation, which provides driving data on local roads and highway. It includes a variety of data attributes such as *datetime*, *latitude*, *longitude*, *speed*, and *laneID*. The data attributes needed for our analysis are *latitude*, *longitude*, *speed*, *laneID*, and *testing datetime*. Note that the recorded data includes data points for both local and highway scenarios. Since we are only interested in the driving scenarios on the highway, the data points with inappropriate speed will be removed.

Furthermore, the fast data sampling rate of the original dataset is not suitable for our case because the short period of time will not generate much difference in vehicle position and speed. Hence, we need to find an appropriate sampling ratio. From the work in [50], the suggested time interval is 1 to 3 seconds. In our work, we select the lower bound of the suggested values, which is 1 second.

3.1.2 Highway Segment Discretization

Since the original database does not provide the detail of the driving route, road geometry, and traffic conditions, there are two key assumptions in this study: A1) Only straight highway geometry is considered; A2) There is no heavy traffic or congestion that will significantly affect the driver's intention. The reason of the assumptions is for our calculations. Specifically, we choose to accumulate the traveling distance between data points with latitudes and longitudes to estimate the length of the highway segment. The estimation is valid because of the short time gap between two consecutive data, regardless of highway geometry. Under these two assumptions, it is feasible to determine the total length of the highway. Calculating the distance that each vehicle traveled between two data points is straightforward. Equations 3.1–3.3 below show the Haversine formulas [96], which is used to calculate the distance d with latitude and longitude:

$$a = sin^2(\frac{\phi_{diff}}{2}) + cos\phi_1 \cdot cos\phi_2 \cdot sin^2(\frac{\lambda_{diff}}{2})) \tag{3.1}$$

$$c = 2 \cdot atan2(\sqrt{a}, \sqrt{1-a}) \tag{3.2}$$

$$d = R \cdot c \tag{3.3}$$

where ϕ_1, ϕ_2, ϕ_{diff}, λ_{diff}, and R represent the two latitudes, the difference of the latitudes (in radians), the difference of the longitude (in radians), and the radius of the earth, respectively. Furthermore, we can find the longest data segment from the filtered dataset. The total length of the highway could be considered as the accumulated distance of the longest data segment.

The average length of common compact cars in the U.S. is from 4.3 to 4.6 meters, while the larger-sized cars could be 0.2 to 0.5 meters longer. Hence, it is reasonable to make the length of each discretized segment to be 10 meters, which is about the length of two vehicles. The distance is practical since it is close enough for the crash imminent braking (CIB) system to be triggered (CIB triggering distance is from 5 to 15 meters as suggested in [5]), while still leaves room for the following vehicles to avoid a potential collision. With the information on the total length of the highway of interest and the length of the proposed discretized segment, it is not difficult to create the discretized model.

Definition 5: A real number is assigned to each transition that represents its *Reward*, denoted by r.

The *Reward* is used to represent the level of occupation of the following place. Different from *prob*, *Reward* is positively correlated with the emptiness of the following place, hence driving safety, which could be more intuitive than *prob*. The value of *Reward* is calculated with the probability *prob* that one vehicle transiting from one block to another, as shown in Equation 3.4.

$$r = -log_{10}^{prob} \tag{3.4}$$

The method to obtain the *prob* values for all transitions is through the prepared dataset. It is simple to know the place that each data point belongs to since we have calculated accumulated distance. Therefore, we have the distribution of the whole dataset in all the places. Then, the *prob* values are calculated with the *Monte Carlo* method. Starting from one certain place, excluding the ones with zero probability, we can have the set of places

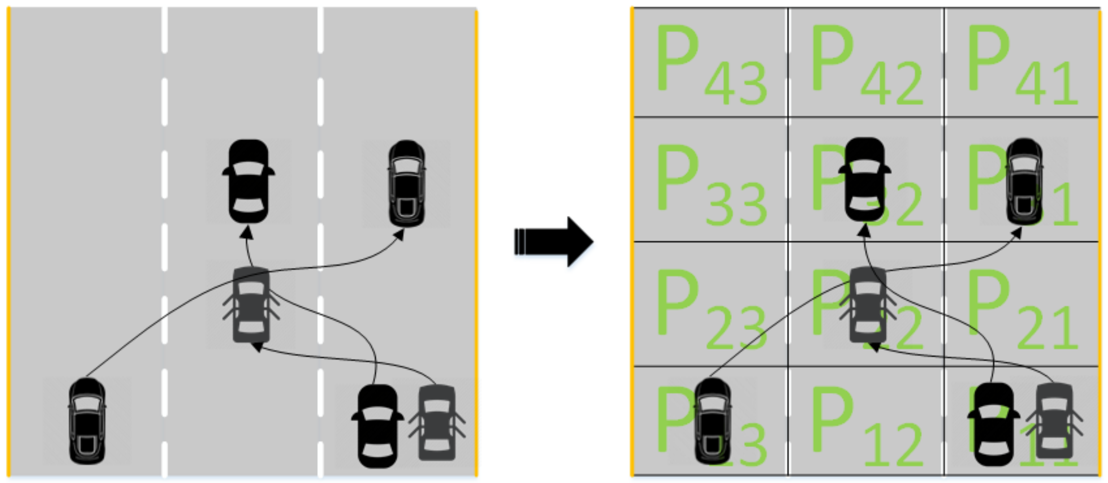

Fig. 3.1. Highway Segment Discretization.

that the token would be transited to. By doing statistics, the probabilities of transiting to these ”son” places could easily be obtained.

Example 9: Consider a highway segment with three lanes. Fig. 3.1 shows the way of the discretization of it:

The discretized highway segment could generate the Probabilistic Petri net that we need. As shown in Fig. 3.1, each block is treated as a place in the Petri net. Vehicles driving on the highway is considered to be the tokens. Transition t_1, t_2 and t_3 will be used to connect places. We assume there are two data points in place p_{11} and one data point in place p_{13}. Therefore, the probability value $prob_1 = prob_2 = 0.5$, since two tokens in p_{11} are transited to p_{22} and p_{32}, respectively. One the other hand, the probability value $prob_3 = 1$ since the only token in p_{13} is transited to p_{31}. Calculating the r values with Eqn. 3.4, the Petri net is created as shown in Fig. 3.2:

In Fig. 3.2, the places that have not evolved in the transition firing are not shown in order to simplify the structure. We can see that p_{22} and p_{32} each obtained one token from the two tokens in place p_{11}. Hence, the probabilities of t_1 and t_2 are both 0.5. Similarly, since the only token in p_{13} has been transited to p_{31}, the probability of t_3 is 1. The reward

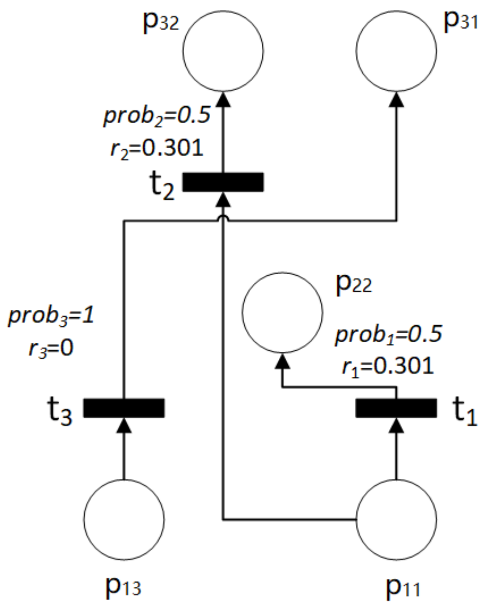

Fig. 3.2. The Probabilistic Petri Net Created from Fig. 3.1.

values are also listed as $r_1 = r_2 = 0.301$ and $r_3 = 0$. Since $prob \in [0,\ 1]$, $-log_{10}^{prob}$ is always positive and is monotonous decreasing related to *prob*. Hence, it is equivalent to find the smallest *prob* value or to obtain the largest *r* value. The corresponding incident matrix *In* of the Petri net in Fig. 3.2 is shown in Equation 3.5:

$$In = \begin{array}{c} p_{11} \\ \cdots \\ p_{13} \\ \cdots \\ p_{22} \\ \cdots \\ p_{31} \\ p_{32} \\ \cdots \end{array} \begin{bmatrix} -1 & -1 & 0 \\ \cdots & \cdots & \cdots \\ 0 & 0 & -1 \\ \cdots & \cdots & \cdots \\ 1 & 0 & 0 \\ \cdots & \cdots & \cdots \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ \cdots & \cdots & \cdots \end{bmatrix} \tag{3.5}$$

Note that the constructed Petri net in Fig. 3.2 has transitions with a single input and a single output, each column of the incident matrix In will at most has two non-zero entries, which are ± 1. The columns can also contain only zeros when there is the existence of the self-loop.

3.1.3 Driving Data Fitting

After obtaining the Petri net with reward values, it is feasible to fit the driving data in the database to a certain model. Note that since we consider only the highway scenarios, there is one constraint for the token moving: The tokens on all lanes have to move synchronously. Since the Petri net model is used to simulate the highway traffic and the vehicles cannot stop on the highway, the tokens representing vehicles are regulated to move synchronously. In order to achieve the constraint in the Petri net, the tokens will be transited in sequence, which means only when all tokens are transited once, the newly generated marking is considered as a reachable marking. We use Fig. 3.1 to illustrate it. In the figure, we can see that two lanes have vehicles in them. Since the vehicles should move synchronously, in one phase, we need to move the vehicle in place p_{11} to place p_{22} or p_{32} and move p_{13} to place p_{31}, according to the possible transition relationship in the figure.

Known the idea above, we can represent the constraint with the state equation of Petri net, which has the form of $Mk_j = Mk_i + In \cdot vf$, where $i, j = 0, 1, 2, ...$ Expanding the state equation into matrix form, we have Equation 3.6 below:

$$\begin{bmatrix} Mk_{j1} \\ Mk_{j2} \\ \dots \\ Mk_{jn} \end{bmatrix} = \begin{bmatrix} Mk_{i1} \\ Mk_{i2} \\ \dots \\ Mk_{in} \end{bmatrix} + \begin{bmatrix} In_{11} & In_{12} & \dots & In_{1m} \\ In_{21} & In_{22} & \dots & In_{2m} \\ \dots & \dots & \dots & \dots \\ In_{n1} & In_{n2} & \dots & In_{nm} \end{bmatrix} \begin{bmatrix} vf_1 \\ vf_2 \\ \dots \\ vf_m \end{bmatrix} \tag{3.6}$$

Applying the constraint to the Petri net in Example 9, the firing vector vf should always have two non-zero entries to represent synchronously moving of the traffic on the two lanes, which means:

$$\sum_{index=1}^{m} vf_{index} = 2 \tag{3.7}$$

The non-zero vf_{index} pairs can be determined from Mk_i. Example 10 has shown the process.

Example 10: For the Petri net in Example 9, consider the starting marking Mk_0 in Equation 3.8 below.

$$Mk_0 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ ... \\ 0 \end{bmatrix} \tag{3.8}$$

Looking up the constructed incident matrix in Equation 3.5, we can find the negative entries in rows corresponding to p_{11} and p_{13}. For p_{11}, negative entries are in column 1 and column 2. For p_{13}, negative entry is in column 3. Hence, We could fire is t_1, t_2, and t_3. The candidates of the firing vector and reward value are as shown in Eqn. 3.9 to Eqn. 3.12.

$$vf_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \tag{3.9}$$

$$vf_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \tag{3.10}$$

$$r_1 = -log_{10}^{0.5 \cdot 1} = 0.301, \tag{3.11}$$

$$r_2 = -log_{10}^{0.5 \cdot 1} = 0.301. \tag{3.12}$$

Thus, substituting the values of the firing vectors, we can also obtain the set of the reachable markings in the next firing stage, denoted as Mk_1, by solving the state equation. In the case of this example, the set includes:

$$Mk_1 = \begin{array}{c} p_{11} \\ p_{12} \\ \dots \\ p_{22} \\ \dots \\ p_{31} \\ \dots \end{array} \begin{bmatrix} 0 \\ 0 \\ \dots \\ 1 \\ \dots \\ 1 \\ \dots \end{bmatrix} \; or\, Mk_1 = \begin{array}{c} p_{11} \\ p_{12} \\ \dots \\ p_{31} \\ p_{32} \\ \dots \end{array} \begin{bmatrix} 0 \\ 0 \\ \dots \\ 1 \\ 1 \\ \dots \end{bmatrix} \tag{3.13}$$

These two Mk_1 markings could be used as the starting markings for the further calculation. By doing the iterative calculations, all the reachable markings that represent the traffic environment could be obtained. They are stored in different layers based on the times of firing from the initial marking Mk_0.

3.1.4 Data Fitting Results

In this section, we will use the method proposed above to discretize the highway segment and model the driving behavior in the dataset. In order to simplify the calculation, We are using data from the first three lanes of the dataset. We apply the data filtering and highway discretization introduced above so that we can obtain the features of the filtered database and discretized highway as shown in Table 3.1:

Note that each filtered data segment is treated as a separate driving circuit in our study. The number of blocks will be equal to the number of places in the Petri net we are constructing.

Furthermore, by fitting the data points into the discretized blocks, we can obtain the values of the reward of each transition. There are only 314 places and 818 transitions in this Petri net since there are places without any connection which are ignored. Because of

Table 3.1.
Features of the Filtered Database and Discretized Highway.

Features of the dataset	Value
#Filtered data segments	1660
#Data points	24600
Time gap (s)	1

Features of the highway	Value
Total length (m)	1261.3
Length of each block (m)	10
#Lanes	3
#Blocks	127*3

the massive size of the Petri net, Fig. 3.3 shows only the first three layers of the generated Petri net. The reward values of transitions have also been provided in Table 3.2 in a matrix corresponding to the incident matrix of the Petri net.

Once the Petri net has been generated, we can obtain its reachable markings. The scenario that we are using is as illustrated in Example 9 and Example 10 in the previous sections, in which the vehicles all follow the driving behavior in the filtered dataset. The total number of the reachable markings is 1984, with 49 layers. The number of markings in each layer is listed in Table 3.3.

Note that every marking could be generated from different paths with different probabilities along with different reward values. Meanwhile, the modeling of the highway traffic with the Probabilistic Petri net could be applied to many realistic problems, such as the path planning for the ego vehicle that has the least collision risk. Moreover, the probability and reward values of the arcs could be converted to the costs or weights values. Therefore, the Probabilistic Petri net could be converted into a traditional Petri net with extra features. In the next section, we would like to introduce another modeling scheme that uses this kind of traditional Petri net.

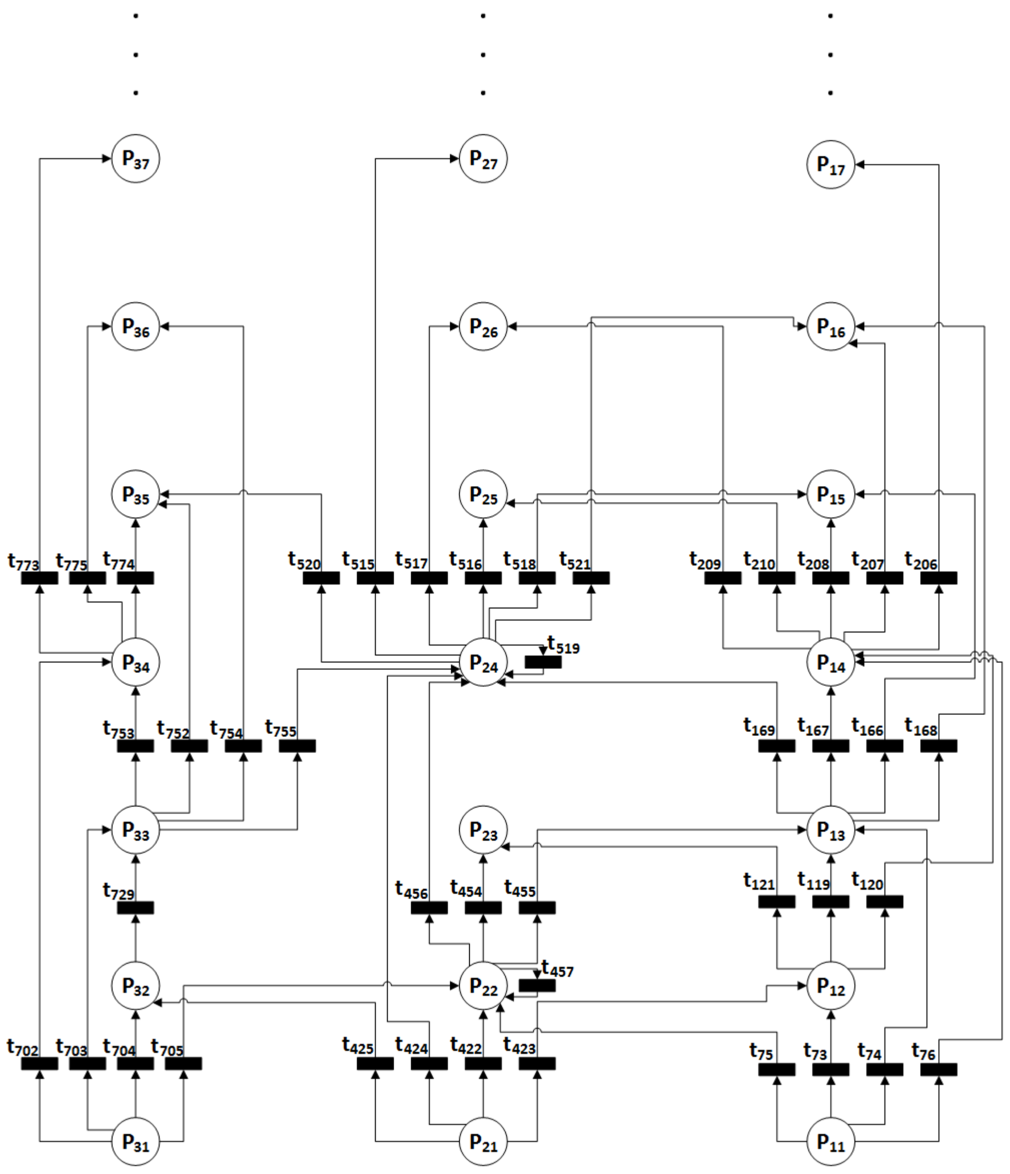

Fig. 3.3. Part of the Petri Net Generated from the Discretized Highway.

Table 3.2.
Reward Values of the Transitions in Fig. 3.3.

Transition	Reward	Transition	Reward
t_{73}	0.1146	t_{455}	5.7944
t_{74}	4.0540	t_{456}	8.6018
t_{75}	6.2416	t_{457}	9.6018
t_{76}	8.4115	t_{516}	0.5595
t_{119}	0.0439	t_{517}	1.6938
t_{120}	7.2288	t_{518}	9.3376
t_{121}	5.4215	t_{519}	9.3376
t_{166}	2.2305	t_{520}	9.3376
t_{167}	0.3831	t_{521}	8.3376
t_{168}	7.2192	t_{702}	4.1293
t_{169}	6.2192	t_{703}	3.5443
t_{206}	7.7649	t_{704}	0.2713
t_{207}	1.0784	t_{705}	5.1293
t_{208}	0.9705	t_{729}	0
t_{209}	6.7649	t_{752}	3.7004
t_{210}	8.7649	t_{753}	0.2410
t_{422}	0.0240	t_{754}	4.7004
t_{423}	6.5580	t_{755}	4.7004
t_{424}	8.1430	t_{773}	4.3923
t_{425}	8.7279	t_{774}	0.5850
t_{454}	0.0319	t_{775}	1.8074

Table 3.3.
The Number of Reachable Markings in each Layer.

Layer	Number	Layer	Number
1	1	26	44
2	16	27	48
3	69	28	44
4	76	29	44
5	70	30	44
6	66	31	32
7	68	32	24
8	72	33	36
9	66	34	28
10	63	35	28
11	66	36	28
12	62	37	24
13	62	38	20
14	63	39	12
15	63	40	16
16	54	41	16
17	55	42	8
18	56	43	12
19	44	44	12
20	60	45	12
21	68	46	12
22	60	47	4
23	60	48	4
24	48	49	4
25	40		

3.2 Highway Traffic Network Modeling with Traditional Petri Net

In this section, we discuss the method of modeling the highway traffic network with the traditional Petri net. Inspired by the study in [97], traffic congestion has a great influence on driving safety and travel efficiency. We would like to further study the topic of optimizing route planning in order to improve safety and efficiency issues. Before that, the traffic network has been analyzed and modeled with the Petri net. In order to simplify the problem, there are several prerequisites: 1) Only highway/freeway and major local roads are considered. 2) Some local roads are considered as one-directional. This section is organized as follows. First, we discuss the method of modeling highway traffic with the Petri net; Then we describe the algorithm for minimum initial marking estimation in labeled Petri nets with unobservable transitions; At last, we provide the estimation results after applying the algorithm to the Petri net generated from the traffic network.

3.2.1 Problem Formulation

The scenario design is as follows: we would like to simulate the daily driving that starts from home and commute to work. During the way, several other destinations of interest ($DOIs$) are designed to be visited. Obviously, the driver has multiple choice for each DOI. For example, the driver would like to visit a McDonald's for lunch, while he will have quite a lot of branches to choose from. Thus, these choices of each DOI will be marked. These $DOIs$ could be used to form the place set Pl in the Petri net we would like to form. Once we have the place set Pl, transition set Tr could be used to represent the traveling between $DOIs$. Hence, the locations of transitions should follow the layout of the target highway traffic network.

Next, we would like to determine the weights of the arcs that connect the places and transitions. The arc weights depend on the traveling time from one DOI to another, which is constrained by the speed limit and also influenced by the distance and the traffic condition. There are many open-source databases online that provide traffic information that provides current real-time traffic speed and speed limits.

Since both the speed limit and real-time traffic speed are available, we could determine the time needed to travel between two *DIOs*. To use the Google map API to check the speed limit, we would like to further discretize the segment between two *DOIs*. Every 0.5 miles, we would obtain the speed limit and calculate the required travel time. Since the interval between data points is very small, our calculation will use the average of the starting and the ending speed limit value of a single interval, in case of that they are different.

Taking into consideration of the traffic congestion, we will make use of the real traffic data from INDoT database. Since the starting point is given in the database and the research in [97] has provided a way to calculate the queue length, it is possible to calculate the required time to travel within the traffic congestion. Hence we could conclude the way to calculate the value of delta arc (w_{diff}) weight in the following equation 3.14:

$$w_{diff} = \Sigma \frac{l_{interval}}{v_{sl_i}} + \Sigma \frac{l_{interval}}{v_{sc_j}},\ i = 1,2...N,\ j = 1,2...,M. \tag{3.14}$$

where $l_{interval}$ is the length of each interval, which equals to 0.5 miles. v_{sl_i} represents the average speed limit in the $i-th$ un-congested interval. v_{sc_j} is the real-time traffic speed in the $j-th$ congested interval. Note that there are N un-congested intervals and M congested intervals, and $N+M$ should be the number of total intervals between the two *DOIs*.

From the equation in Eqn. 3.14, we could see that the difference of the arc weights could be affected significantly by the number of congested intervals as well as the traffic speed in each congested interval. Hence, traffic congestion will be somehow reflected as a large arc weight difference in this generated Petri net. When applying optimization algorithms, those large arc weights should be considered to be avoided.

The arc weights have some features, which represent the consumption of time: 1) The weight of the input arc of a transition t_i will also be strictly larger than the weight of its output arc. Hence, w_{diff} will be always strictly positive. 2) The arc weights on both input and output arcs should be the smallest numbers that could generate w_{diff}, so that we have the following values in 3.15:

$$In_{t_i}^{-} = 1,\ In_{t_i}^{+} = 1 + w_{diff},\ i = 1,2,...m \tag{3.15}$$

3.2.2 Illustrative Example

We would like to take the metropolitan traffic network of Indianapolis as an example to show the modeling method. Consider the map in the following Fig. 3.4:

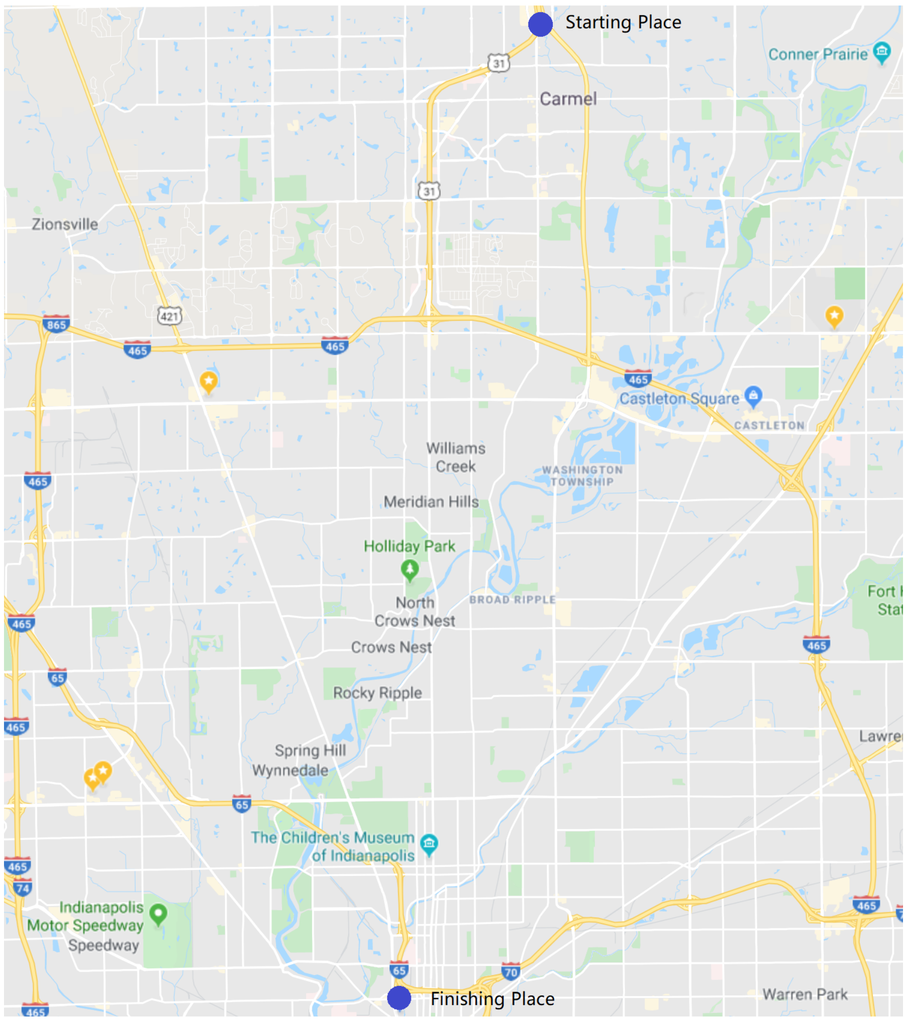

Fig. 3.4. The Metropolitan Highway Traffic Network around Indianapolis.

From the study in [97], it is possible to obtain real-time traffic information from the Indiana Department of Transportation (INDoT) database [98]. INDoT has provided us the real-time probe vehicle queue data that is open to the public. The data, which will be updated every minute, could be accessed on a server with the *.json* format. We could develop a program to fetch the data. Each data segment includes the road name, road direction, mileage of the data point, and current real-time traffic speed. Speed limit information could be obtained from the Google map API [99]. Some local roads that don't have real-time speed information provided in the database. In that case, the traveling speed could be estimated from the historical data in Google map API.

As shown in the figure, the starting place for the commuting route is set at the north-most part of the map at the city Carmel, since it is a common residential area around Indianapolis. The finishing place is at the IUPUI campus, which is at the south-most part of this map. The traffic network consists of highway/freeway (Interstate-465, Interstate-65, Interstate-70, and US-31), and major local roads (Meridian St., Michigan Rd., Binford Blvd, etc.). During the path of driving from the starting place to the finishing place, several *DOIs* are required to be visited. Simulating the real-daily life, we would like to visit a McDonald's, a Walmart supermarket, an Asian market, and a gas station. We could search the map for these *DOIs*, and save those in the traffic network we formed. Thus, connecting these *DOIs*, we could mark all of them on the map and obtain the following Fig. 3.5.

The *DOIs* we have saved will generate the place set Pl in a Petri net. The transition set Tr represents the traveling within *DOIs* along the highway traffic network. Since the arc weights depend on the traveling time from one *DOI* to another, which is influenced by the distance, we need to obtain the distance between the *DOIs*. The distances could be measured by using the measuring tools provided in the online map.

As we have talked about previously, we also consider the traffic condition and speed limit to settle the arc weights. Equation 3.14 could be used to determine all the arc weights we need. We use the following Example 11 to show the detail of arc weights calculation:

Example 11: Consider the highway traffic segment that is between two *DOIs*. The direction is westbound and the total distance is 4.72 miles. Based on the INDoT database, the first 0.5 miles on the road are shown to have slow traffic (marked as red).

Since we have previously defined that the unit distance of each interval as $l_{interval} = 0.5\ mile$, we could divide the segment into 10 intervals (rounded up if not enough), with nine un-congested intervals and one congested interval ($N = 9,\ M = 1$). For the first 0.5 miles with slow traffic, the real-time traffic speed is shown as 47 mph, which is $v_{sc_1} =$

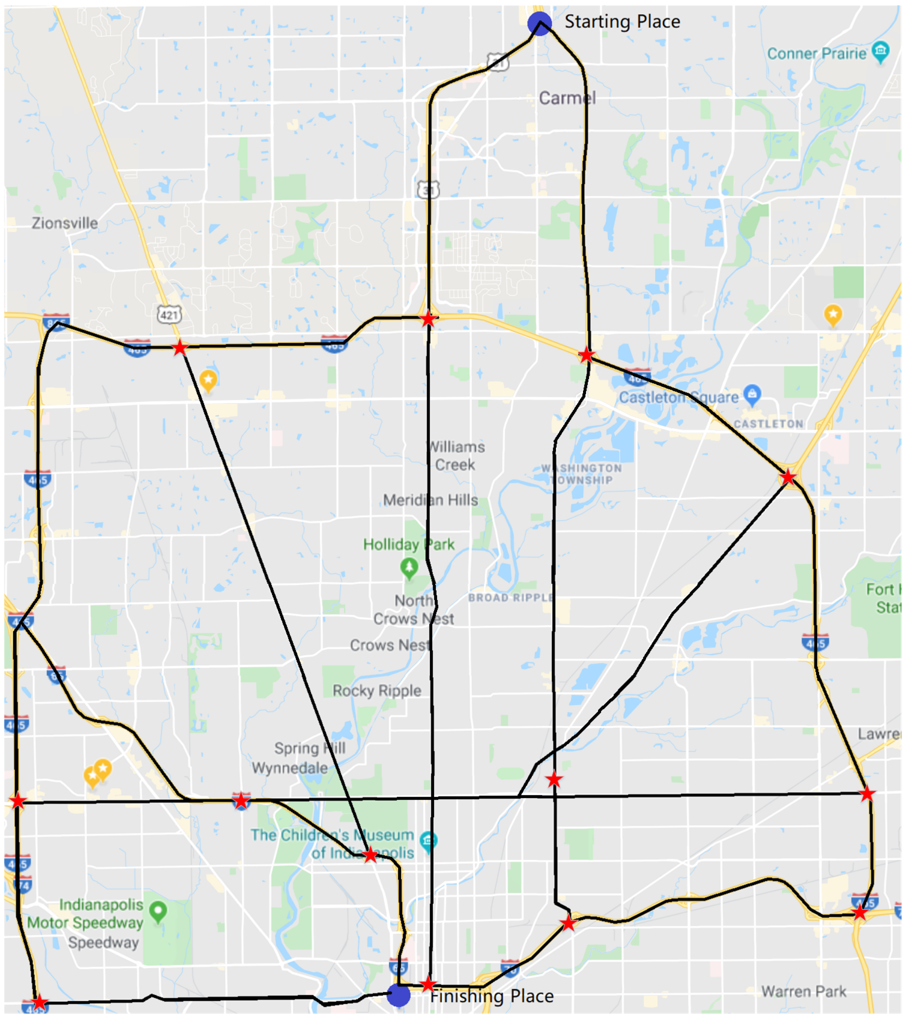

Fig. 3.5. The Highway Traffic Network with *DOI*s Marked.

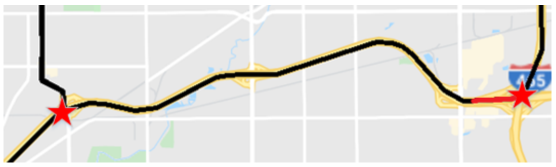

Fig. 3.6. Traffic Segment Between Two *DOI*s used in Example 11.

0.7833 $mile/minute$. For the other un-congested miles, the speed limit is constantly 55 mph, which is $v_{sl_1} = ... = v_{sl_9} = 0.9167\ mile/minute$. Hence, based on Equation (3.14), we could have:

$$w_{diff} = \Sigma\frac{l_{interval}}{v_{sl_1}} + ... + \Sigma\frac{l_{interval}}{v_{sl_1}} + \Sigma\frac{l_{interval}}{v_{sc_1}} \approx 6. \tag{3.16}$$

Therefore, from Equation 3.15, the weights of the input and output arcs of the target transition could be determined as:

$$In_t^- = 1,\ In_t^+ = 7. \tag{3.17}$$

The highway segment in Fig. 3.6 could be converted into the Petri net structure in the following Fig. 3.7:

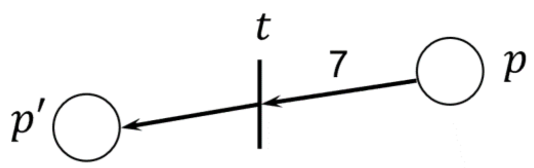

Fig. 3.7. Generated Petri Net from the Highway Segment in Fig. 3.6.

Therefore, for the complete traffic network in Fig. 3.5, we could convert it into a Petri net structure with the obtained place set, transition set, and arc weights. The converted Petri net is shown as follows in Fig. 3.8.

Consider the Petri net in Fig. 3.8, which has 13 places and 37 transitions, the starting point, and finishing point are marked as transition t_1 and t_{37}. Since t_1 is the only output transition of the place p_1, when the token transition happens starting from p_1, we can observe t_1 firing. We have a similar case for t_{37} since it is the only input transition for the destination p_{13}. Every other place of the Petri net represents a destination of interest. Most of them appear at a traffic intersection, which is reasonable since *DOIs* like markets or restaurants are normally built where traffic and population gather. Note that transitions t_{14}, t_{15}, and t_{34} are unobservable transitions, which could provide an alternative firing route. Moreover, we could consider the traffic condition in rush hour and the generated Petri net

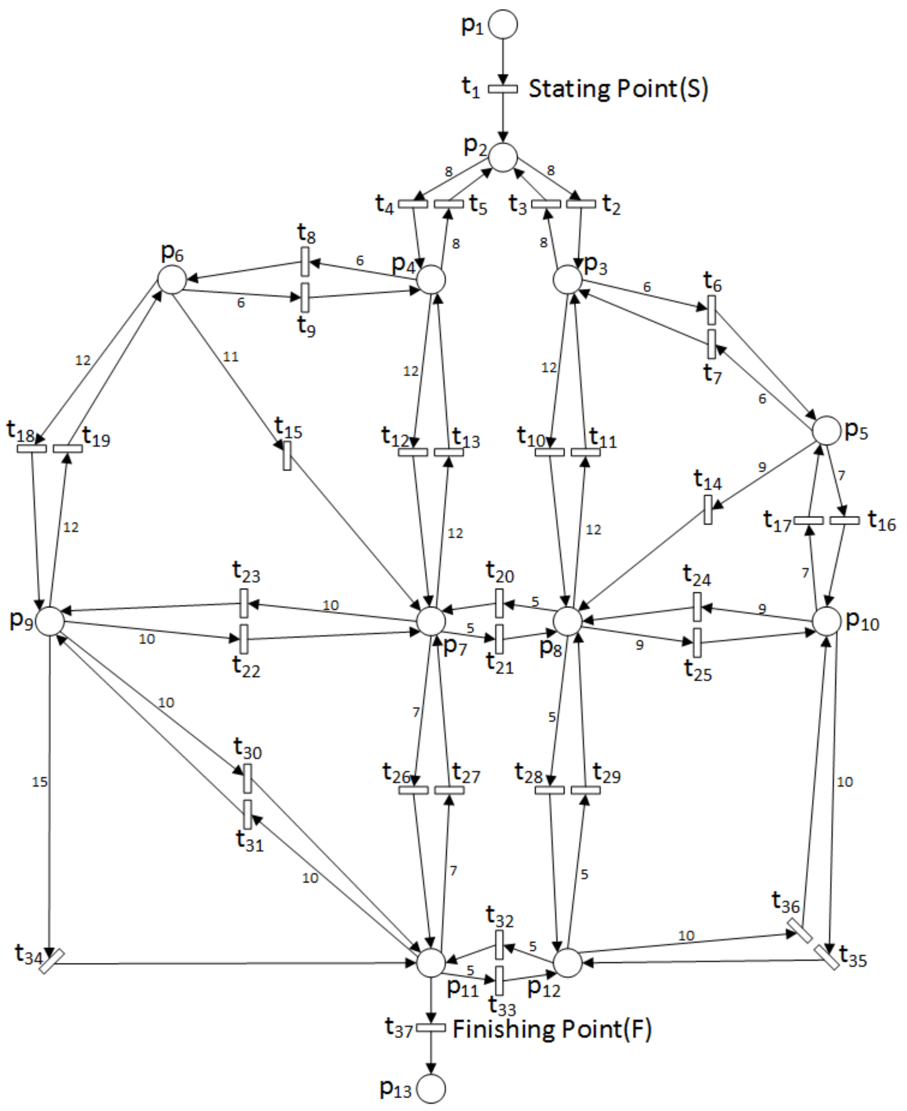

Fig. 3.8. The Petri Net Structure Converted from the Highway Traffic Network without Congestion.

structure could be slightly different from the previous one, where the differences are mainly reflected in the values of arc weights. The more congestion or slow traffic, the weights of the transition input arcs will be larger. We use the traffic data collected at 8 am as an instance and could have the following Petri net structure shown in Fig. 3.9. With the Petri nets generated from the traffic network around Indianapolis, we could apply the algorithm we present in the following chapter for route optimization.

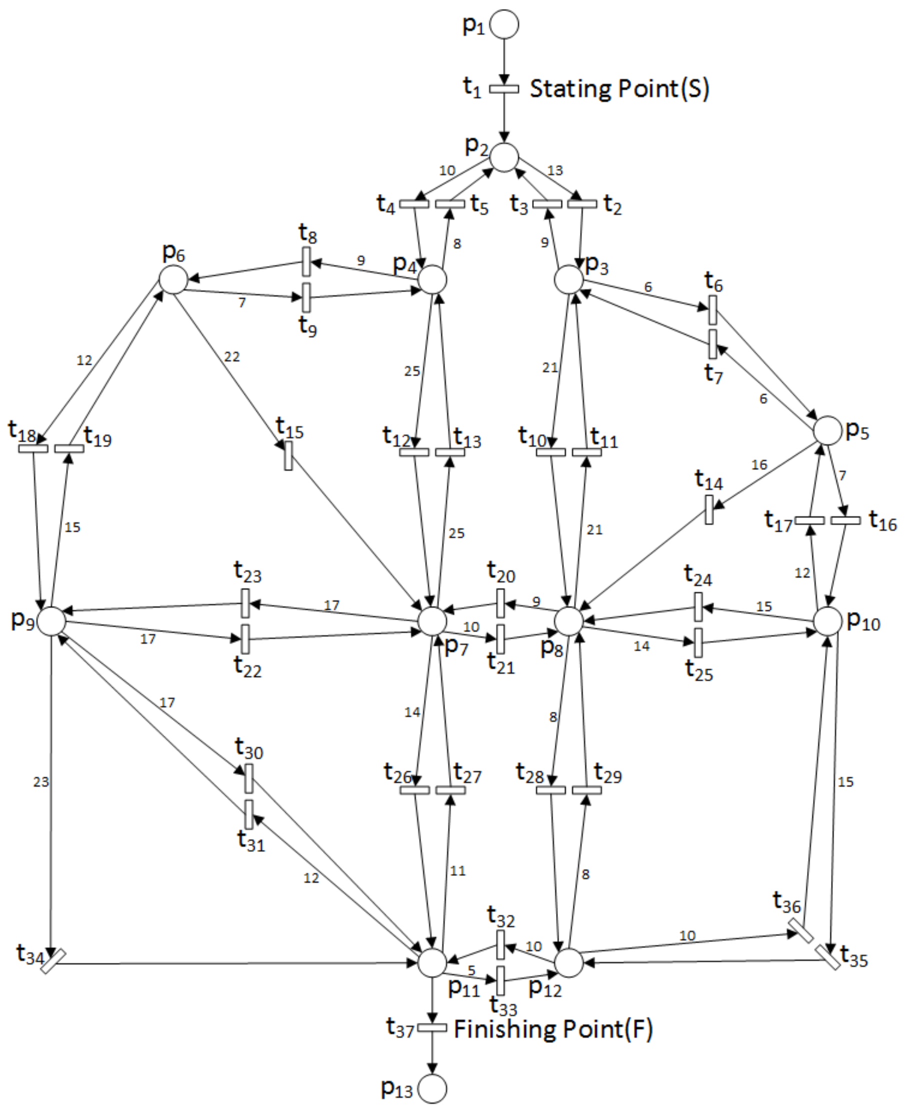

Fig. 3.9. The Petri Net Structure Converted from the Highway Traffic Network with Congestion.

3.3 Summary

In this chapter, firstly, we proposed a method to discretize the highway segment and model driving behaviors with probabilistic Petri nets. This method discretized the continuous highway into discrete blocks. By fitting real-time traffic data, a probabilistic Petri net structure was developed, with places representing blocks on the highway, transitions representing driver's intention, and tokens representing the vehicles. Additional reward values are marked to each transition based on the probability of each transition firing. After the Petri net model is developed, the reachable markings could be obtained. Note that the reachable markings in our case are different from the well-known concept. Because our Petri net is to model the highway traffic, only the reachable markings with all tokens moved synchronously are considered.

Secondly, a scheme that is used to discretize the highway traffic network has been proposed based on the traditional Petri net. The highway network around the Indianapolis metropolitan has been used as an example. The highway network consists of highway/freeway and major local roads. The scenario that we studied is a daily driving case, that starts at home and ends at school. During the way, several other $destinations\ of\ interest$ (DOI) are required to be visited. Based on this scenario, we discretized the continuous traffic network and modeled it with the Petri net structure. We represented the *DOIs* with the place set *Pl* and represented the traffic network connection between *DOIs* with transition set Tr. Arc weights are determined from the distances between *DOIs* and traffic loads. When traveling between the places, time will be consumed. The discretized time units (set to be one minute) are represented by tokens in the net.

Note that, although the first scheme is as powerful as the second scheme and has many possible realistic applications, we focus on the second scheme in our research in this dissertation because of the following reasons: 1) As mentioned previously, the Probabilistic Petri net could be converted into a traditional Petri net by converting the probability or reward values into cost or weight values. 2) The traditional Petri net structure is sufficient for the algorithms developed in this dissertation.

4. MINIMUM INITIAL MARKING ESTIMATION IN LABELED PETRI NETS WITH UNOBSERVABLE TRANSITIONS

In this chapter, we consider the problem of minimum initial marking estimation in labeled Petri nets. The problem is based on the externally observed label sequence and with the purpose of seeking the initial marking that has the minimum token sum. In order to do that, we developed an algorithm based on the conclusion of the study in [84]. The main difference between the study in [84] and ours is the existence of the unobservable transitions. After including the unobservable transitions into the Petri net, more potential firings are necessary to be considered.

This chapter is organized as follows: firstly, some of the important conclusions in [84] have been reviewed at the beginning. The definitions of the initial markings have been given. Meanwhile, the way to find the minimum initial marking in the Petri net that is unobservable transition-free has been reviewed. Secondly, we extend the method into the Petri net with the existence of the unobservable transitions. The differences have been discussed. After a simple example, we then describe the proposed algorithm in detail, along with the complexity analysis. In order to increase the complexity cost, we have also used the other two heuristic methods. After that, we apply the algorithm to an illustrative simulation. An application in a scenario related to traffic planning has also been provided.

4.1 Problem Formulation

4.1.1 Petri Nets with Observable Transitions

In this section, we first consider the labeled Petri net that has observable transitions only. Suppose we are given a Petri net structure with labeling function *Labels*: Ne_{PL}=(Pl, Tr, Ar, We, $Labels$, Δ). An observed label sequence $\theta = l_1 l_2 \dots l_k$, where $l_j \in \Delta$ and $j = 1, 2, \dots, k$, has been generated by underlying transition activities. The study in [84] has proposed a method to find the minimum initial marking estimate set with computational complexity that is polynomial in terms of the length of θ. There are some necessary definitions proposed in [84] that could be used by us in the later sections, when dealing with the nets with the existence of the unobservable transitions. These definitions are shown as follows:

Definition 6: [84] Given a label observation sequence θ, the *initial marking estimates* set with respect to θ is defined by $Z(\theta) = \{Mk \in (Z_0^+)^n \mid \exists \delta \in Tr^* : Mk[\theta\rangle$ and $Labels(\delta) = \theta\}$.

Definition 7: [84] Given a label observation sequence θ, the *minimal initial marking estimates* set with respect to θ is defined by $Z_{minimal}(\theta) = \{Mk \in Z(\theta) \mid \nexists Mk' \in Z(\theta) : Mk' \leq Mk$ and $Mk' \neq Mk\}$.

Definition 8: [84] Given a label observation sequence θ, the *minimum initial marking estimates* set with respect to θ is defined by $Z_{minimum}(\theta) = \{Mk \in Z_{minimal}(\theta) \mid |Mk| \leq |Mk'|$ for all $Mk' \in Z_{minimal}(\theta)\}$.

[84] has also provided the equation to calculate the minimal initial markings, which are shown in Equation 4.1 below:

$$Mk_0^{j+1} = \max\{Mk_0^j + In \cdot y_{j-1},\ In^-(:,t_{i_j})\} - In \cdot y_{j-1}, j = 1,2,\ldots,k. \tag{4.1}$$

In the equation, y_{j-1} represents the $(j-1)^{th}$ firing vector at the $j-1^{th}$ time epoch of the transition firing sequence $t_{i_1}t_{i_2}\ldots t_{i_{j-1}}$. Mk_0^1 denotes an n-dimensional vector with all zero entries, i.e., $\vec{\mathbf{0}}_n$. Similarly, y_0 denotes an m-dimensional vector with all zero entries, i.e., $\vec{\mathbf{0}}_m$. Note that markings Mk_0^j and Mk_0^{j+1} represent the initial marking estimate before and after the firing of transition t_{i_j}.

To show the calculation procedure of the Equation 4.1 more intuitively, the *trellis diagram* has been introduced in [84], which enumerates all possible firing path. The structure of the trellis diagram is as shown in Fig. 4.1.

In the diagram, each node, which is denoted as a black dot, represents a (y_{ij}, Mk_{ij}^0) pair, where Mk_{ij}^0 is a minimal initial marking estimate, which is consistent with the observed label sequence θ, i.e., $Mk_{ij}^0 \subseteq Z(\theta)$. y_{ij} is the firing vector belonging to one or more

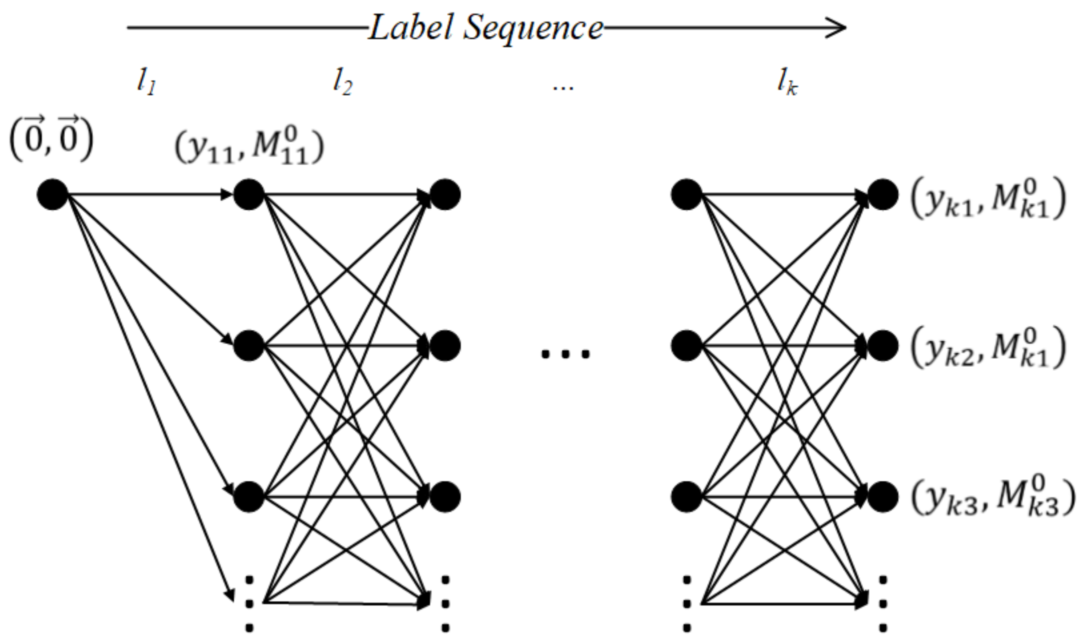

Fig. 4.1. Trellis Diagram that Capturing the Firing Procedure while Showing Firing Vectors and Minimal Markings.

transition firing sequences that are fired from the corresponding Mk_{ij}^0 and are consistent with the observed label sequence θ at i^{th} time epoch.

After scanning the entire observed label sequence, the minimum initial marking estimate set can be found from the minimal initial marking estimate set, which are the ones with the least token sum, that is:

$$Mk_{Minimum}^0 = \arg\min_{Mk_{ki}^0} \sum_{i=1}^{n} Mk_{ki}^0(p_i). \tag{4.2}$$

4.1.2 Petri Nets with the Existence of Unobservable Transitions

Before we could get any deeper, it is necessary to make it clear that if the Petri net has the unobservable subnet, the number of firing sequences can grow exponentially in terms of the length of the observed label sequence. In the worst case, there could be infinite possibilities of the transition firings for one observed label. In that case, we assume that all the unobservable transitions are forward-backward-conflict-free ($|^{\rightarrow}p_i^{\rightarrow}| = 1$) and forward-backward-concurrent-free ($|^{\rightarrow}t_i^{\rightarrow}| = 1$), or *contact-free* by combing the two concepts. Before establishing more procedures, We need to define a notion called *implication* that helps to make ideas of the algorithm clear.

Definition 9: [84] Consider a labeled Petri net with unobservable transitions. Given an observable transition $t \in Tr_o$, an *implication* δ is a firing sequence consistent with unobservable transitions $t_i \in Tr_u, i = 1, 2...|Tr_u|$, that starting from an initial marking Mk_0, and can lead the markings to enable t, i.e., $E(Mk_0, t) = \{\delta \mid Mk_0[\delta\rangle Mk',\ Mk' \geq In^-(t)\}$.

Definition 10: Define $E_{minimal}(Mk_0, t)$, the *minimal implication* towards $t \in Tr_o$, to be the *implication* δ starting from a marking $Mk \in Z_{minimal}(\theta)$ that satisfy the observation label sequence.

Definition 11: Define $E_{minimum}(Mk_0)$, the *minimum implication* towards $t \in Tr_o$, to be the *implication* δ starting from a marking $Mk \in Z_{minimum}(\theta)$ that satisfy the observation label sequence θ.

Remark 1: The problem of *MIM-UT* in Petri net that has unobservable transitions can be treated as problems of finding *minimum implications* $E_{minimum}(Mk_0)$ of the target observable transition $t \in Tr_o$.

To simplify the calculation, in this chapter we would like to make the following assumptions: $A1)$ The unobservable subnet is contact-free. $A2)$ One and only one unobservable transition can be fired before each observable transition firing that is consistent with the current label.

Note that since t_u does not have any corresponding observed label, it can potentially be fired for infinite times before the next observation. Therefore, assumption $A2)$ is a constraint for the firing of each t_u, which makes it possible to enumerate all firing sequences. Based on assumption $A2)$, for each observed label l_j, the consistent firing sequences should be in the form of t or $t_u t$, where $t \in Tr_o, t_u \in Tr_u$, and $Labels(t) = l_j$. Known these, we could apply Equation 4.1 to these sequences after each observed label to obtain the firing vectors y along with the associated marking estimates. We will demonstrate the procedure of determining the minimum initial marking estimates in a Petri net that has unobservable transitions with the following example:

Example 12: Consider the labeled Petri net with the unobservable transitions shown in Fig. 2.5.(a). The goal is to obtain the minimum initial marking estimates based on the observed label sequence $\theta = aa$. Through the conclusion in [84], we could know that the trellis diagram shows the enumeration of all possible firing path. The trellis diagram of this example is shown in Fig. 4.2 as follows:

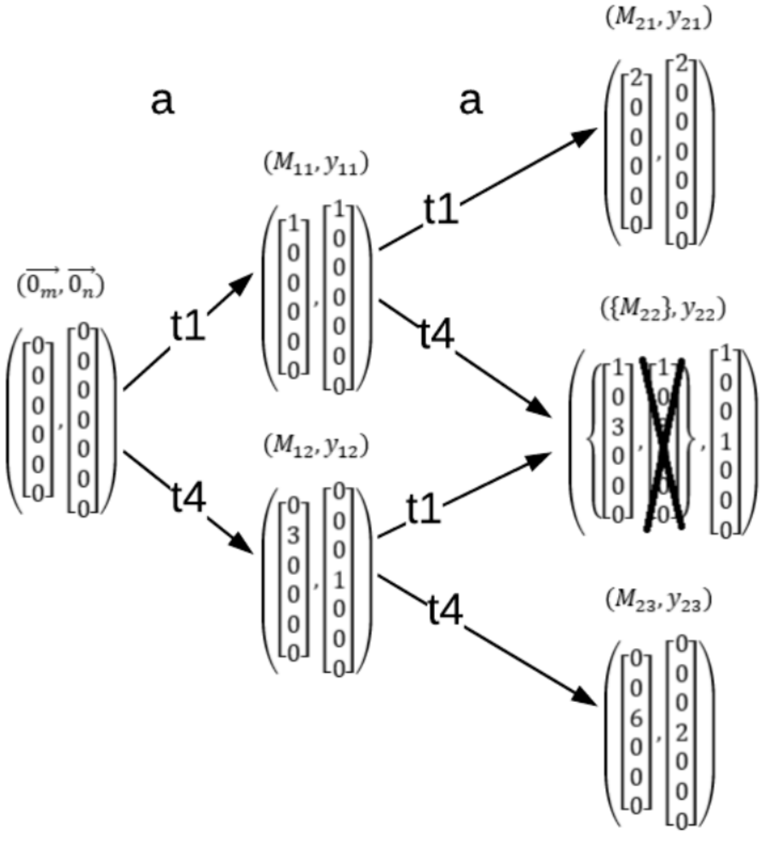

Fig. 4.2. Trellis Diagram of the Observable Subnet in Example 12 [84].

In this example, only observable transitions are considered. It is clear that the minimum initial marking estimate that is consistent with the observed label sequence *aa* is $[2,\ 0,\ 0,\ 0,\ 0,\ 0]^T$ via transition sequence t_1t_1 and firing vector $y = [2,\ 0,\ 0,\ 0,\ 0,\ 0,\ 0]^T$. However, when unobservable transitions are included, the number of transition firing sequences will be significantly larger. For example, for each observed label a, the following firing sequences will need to be considered: $\{t_1, t_4, t_5t_1, t_6t_1, t_5t_4, t_6t_4\}$. By applying Equation 4.1 to these firing sequences, the minimal initial marking estimate set could be obtained, where we can obtain the minimum initial marking estimate(s). The obtained minimal initial marking estimate set and the associated firing vectors are shown in Table 4.1. Due to space considerations, Table 4.1 does not include all transition firing sequences. Those marking estimates that are not minimal have been removed.

Table 4.1.
The Estimated Minimal Initial Markings and Associated Firing Vectors in Example 12.

Labels	Sequences	Minimal marking estimates	Firing vectors
	t_1t_1	$[2\ 0\ 0\ 0\ 0\ 0]^T$	$[2\ 0\ 0\ 0\ 0\ 0\ 0]^T$
	t_1t_4	$[1\ 0\ 3\ 0\ 0\ 0]^T$	$[1\ 0\ 0\ 1\ 0\ 0\ 0]^T$
	t_4t_4	$[0\ 0\ 6\ 0\ 0\ 0]^T$	$[0\ 0\ 0\ 2\ 0\ 0\ 0]^T$
	$t_6t_1t_1$	$[2\ 0\ 0\ 0\ 1\ 0]^T$	$[2\ 0\ 0\ 0\ 0\ 1\ 0]^T$
	$t_1t_5t_1$	$[2\ 0\ 0\ 0\ 0\ 0]^T$	$[2\ 0\ 0\ 0\ 1\ 0\ 0]^T$
	$t_1t_6t_1$	$[2\ 0\ 0\ 0\ 1\ 0]^T$	$[2\ 0\ 0\ 0\ 0\ 1\ 0]^T$
	$t_6t_1t_4$	$[1\ 3\ 0\ 0\ 1\ 0]^T$	$[1\ 0\ 0\ 1\ 0\ 1\ 0]^T$
	$t_1t_5t_4$	$[1\ 0\ 0\ 0\ 0\ 0]^T$	$[1\ 0\ 0\ 1\ 1\ 0\ 0]^T$
	$t_5t_4t_4$	$[0\ 3\ 1\ 0\ 0\ 0]^T$	$[0\ 0\ 0\ 2\ 1\ 0\ 0]^T$
aa	$t_6t_4t_4$	$[0\ 6\ 0\ 0\ 1\ 0]^T$	$[0\ 0\ 0\ 2\ 0\ 1\ 0]^T$
	$t_5t_1t_5t_4$	$[1\ 0\ 1\ 0\ 0\ 0]^T$	$[1\ 0\ 0\ 1\ 2\ 0\ 0]^T$
	$t_6t_1t_5t_4$	$[1\ 0\ 0\ 0\ 1\ 0]^T$	$[1\ 0\ 0\ 1\ 1\ 1\ 0]^T$
	$t_6t_1t_6t_4$	$[1\ 3\ 0\ 0\ 2\ 0]^T$	$[1\ 0\ 0\ 1\ 0\ 2\ 0]^T$
	$t_5t_1t_5t_1$	$[2\ 0\ 1\ 0\ 0\ 0]^T$	$[2\ 0\ 0\ 0\ 2\ 0\ 0]^T$
	$t_6t_1t_5t_1$	$[2\ 0\ 0\ 0\ 1\ 0]^T$	$[2\ 0\ 0\ 0\ 1\ 1\ 0]^T$
	$t_6t_1t_6t_1$	$[2\ 0\ 0\ 0\ 2\ 0]^T$	$[2\ 0\ 0\ 0\ 0\ 2\ 0]^T$
	$t_5t_4t_5t_4$	$[0\ 0\ 2\ 0\ 0\ 0]^T$	$[0\ 0\ 0\ 2\ 2\ 0\ 0]^T$
	$t_5t_4t_6t_4$	$[0\ 3\ 1\ 0\ 1\ 0]^T$	$[0\ 0\ 0\ 2\ 1\ 1\ 0]^T$
	$t_6t_4t_6t_4$	$[0\ 6\ 0\ 0\ 2\ 0]^T$	$[0\ 0\ 0\ 2\ 0\ 2\ 0]^T$

It is obvious that the minimum initial marking estimate that is consistent with the observed label sequence aa is $[1,\ 0,\ 0,\ 0,\ 0,\ 0]^T$ via transition firing sequence $t_1t_5t_4$. This minimum initial marking estimate has a smaller token sum compared to the result obtained by the method that considers observable transitions only.

4.2 Descriptions of the Algorithms

4.2.1 Main Algorithm

In this section, we propose a recursive algorithm to seek the minimum initial marking estimate set based on an observed label sequence $\theta = l_1l_2\dots l_k$ that has length k in a labeled Petri net with the unobservable transitions. In Algorithm 1, for each observed label, we consider one unobservable transition, which is followed by the observable transitions that are consistent with the observed label (based on assumption $A2$)). The firing vectors and the corresponding minimal initial marking estimate sets are calculated recursively based on the trellis diagram. An indicator $Flag$ is used to denote the transition under consideration. When $Flag$ is set to be $TRUE$, we are dealing with an observable transition. Otherwise, an unobservable transition is being considered.

4.2.2 Complexity Analysis

Based on the results in [84], the upper bound of the number of firing vectors in observable transition only cases at the j^{th} stage is $O(n_j) = O(j^b)$, where b is a parameter related to the structure of the Petri net. Meanwhile, since we can fire one and only one unobservable transition for each observed label, the total number of different unobservable firing vectors is m_u, where m_u is the number of unobservable transitions. Therefore, at the j^{th} stage, there would be $n_j = (m_u j)^b$ different firing vectors. Meanwhile, at the $(j-1)^{th}$ stage, each stored firing vector can generate at most $m_u \times m_o$ new firing vectors, where m_o denotes the number of observable transitions. Thus, at the j^{th} stage, the number of newly generated firing vectors is upper bounded by $O(m_u \times m_o \times n_{j-1}) = O(m_o m_u (m_u j)^b)$. When doing the

Algorithm 1
Minimum Initial Marking Estimation in a Labeled Petri Net with Contact-free Unobservable Transitions.

Input: A labeled Petri net with contact-free Tr_u and $\theta = l_1 l_2 \ldots l_k$ of length k.

Output: Minimum initial marking estimate(s).

```
1: $Mk_0^1 = \vec{0}_n$, $y_0 = \vec{0}_m$, $Flag == TRUE$.
2: $nodes = (Mk_0^1, y_0, Flag)$
3: for $i = 1$ to $k$ do
4: Search nodes to find the ones with $t_i \in Tr_{l_i}$.
5: for each $node \in nodes(:, i)$ with $t_i \in Tr_{l_i}$ do
6: for each transition $t \in Tr_{l_i} \cup Tr_u$ do
7: Call Algorithm 2.
8: end for
9: end for
10: end for
11: Output the minimum initial marking estimate set
```

comparisons of the firing vectors, which determine their uniqueness, the largest number is represented by n_j while each comparison is upper bounded by $O(m_u + m_o) = O(m)$. If the uniqueness check fails, we need to do the comparison between the newly calculated marking estimate with the existing ones. The maximum number of comparisons is represented by q_j, while the complexity for each comparison is upper bounded by $O(n)$.

The parameter q_j denotes the number of minimal initial marking estimates corresponding to each firing vector at the j^{th} stage. We are trying to find markings that share the same firing vector but not comparable. For observable cases, based on the results from the existing study, the q_j is bounded by $O(j^n)$. For unobservable cases, since we assume that each unobservable transition can fire one and only one time for each observed label, the firing sequence under consideration would have a length of $2j$. In that case, q_j is upper bounded by $O((2j)^n) = O(j^n)$. Based on the analysis above, it is clear that the computational com-

Algorithm 2
Main Loop of Algorithm 1

```
1: Update $Mk_0^{i+1}$ using Equation (4.1) and the values of $Mk_0^i$ and $y_{j-1}$ stored in $node$.
2: Update the firing vector $y_i$ with the values of $y_{i-1}$ and the current transition $t$.
3: **if** There is duplication of the value of $y_i$ **then**
4: Compare $Mk_0^{i+1}$ with $Mk_{minimal}$ that share the same $y_i$.
5: **if** $Mk_{i+1}^0$ is not comparable with $Mk_{minimal}$ **then**
6: Store $node$ in $nodes(:,i)$
7: **else if** $Mk_0^{i+1} \leq Mk_{minimal}$ **then**
8: Replace the original $node$ in $Mk_{minimal}$ with the current $node$ in $nodes(:,i)$.
9: **end if**
10: **end if**
11: **if** $t \in Tr_u$ **then**
12: Set $Flag == FALSE$
13: **else**
14: Set $Flag == TRUE$
15: **end if**
16: **if** $Flag == FALSE$ **then**
17: **for** each $t \in Tr_{l_i}$ **do**
18: Goto 7
19: **end for**
20: **end if**
```

plexity of Algorithm 1 is upper bounded by: $\sum_{j=1}^{k}[O(n_{j-1} \times m_o \times m_u \times (m \times n_j + q_{j-1} \times q_j \times n))]$, which is $\sum_{j=1}^{k}[O(m_o m_u (m_u j)^b \times [m m_u (m_u j)^b + j^n \times n \times j^n])]$. It can be simplified as:

$$O(m_o m m_u^{2b+2} k^{2b+1} + m_o n m_u^{b+1} k^{2n+b+1}) = O(k^{2b+1} + k^{2n+b+1}).$$

We can see that the complexity of Algorithm 1 is polynomial in terms of the length of the observed label sequence k. On the other hand, it also has exponential complexity with respect to some parameters related to the structure of the Petri net, such as the number of the places n and the structural parameter b mentioned above.

4.2.3 Heuristic Methods

In some practical applications, it is more important for the algorithm to work in real-time than to be exactly accurate. Thus, there could be a trade-off between computational cost and accuracy. Hence, when high accuracy is not required, we can further reduce the complexity of the proposed algorithm. In this section, two heuristic methods are proposed, which can improve the time cost of solving the minimum initial marking estimation problem but can only find a subset or an approximation of the real solution.

The first heuristic algorithm is inspired by the $Hill-Climbing$ algorithm [101]. In each iteration, from the legal candidates, only the marking(s) with the smallest token sum, which is(are) called the $current-best$, will be kept for further calculations.

Algorithm 3 Heuristic Method 1 (Hill-Climbing) for the Minimum Initial Marking Estimation Problem

Input: A labeled Petri net with contact-free Tr_u and $\theta = l_1 l_2 \dots l_k$ of length k.
Output: Minimum initial marking estimate(s).

```
1: $Mk_0^1 = \vec{0}_n$, $y_0 = \vec{0}_m$, $Flag == TRUE$.
2: $nodes = (Mk_0^1, y_0, Flag)$
3: for $i = 1$ to $k$ do
4: Search $nodes$ to find all candidates that have the smallest token sum in the current stage, and saved as set $nodes_{min}$.
5: for each $node \in nodes_{min}$ do
6: for each transition $t \in Tr_{l_i} \cup Tr_u$ do
7: Update $Mk_0^{i+1}$ by applying Equation (4.1) from the values of $Mk_0^i$ and $y_{j-1}$ .
```

```
8:  Update the firing vector $y_i$ based on the values of $y_{i-1}$ and the transition $t$.
9:  if Uniqueness check of $y_i$ fails then
10: Compare $Mk_0^{i+1}$ with existing $Mk_{minimal}$ that share the $y_i$.
11: if $Mk_{i+1}^0$ is not comparable with $Mk_{minimal}$ then
12: Store $node$ in $nodes(:,i)$
13: else if $Mk_0^{i+1} \leq Mk_{minimal}$ then
14: Replace the original $node$ in $Mk_{minimal}$ with the newly calculated $node$.
15: end if
16: end if
17: if $t \in Tr_u$ then
18: Set $Flag == FALSE$
19: else
20: Set $Flag == TRUE$
21: end if
22: if $Flag == FALSE$ then
23: for each $t \in Tr_{l_i}$ do
24: Goto 7
25: end for
26: end if
27: end for
28: end for
29: end for
30: Output the minimum initial marking estimate set
```

The second heuristic method is similar to the work introduced in the study in [84], which considers only the observable part of the Petri net. Without the unobservable subnet, the candidates of the consistent transition sequences would be much less, which also makes the structure of the associated trellis diagram much simpler. But it could still be ideal for some practical applications when the unobservable events would not affect much or the size of the unobservable part is relatively small.

Algorithm 4 Heuristic Method 2 (Observable Subnet Only) for the Minimum Initial Marking Estimation Problem

Input: A labeled Petri net with contact-free Tr_u and $\theta = l_1 l_2 \dots l_k$ of length k.

Output: Minimum initial marking estimate(s).

```
1: $Mk_0^1 = \vec{0}_n$, $y_0 = \vec{0}_m$, $Flag == TRUE$.
2: $nodes = (Mk_0^1, y_0, Flag)$.
3: for $i = 1$ to $k$ do
4: Search $nodes$ to find all candidates with $t_i \in Tr_{l_i}$.
5: for each $node \in nodes(:,i)$ with $t_i \in Tr_{l_i}$ do
6: for each transition $t \in Tr_{l_i}$ do
7: Update $Mk_0^{i+1}$ by applying Equation (4.1) from the values of $Mk_0^i$ and $y_{j-1}$.
8: Update firing vector $y_i$ from the values of $y_{i-1}$ and the transition $t$.
9: if Uniqueness check of $y_i$ fails then
10: Compare $Mk_0^{i+1}$ with minimal initial marking estimates $Mk_{minimal}$ that share the same $y_i$.
11: if $Mk_{i+1}^0$ is not comparable with $Mk_{minimal}$ then
12: Store $node$ in $nodes(:,i)$.
13: else if $Mk_0^{i+1} \leq Mk_{minimal}$ then
14: Replace the original $node$ in $Mk_{minimal}$ with the newly calculated $node$.
15: end if
16: end if
17: if $t \in Tr_u$ then
18: Set $Flag == FALSE$
19: else
20: Set $Flag == TRUE$
21: end if
22: if $Flag == FALSE$ then
23: for each $t \in Tr_{l_i}$ do
```

```
24: Goto 7
25: end for
26: end if
27: end for
28: end for
29: end for
30: Output the minimum initial marking estimate set
```

The performance of the main algorithm and the two heuristic algorithms will be compared in the following illustrative example.

4.3 Illustrative Example

In this section, an example of a manufacturing system that is modeled by Petri nets will be used to evaluate the performance of the three proposed algorithms. Consider the labeled Petri net shown in Fig. 4.3, which is modified from an example in [94]. This Petri net has 10 different places $Pl = \{p_1, p_2, \ldots, p_{10}\}$ and 12 different transitions $Tr = \{t_1, t_2, \ldots, t_{12}\}$. The labeling function is shown as: $Labels(t_3) = Labels(t_5) = a$, $Labels(t_6) = b$, $Labels(t_7) = Labels(t_8) = c$, $Labels(t_9) = d$, $Labels(t_1) = e$, $Labels(t_2) = f$, $Labels(t_{11}) = g$, $Labels(t_{12}) = h$ and $Labels(t_4) = Labels(t_{10}) = \varepsilon$. Note that the two unobservable transitions t_4 and t_{10} are contact-free.

The observed label sequence is also provided as: $\theta = efabcdcbgh$ with a length of 10. Algorithm 1 and the two heuristic algorithms have been applied to this example. Performance is evaluated in the following two aspects: 1) The number of minimal initial marking estimates for each observed label; 2) The accuracy of the final output of the minimum initial marking estimate(s).

Table 4.2 lists the number of minimal initial marking estimates obtained by the three algorithms for each observed label. In the headers of the table, *Main*, *HM*1, *HM*2 stand for the main algorithm (Algorithm 1), Heuristic Method 1 and Heuristic Method 2, respectively. It is clear that the main algorithm generates many more candidates than the two

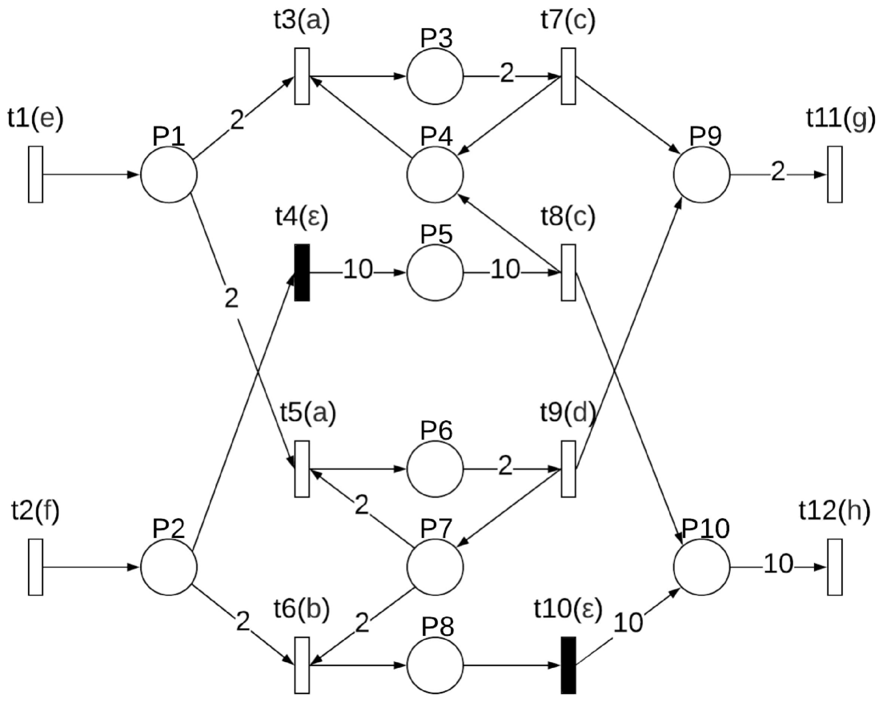

Fig. 4.3. Petri Net Model of the Illustrative Example.

heuristic algorithms. Fig. 4.4 plots the growing trends of minimal initial marking estimates of the three algorithms with respect to the length of the observed label sequence, respectively. In order to show that the algorithmic complexity is polynomial in terms of the length of the observed label sequence, the curve of the function $y = k^4$ has been included in Fig. 4.5 for comparisons. It is obvious that the growing trends of the number of minimal initial markings obtained by the three algorithms are bounded by $O(k^4)$, which shows that they have polynomial computational complexities in terms of the length of the observed label sequence k.

In Table 4.3, the sets of minimum initial marking estimate(s) have been listed as the final output of the three algorithms. As expected, the main algorithm has taken more minimal initial marking estimates into consideration during each step. Hence, it obtained the most completed and accurate minimum initial marking estimate(s) set, where each marking

Table 4.2.
The Numbers of Minimal Initial Marking Estimates Obtained by the Three Algorithms.

Length of the Label Sequence	Main	HM1	HM2
1	5	5	1
2	11	5	1
3	39	8	2
4	62	5	2
5	184	8	4
6	310	5	4
7	468	8	7
8	694	5	7
9	1,140	5	7
10	1,318	5	7

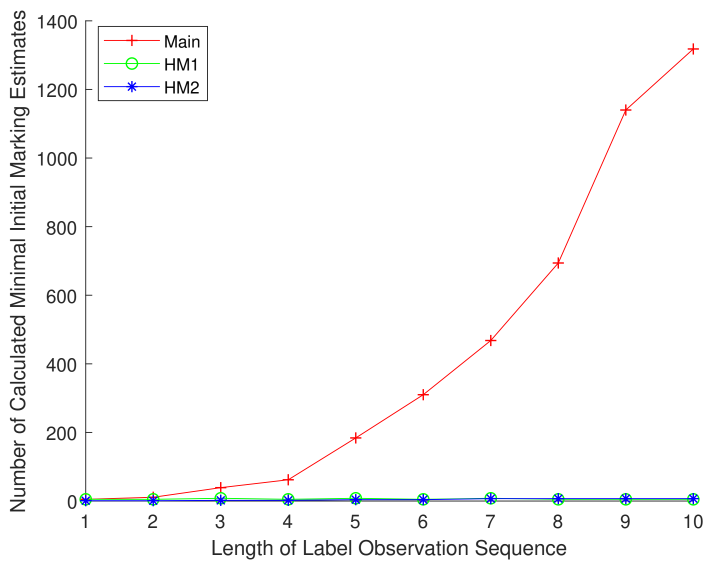

Fig. 4.4. The Numbers of Minimal Initial Marking Estimates Obtained by the Three Algorithms.

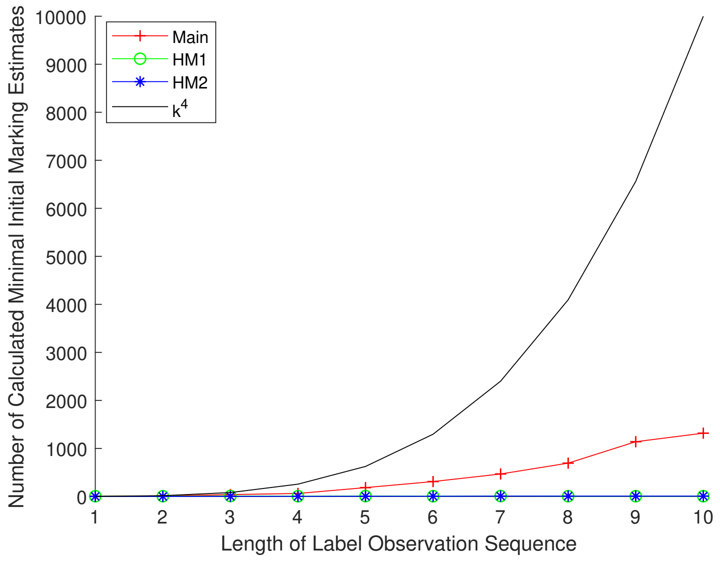

Fig. 4.5. The Number of the Minimal Initial Marking Estimates Obtained by the Three Algorithms, Compared with a Polynomial Function k^4.

Table 4.3.
Final Output of the Minimum Initial Marking Estimates Obtained by the Three Algorithms.

Main	HM1	HM2
$[1\ 4\ 1\ 2\ 0\ 2\ 3\ 0\ 0\ 0]^T$	$[1\ 4\ 1\ 2\ 0\ 2\ 3\ 0\ 0\ 0]^T$	$[1\ 3\ 4\ 0\ 0\ 1\ 5\ 0\ 0\ 10]^T$
$[1\ 4\ 2\ 0\ 0\ 1\ 5\ 0\ 0\ 0]^T$		
$[1\ 5\ 0\ 0\ 0\ 1\ 5\ 0\ 1\ 0]^T$		

estimate has a token sum of 13. Meanwhile, $HM1$ and $HM2$ ignored part of the candidates during the calculation, which accelerated the speed of computation, while the provided solutions are not as good as the main algorithm. For instance, $HM1$ is able to find only a subset of the estimates obtained by the main algorithm (one marking that has a token sum of 13) and $HM2$ can only find an approximation (one marking that has a token sum of 24). Therefore, after the comparison of the three algorithms, we could see that the main algorithm is able to obtain a complete set of minimum initial marking estimates with a higher computational cost. The two heuristic methods are able to find a partial or an approximation of the set of the estimates from the main algorithm with a lower computational cost. For different application purposes, different methods could be applied depending on needs.

4.4 Application of the Minimum Initial Marking Estimation in a Traffic Scenario for Time Cost Minimization

4.4.1 Scenario Description and Simulation Results

In this section, we illustrate the way to apply the algorithm of minimum initial marking estimation to the Petri net in Fig. 3.5. First of all, since we would like to always start at p_1 and end at p_{13}, we assign label s to transition t_1 and f to transition t_{37} and insert s to the beginning of the label sequence and f to the end of the label sequence. The other observable transitions are assigned with labels based on different *DOIs* types, where a, b and c represent food/coffee/restaurants, markets, and the others, respectively. The labeling mapping relationship is shown as follows:

$$Labels(t_1) = s \tag{4.3}$$

$$Labels(t_2,t_3,t_4,t_5,t_6,t_7,t_{12},t_{13},t_{18},t_{19},t_{29}) = a \tag{4.4}$$

$$Labels(t_8,t_9,t_{10},t_{11},t_{16},t_{17},t_{22},t_{23},t_{35},t_{36}) = b \tag{4.5}$$

$$Labels(t_{20},t_{21},t_{24},t_{25},t_{26},t_{27},t_{28},t_{30},t_{31},t_{32},t_{33}) = c \tag{4.6}$$

$$Labels(t_{37}) = f \tag{4.7}$$

Thus, we could have the observation label sequence θ. Different from the fixed sequence in the algorithm mentioned above, the label sequence in the scenarios of this simulation example could be variable, which only needs to satisfy the requirement of the number of all the labels. For instance, we require the label sequence with two a, three b, and three c in the simulation example. Thus, after adding s in front and f at the end, we could have 560 different label sequence combinations and they are similar to the format of the sequence

$\theta = saabcbbccf$, where the sequence length is ten. We first run the updated algorithm and two heuristic algorithms on the Petri net in Fig. 3.8, which is based on the situation without heavy traffic. Then, we run the algorithms on the Petri net in Fig. 3.9, which is based on the traffic situation in rush hours. The performance of these algorithms are compared in several aspects: 1) The number of solutions, and 2) The optimality of the obtained solutions.

For both Petri nets that are generated from un-congested and congested traffic networks, we have 39 different feasible label sequences, as shown in Table 4.4 and Table 4.5:

As we can see in Table 4.4 and Table 4.5, in situation one, there is no traffic considered and the optimal solutions are label sequences $sabcbbcacf$ and $sabccabbcf$ (the corresponding firing sequences are $t_1t_2t_{10}t_{20}t_{23}t_{22}t_{21}t_{28}t_{32}t_{37}$ and $t_1t_2t_{10}t_{20}t_{21}t_{28}t_{36}t_{35}t_{32}t_{37}$), which have 53 tokens. That means the minimum required time should be 53 minutes if we want to visit all these *DOI*s during the path from the starting point to the finishing point. On the other hand, in situation two, the traffic data was captured at 8 am on non-holiday weekday, traffic congestion was happening in the traffic network. The optimal solutions are label sequence $sabaccbbcf$ (the corresponding firing sequence is $t_1t_4t_8t_{18}t_{30}t_{33}t_{36}t_{35}t_{32}t_{37}$), which has 81 tokens. That means the minimum required time should be 81 minutes in rush hours. Fig. 4.6 and Fig. 4.7 show the analysis of the results:

In the un-congested cases, the average value of time cost for observable paths only is 59.0857 minutes and the time cost for both observable and unobservable paths is 79.7414 minutes, 35% higher. Meanwhile, in the congested cases, the values are 98.1429 minutes and 134.431 minutes 37% higher. Moreover, for observable paths only situation, congested traffic cases cost averagely 98.1429 minutes, 66% higher than the un-congested case. For situations with both observable and unobservable paths, congested traffic cases cost averagely 134.431 minutes, 69% higher than the un-congested case. The increasing rates for both situations are similar. These plotted curved could be fitted with linear functions and see the trends of how the time cost growing when the route is changed. The functions are shown in the equations below from Eqn. 4.8 to 4.11, including all the cases mentioned above (the observable and unobservable paths in congested case, observable paths only in congested case, observable and unobservable paths in un-congested case, and observable

Table 4.4.
First Half of List of Route Solutions for Petri Net in Fig. 3.8.

ID	θ	#Firing Sequences	Optimal Solution (No traffic / Congested)
1	*saabbbcccf*	1 (0 w/ unobservable)	55 / 82
2	*saabbcbccf*	10 (10 w/ unobservable)	68 / 109
3	*saabbccbcf*	3 (0 w/ unobservable)	56 / 87
4	*saabbcccbf*	4 (4 w/ unobservable)	72 / 119
5	*saabcbbccf*	2 (1 w/ unobservable)	59 / 99
6	*saabccbbcf*	2 (0 w/ unobservable)	55 / 91
7	*saabccbcbf*	1 (1 w/ unobservable)	76 / 123
8	*saacbbbccf*	1 (1 w/ unobservable)	77 / 119
9	*saacbbbccf*	2 (0 w/ unobservable)	66 / 119
10	*saacbbccbf*	2 (2 w/ unobservable)	76 / 123
11	*saacbcbbcf*	1 (1 w/ unobservable)	74 / 115
12	*saacbccbbf*	1 (1 w/ unobservable)	83 / 134
13	*saaccbbbcf*	1 (0 w/ unobservable)	56 / 96
14	*saaccbbcbf*	2 (2 w/ unobservable)	79 / 132
15	*sababbcccf*	5 (1 w/ unobservable)	57 / 90
16	*sababccbcf*	5 (0 w/ unobservable)	58 / 90
17	*sabacbbccf*	1 (1 w/ unobservable)	75 / 111
18	*sabaccbbcf*	4 (0 w/ unobservable)	57 / 81
19	*sabbabcccf*	3 (0 w/ unobservable)	55 / 93
20	*sabbacbccf*	5 (5 w/ unobservable)	78 / 132

Table 4.5.
Second Half of the List of Route Solutions for Petri Net in Fig. 3.8.

ID	θ	#Firing Sequences	Optimal Solution (No traffic / Congested)
21	$sabbaccbcf$	3 (0 w/ unobservable)	54 / 93
22	$sabbacccbf$	2 (2 w/ unobservable)	82 / 142
23	$sabbbcaccf$	1 (1 w/ unobservable)	76 / 129
24	$sabbcacbcf$	1 (1 w/ unobservable)	75 / 124
25	$sabbcbcacf$	2 (2 w/ unobservable)	75 / 120
26	$sabcabbccf$	6 (6 w/ unobservable)	64 / 111
27	$sabcabccbf$	2 (2 w/ unobservable)	75 / 124
28	$sabcaccbbf$	1 (1 w/ unobservable)	76 / 129
29	$sabcbacbcf$	3 (2 w/ unobservable)	62 / 107
30	$sabcbaccbf$	1 (1 w/ unobservable)	78 / 136
31	$sabcbbaccf$	1 (1 w/ unobservable)	80 / 137
32	$sabcbbcacf$	3 (0 w/ unobservable)	53 / 93
33	$sabcbcabcf$	2 (2 w/ unobservable)	77 / 146
34	$sabcbcbacf$	1 (1 w/ unobservable)	81 / 144
35	$sabcbccabf$	1 (1 w/ unobservable)	93 / 153
36	$sabccabbcf$	2 (0 w/ unobservable)	53 / 89
37	$sabccbacbf$	2 (2 w/ unobservable)	75 / 128
38	$sabccbbacf$	1 (1 w/ unobservable)	64 / 107
39	$sabcccabbf$	2 (2 w/ unobservable)	84 / 155

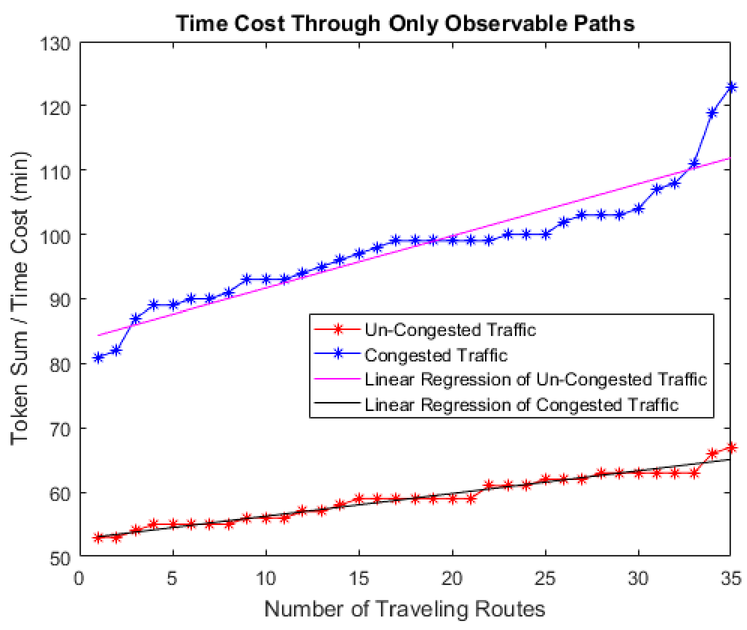

Fig. 4.6. Time Cost Analysis through Only Observable Paths.

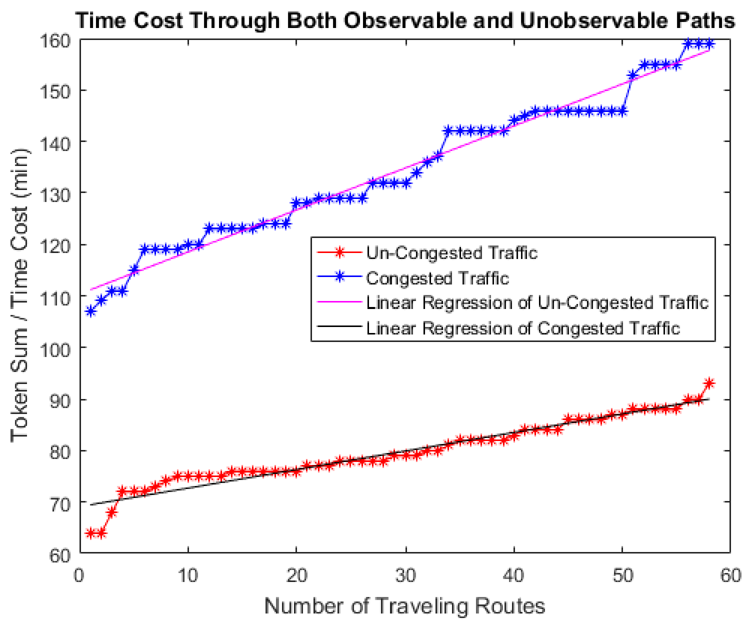

Fig. 4.7. Time Cost Analysis through Both Observable and Unobservable Paths.

paths only in un-congested case, respectively). From Equation 4.8 to 4.11, we can see that the two congested ones have similar increasing rates (higher than 0.8), which is higher than the un-congested ones (around 0.36). It will be always more time-consuming in rush hour when we change the route in the Indianapolis area.

$$y = 0.8166x + 110.34 \tag{4.8}$$

$$y = 0.36048x + 69.107 \tag{4.9}$$

$$y = 0.80924x + 83.576 \tag{4.10}$$

$$y = 0.35322x + 52.728 \tag{4.11}$$

Based on the analysis, we can see that the in the area of Indianapolis metropolitan, in most of the case, use only the paths represented by observable transitions will have much lower time cost, which means that these paths satisfy the needs of visiting the *DOI*s and daily commuting. This conclusion is for the particular label set and label function in our illustration example. For label sets or label functions based on the situation of other cities, the conclusion could be different.

4.4.2 Comparison with Other Shortest Path Algorithm

From the description above, our algorithm has many similarities with the other shortest path algorithms. Therefore, we would like to take the Dijkstra algorithm as an example to compare our algorithm with these algorithms in the aspects of the applicable scenarios and computational complexity. First, we consider the applicable scenarios of the two algorithms. As shown in [100], the Dijkstra algorithm aims to find an optimized route in terms of arc weights between two points beforehand, regardless of any action observation. On the other hand, our algorithm is based on a sequence of observed action labels, from which the optimized route is generated accordingly. Thus, the Dijkstra algorithm cannot be used to solve the problem in this chapter, which has the constraint of the label observation. We could easily find a counterexample that the Dijkstra algorithm cannot find the optimal solution. Consider the following Petri net in Fig. 4.8:

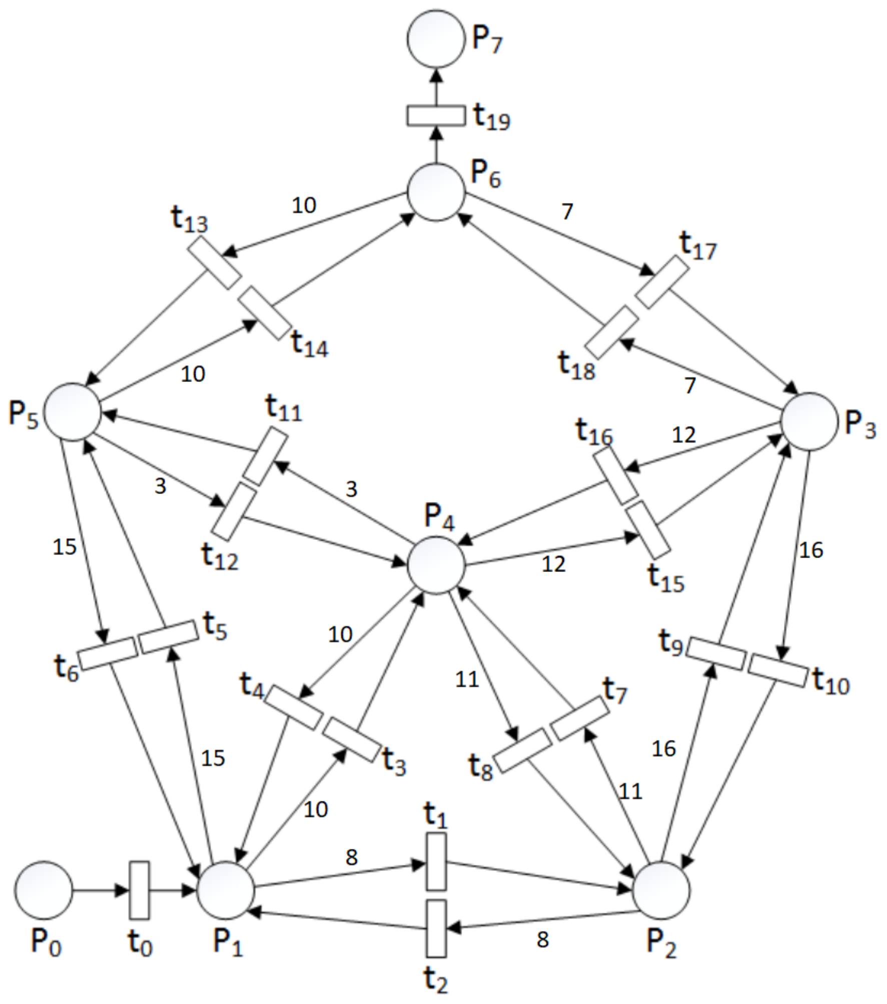

Fig. 4.8. The Petri Net Structure Converted from a Dijkstra Example.

The transitions are labeled following the label functions:

$$Label(t_1, t_4, t_5, t_{10}, t_{13}, t_{16}) = a \tag{4.12}$$

$$Label(t_2, t_6, t_8, t_{11}, t_{14}, t_{17}) = b \tag{4.13}$$

$$Label(t_3, t_7, t_9, t_{12}, t_{15}, t_{18}) = c \tag{4.14}$$

This Petri net is converted from the example on the Wikipedia web page of the Dijkstra algorithm. To satisfy the condition of an undirected graph, every two places are connected with two transitions that form a loop. Applying the Dijkstra algorithm to the Petri net, since only the arc weights are taken into consideration, the optimized route from the starting place p_0 to the finishing place p_7 should be $\delta_{dij} = t_3t_{11}t_{14}$, which has cost 20. However, if we define the label observation sequence to be *acbcc*, the feasible transition sequence set will not include δ_{dij} because it does not satisfy the requirement of initial marking estimates in *Definition 3*. Instead, there are two solutions that are shown below, which satisfy the observation: $\delta_{MIM1} = t_1t_7t_8t_9t_{18}$ or $\delta_{MIM2} = t_5t_{12}t_8t_9t_{18}$. $\delta_{MIM2} = t_5t_{12}t_8t_9t_{18}$ has smaller cost that equals 47, which makes it the optimal solution. Although δ_{MIM2} has a higher cost than δ_{dij}, it is the solution we want in the scenario of this chapter. Therefore, it is clear that the existing shortest path algorithms are not suitable to solve the kind of optimal path problems with observation information mentioned in this chapter. It is also obvious in Fig. 4.8 that the Dijkstra algorithm doesn't give the correct answer following the label sequence.

Furthermore, we could compare the performance of the two algorithms in terms of computational complexity. From the conclusions in [100], we could know that the complexity of Dijkstra algorithm is $O(|E| + |V|log\frac{|E|}{|V|log|V|})$, where $|E|$ is the number of the edges and $|V|$ is the number of vertices, which could be written as $O(|T| + |P|log\frac{|T|}{|P|log|P|})$ in our case, where $|T|$ is the number of transitions and $|P|$ is the number of places. We can see that the Dijkstra algorithm has polynomial computational complexity in terms of the size of the graph structure. For the minimum initial markings estimation algorithm, on the other hand, has relatively higher complexity according to the conclusion from [43], which is $O(k^b)$, where k represents the length of the observation sequence and b is a parameter related to the Petri net structure. Moreover, in the scenario of this chapter, since we have multiple combinations of observation sequences instead of one, and the number of observation sequences is $\frac{k!}{i_1!i_2!...i_j!...i_l!}$, where i_j represents the number of the j^{th} label ($1 \leq j \leq l$, $l \leq k$) in the label sequence and l is the number of different label types that appear in the label sequence. We have $k = i_1 + i_2 + ... + i_l$. It can be proved that the minimum value of $\frac{k!}{i_1!i_2!...i_j!...i_l!}$ appears when all the terms in the numerator are all the same, which also means that in the worst case, there are $\frac{k!}{l*\frac{k}{l}!}$ different combinations of observation label sequences. Hence, the revised computational complexity of the minimum initial estimation algorithm in this chapter is $O(\frac{k!}{l*\frac{k}{l}!} * k^b)$.

4.5 Summary

In this chapter, we first talked about the problem of minimum initial marking estimation in labeled Petri nets that have contact-free unobservable transitions. Extended from the similar problem on the Petri net structures that have observable transitions only, an algorithm has been developed that can obtain the minimum initial marking estimate set with a computational complexity that is polynomial in terms of the length of the observed label sequence. In addition, the other two heuristic algorithms have been proposed, which can find a partial/approximated set of solutions, but with a lower computational cost. An illustrative example has been provided, which is used to evaluate the performance of these three proposed algorithms.

After that, the algorithm has been applied to the traditional Petri net obtained from the traffic network modeling in the previous chapter to estimate the initial required time. The algorithm needs labels as input. The labels are assigned to transitions and a label set is formed with a certain number of each label. The label set was used as the constraint and would like to find an optimal route that meets the requirement. Applying the algorithm to the discretized Petri net generated from the traffic network, the candidates of the possible routes were obtained in un-congested and congested traffic conditions. Then the results have been analyzed in different aspects and assess the traffic condition and driving strategy in the Indianapolis area.

5. RECONSTRUCTION OF THE UNKNOWN PETRI NET STRUCTURES FROM ASYNCHRONOUS OBSERVATIONS OF TOKEN CHANGE SEQUENCES

In some cases, we would like to know the structure but can only observe the asynchronous state evolution of a system. In that case, in this chapter, we would like to propose a way to obtain necessary information from the observation and construct an estimation of the system structure with the minimum scale, which can be consistent with the observation.

This chapter is organized as follows: In the first section, we discuss how this problem could be turned into the Petri net language. In the second section, we talk about the necessary parameters with the way to determine their values or bounds. In the third section, the detailed algorithm will be provided with the complexity analysis. In the fourth section, an illustrative example will be provided to evaluate the performance of the algorithm. In the fifth section, we apply the algorithm to a traffic scenario, which is to estimate the structure traffic network between several stations. In the sixth section, another algorithm will be given for the consistency check of the output of the algorithm in the third section. We summarize in the seventh section.

5.1 Problem Formulation

The problem we deal with in this chapter is as follows. Given the observed token change sequence set $S = \{s_1,\ s_2,\ \dots\ s_n\}$ for the n places in the net. The objective is to find the optimal Petri net structure(s) that is (are) consistent with S. The optimality is in terms of the least scale of the Petri net structure, which has the minimum number of transitions and the minimum number of connections (i.e., the most sparse incident matrix In).

In this chapter, we have made two assumptions for our subsequent reconstruction procedure: 1) The underlying Petri net structure does not have self-loops and source/sink transitions; 2) All transitions should be fired at least once.

Based on the setup above, this chapter is organized as follows: we first find bounds of the number of total transition firings, along with the bounds of the number of transitions. Based on different combinations of the numbers of transition firing and transitions, we could form different transition firing sequences. Then we form marking sequences based on the given observation sequence S. With the candidates of the transition firing sequences and marking evolution sequences, we can solve the state equation and obtain the optimal solution(s) that has(have) the minimum number of nonzero entries in In. We propose a detailed reconstruction algorithm that has a complexity that is polynomial in terms of the number of transitions in the net. Illustrative examples and applications have also been given.

5.2 Problem Analysis

Given the set of place token change sequences S. We first consider the token evolution in S for further analysis and then use a simple example to illustrate the process.

Definition 12: Given S, $S_{inc}(p_i)$ is defined as the set of token-increasing occurrences for each adjacent token observations for each place p_i, where $i = 1, 2, \ldots, n$; Similarly, the set of token-decreasing occurrences for each place is defined as $S_{dec}(p_i)$.

Definition 13: Given S, $L(s_i)$, where $i = 1, 2, \ldots, n$, is defined as the total number of the token change occurrences in s_i for place p_i. Moreover, $L_{max} = max\{L(s_1), L(s_2), \ldots, L(s_n)\}$ is defined to be the maximum number of token change occurrences among all s_i for each place p_i.

Example 13: Consider the token change sequences S shown below:

$$\begin{aligned}
s_1 &: \; 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3, \\
s_2 &: \; 1 \rightarrow 2 \rightarrow 1 \\
s_3 &: \; 1 \rightarrow 2 \rightarrow 0 \rightarrow 1, \\
s_4 &: \; 1 \rightarrow 2 \rightarrow 1 \rightarrow 0.
\end{aligned}$$

It is obvious that $S_{inc}(p_1)$ = $\{1 \rightarrow 2,\ 2 \rightarrow 3,\ 2 \rightarrow 3\}$, $S_{inc}(p_2)$ = $\{1 \rightarrow 2\}$, $S_{inc}(p_3)$ = $\{1 \rightarrow 2,\ 0 \rightarrow 1\}$, and $S_{inc}(p_4)$ = $\{1 \rightarrow 2\}$; There are seven token increasing occurrences in total. Hence we have $|S_{inc}| = 7$, where $|\cdot|$ denotes the size of the set. Similarly, for the decreasing occurrences, we have $S_{dec}(p_1)$ = $\{2 \rightarrow 1, 3 \rightarrow 2\}$, $S_{dec}(p_2)$ = $\{2 \rightarrow 1\}$, $S_{dec}(p_3)$ = $\{2 \rightarrow 0\}$, and $S_{dec}(p_4)$ = $\{2 \rightarrow 1,\ 1 \rightarrow 0\}$; There are six token decreasing occurrences in total and we have $|S_{dec}| = 6$. Besides, we have $L(s_1) = 5$, $L(s_2) = 2$, $L(s_3) = 3$, $L(s_4) = 3$. Therefore, $L_{max} = L(s_1) = 5$.

Proposition 1: The total number of transition firings, denoted by F_a, satisfies the following conditions: $L_{max} \leq F_a \leq \min\{|S_{inc}|, |S_{dec}|\}$.

Proof: Since each observation of token changes implies that at least one transition has fired, in order to be consistent with the given token change sequences S, the number of transition firings, F_a, should at least be as large as the longest observation sequence in S, which is L_{max}. Thus, we have $L_{max} \leq F_a$.

When a certain transition fires, it should remove move token(s) in at least one of its input places and at least one of its output places (since there is no self-loop or source/sink transition based on assumption 1). The largest number of transition firings corresponds to the number of steps of token changes in S, which is captured by $\min\{|S_{inc}|, |S_{dec}|\}$. Thus, we have $F_a \leq \min\{|S_{inc}|, |S_{dec}|\}$. ■

Remark 1: From Proposition 1, it is obvious that $L_{max} \leq min\{|S_{inc}|, |S_{dec}|\}$. Suppose we are given an observation sequences S, and the k^{th} sequence has the largest length, i.e., $|s_k| = L_{max}$. Assume that there are L_{inc} increasing occurrences and L_{dec} decreasing occurrences in s_k, we have $L_{max} = L_{inc} + L_{dec}$. Then, in other sequences s_i where

$i = 1, 2, \ldots, k-1, k+1, \ldots, n$, the minimum number of decreasing occurrences L'_{dec} should be $L'_{dec} = L_{inc}$. Similarly, the minimum number of increasing occurrences L'_{inc} should be $L'_{inc} = L_{dec}$. Therefore, the minimum number of increasing occurrences in S should be $L_{inc} + L'_{inc} = L_{inc} + L_{dec} = L_{max}$ and the minimum number of decreasing occurrences should be $L_{dec} + L'_{dec} = L_{dec} + L_{inc} = L_{max}$. When multiple-input multiple-output transitions exist, $L_{inc} + L'_{inc}$ and $L_{dec} + L'_{dec}$ will be larger, which makes L_{max} the lower bound of their values.

Proposition 2: The bounds of the number of transitions m satisfies $1 \leq m \leq Fa$.

Proof: It is clear that the lower bound is one since a legal Petri net should have at least one transition. For the upper bound, based on assumption 2, each transition should be fired at least once, which implies that m should be no larger than the number of transitions firings Fa. Otherwise, it will be a contradiction of assumption 2. Therefore, the result follows. ■

Known the bounds of Fa and m, the structure of the Petri net could be analyzed based on different possible (Fa, m) pair. Before that, we also need to characterize all possible markings from the observation sequences S. Therefore, the number of possible markings for a certain (Fa, m) pair should be calculated.

For each sequence s_i $(i = 1, 2, \ldots, n)$, let $I_i = 1, 2, 3, \ldots, |s_i|$ be the set of *position indices* for sequence s_i. More specifically, for $k_i \in I_i$, $s_i[k_i]$ is the number of tokens at place p_i after $(k_i - 1)^{th}$ token changes in that place.

Proposition 3: Consider one position index increase for every place p_i (i.e., $s_i[k_i - 1] \rightarrow s_i[k_i]$ for every $i \in \{1, 2, \ldots, n\}$) and $k_i \in I_i$. Define the number of increasing occurrences as n_{inc} and the number of decreasing occurrences as n_{dec}. Then, the upper bound of the number of markings during one position index increasing should be $Mk_{max\#} = (2^{n_{inc}} - 1)(2^{n_{dec}} - 1)$.

Proof: Since $n_{inc} + n_{dec} = n$, we have the bounds of both n_{inc} and n_{dec} to be in between of 1 and $n - 1$ based on assumption 1. Then, we use combinatorial operations to find the upper bound of the number of markings. In particular, it is obvious that there are n_{inc} ways to pick up one increasing occurrence among all candidates, $\frac{n_{inc}(n_{inc}-1)}{2!}$ ways to pick up two increasing occurrences from all candidates, $\ldots$, one way to pick up all increasing occurrences. Summing up all the above possibilities, we have $(n_{inc} + \frac{n_{in}(n_{inc}-1)}{2!} + \ldots + \frac{n_{inc}!}{n_{inc}!})$ ways for candidate selection, which equals $(2^{n_{inc}} - 1)$ according to Binomial Series. Similarly, we can also find the number of ways of candidate selection for decreasing occurrences. Thus, the upper bound of markings during one position index increasing is the product, i.e., $Mk_{max\#} = (2^{n_{inc}} - 1)(2^{n_{dec}} - 1)$. The results follow. ■

Example 14: Consider the observation sequences S in Example 13. For every place p_i, from $s_i[0] \rightarrow s_i[1]$ we have three token increasing occurrences and one token decreasing occurrences. Therefore, we have $n_{inc} = 3$ and $n_{dec} = 1$. It is obvious that n_{inc} and n_{dec} could have different values for different position index increasing.

Proposition 3 provides the upper bound of the number of markings that could be generated during one position index increasing. Therefore, given S, the total number of markings should be upper bounded by $Mk_{max\#} \times L_{max}$. This result will be used for algorithmic complexity analysis in later sections. In the next section, we will go through the following steps and introduce the detailed reconstruction algorithm.

5.2.1 Marking Scanning

Given S, in order to find the marking evolution sequences, we start from the initial marking and consider one position index increasing at a time to find all markings. Repeat this process until we reach the final marking, where inconsistent marking sequences will be neglected. After we enumerated all markings that are consistent with the observation, every path of marking evolution from the initial marking to the final marking could be identified and stored.

5.2.2 Firing Vector Searching

To obtain the firing vectors vf, we try all combinations of the (F_a, m) pair. Note that we characterized the range of F_a and m in the two propositions mentioned above. Hence, for each possible (F_a, m) pair, we calculate the number of ways of distributing F_a into m (we have proved that $m \leq Fa$). A simple example is used to illustrate the process, shown as follows:

Example 15: Consider the observation sequences S, which has six times of transition firings, i.e., $Fa = 6$, and five transitions, i.e., $m = 5$. Since we have the assumption that each transition should be fired at least once, the problem of determining the number of possible transition firing vectors vf is equivalent to separating six elements into five slots with no empty slot. Based on the rules of combinatorial calculation, the total number of different vfs is $5 \times P(5,5)$, where $P(n,k)$ stands for k-permutations of n.

Note that the lower and upper bounds for F_a and m can be found in Propositions 1 and 2, respectively.

5.2.3 State Equation Solving

Now we have obtained the sequence of marking evolution from the given S and the transition firing sequences based on different values of vf, we can solve the state equation and obtain the incident matrices In. In order to do that, we pair each transition firing sequence with different marking evolution sequences. The incident matrix In has dimension $n \times m$ where n is the number of places and m is the number of transitions. After all the transition firing sequences and marking evolution sequences are paired for the calculation, the legal candidates of incident matrices In will be stored. The most sparse ones, which have the largest number of zero entries will be the optimal solutions.

For the same S, it is possible to find multiple optimal solutions, which have the same number of transitions and the same number of nonzero entries. Our algorithm will keep all of these solutions.

5.3 Reconstruction Algorithm

Based on the analysis in the previous sections, we propose the details of the algorithm as follows in Algorithm 5, which summaries the procedure of seeking the set of the optimal incident matrix(matrices), starting from determining the bounds of the parameters and ending up with the solving of the state equation.

5.3.1 Description of the Algorithm

Algorithm 5
Petri net Structure Reconstruction and Optimization based on External Token Change Observation Sequences

```
Input:
    Observed token change sequences $S$.
Output:
    Optimal incident matrices $In_{opt}$, which has the least scale.
1: Load $S$.
2: Obtain the lower and upper bounds of the number of transition firings $Fa$.
3: Obtain the lower and upper bounds of the number of transitions $m$.
4: Obtain valid marking evolution sequences from $S$.
5: Obtain valid firing vectors $vf$ based on the range of $F_a$ and $m$.
6: Obtain valid transition firing sequences based on different values of $vf$.
7: Obtain the candidates of the incident matrix $In$ by solving the state equation.
8: Output those $In_{opt}$, which are the most sparse.
```

5.3.2 Complexity Analysis

It has been proved in the previous sections that for one position index increasing, the number of markings is upper bounded by $(2^{n_{inc}}-1)(2^{n_{dec}}-1)$, which is $O(2^n)$ since $n_{inc}+n_{dec}=n$. It is exponential with respect to the number of places, i.e., n. In our case, when S is given as the external observation, the number of places is fixed by the number of observation sequences. Hence, n is constant. Therefore, the computational complexity of this step could be treated as a constant. To find all the firing vectors, a table with dimension $(Fa-m)\times m^{(Fa-m)}$ has been used. Since the range of m has been proved to be $1 \le m \le Fa$, which shows that in the worst case the dimension of the table is $(Fa-1)\times m^{(Fa-1)}$. Note that $Fa \le L_{max}\times n$, where L_{max} and n are fixed by S, i.e., constant. The complexity in this part should be $O(m^{(L_{max} \times n-1)})$. For the state equation solving, since we have m equations and $m \times n$ unknowns, the complexity of solving these equations will be $O((m \times n)^3)$ = $O(m^3)$. To sum up, the total computational complexity is $O(m^{(L_{max}n+2)})$, which is polynomial in terms of the number of transitions m.

5.4 Illustrative Example

5.4.1 Marking Evolution Sequences

In this section, we use a simple example to illustrate Algorithm 5. Consider the observation sequences S as follows:

$$s_1: \quad 2 \rightarrow 1 \rightarrow 0$$

$$s_2: \quad 1 \rightarrow 0$$

$$s_3: \quad 0 \rightarrow 1 \rightarrow 2 \rightarrow 3$$

It is clear that the s_3 has largest length, which is $L_{max} = 3$. We can also find the number of increasing occurrences to be $|S_{inc}| = 3$ and the number of decreasing occurrences to be $|S_{dec}| = 3$. Therefore, the number of transition firings Fa, which should satisfy $L_{max} \leq Fa \leq \min(|S_{inc}|, |S_{dec}|)$, can only be 3.

In the next step, we would like to find the marking evolution sequences that are valid to the firing rules and are consistent with the observed sequences. By searching through S, we can obtain the marking evolution sequences as shown in Table 5.1:

5.4.2 Transition Firing Sequences

In the next step, we would like to find all the legal transition firing sequences from different combinations of Fa and m. Since Fa has only one value, which is 3, so there is only one loop for Fa. The value of m is between 1 and 3. Here we use $m = 2$ as an example.

Table 5.1.
Marking Evolution Sequences.

	Marking 1	Marking 2	Marking 3	Marking 4
1	$[2,\ 1,\ 0]^T$	$[1,\ 1,\ 1]^T$	$[0,\ 1,\ 2]^T$	$[0,\ 0,\ 3]^T$
2	$[2,\ 1,\ 0]^T$	$[1,\ 1,\ 1]^T$	$[1,\ 0,\ 2]^T$	$[0,\ 0,\ 3]^T$
3	$[2,\ 1,\ 0]^T$	$[2,\ 0,\ 1]^T$	$[1,\ 0,\ 2]^T$	$[0,\ 0,\ 3]^T$

Firstly, we determine the value of the firing vector vf. Because we have assumed that every transition should be fired for at least once, we assign 1 to each entry of vf, and we can get:

$$vf = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Since in this example we have $Fa = 3$, which indicates that we can assign an additional 1 to one of the two entries in vf, which will generate the following two firing vectors shown below:

$$vf_1 = \begin{bmatrix} 2 \\ 1 \end{bmatrix},\ vf_2 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

Note that those firing vectors are not sufficient for state equation solving. We need to determine the transition firing sequences based on the firing vectors. Hence, in the next step, we divide the firing vectors into one-step firing vectors. It is obvious that both vf_1 and vf_2 can be separated into three one-step firing vectors, $[1,\ 0]^T, [1,\ 0]^T$, and $[0,\ 1]^T$, but with different orders. These one-step firing vectors are listed in Table 5.2 with all the possible orders:

We repeat these operations to try all (Fa, m) pairs. Using the marking evolution sequence and firing sequence pair that has the same Fa value, we solve the state equation to obtain the legal values of the incident matrix In. In this example, we have three valid incident matrices In_1, In_2, and In_3, which are shown as follows:

Table 5.2.
One-step Firing Vectors from vf_1 and vf_2 with Different Orders.

	1	2	3
1	$[1,\ 0]^T$	$[1,\ 0]^T$	$[0,\ 1]^T$
2	$[1,\ 0]^T$	$[0,\ 1]^T$	$[1,\ 0]^T$
3	$[0,\ 1]^T$	$[1,\ 0]^T$	$[1,\ 0]^T$

$$In_1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 1 \end{bmatrix}; \quad In_2 = \begin{bmatrix} 0 & -1 \\ -1 & 0 \\ 1 & 1 \end{bmatrix};$$

$$In_3 = \begin{bmatrix} -1 & -1 & 0 \\ 0 & 0 & -1 \\ 1 & 1 & 1 \end{bmatrix}.$$

Obviously, In_3 has larger scale than In_1 and In_2. Hence it should be excluded from the candidates. The incident matrices In_1 and In_2 have the same sparse level, which means the same number of zero entries. Therefore, they are both optimal solutions reconstructed from the given observation. The corresponding Petri net structures built from these two matrices are shown in Fig. 5.1:

Note that the computational cost of the algorithm would be affected by the length of the observation sequences significantly. We provide the time cost of the observation with different length, which are shown in Table 5.3. From the values in Table 5.3, we could also obtain a plot to show growing trend of the algorithm's computational cost in terms of the observation length in a more intuitive way. The plot is shown in Fig. 5.2. It is clear that although the algorithm would be affected by the length of the observation, in this example, it still has a better growing trend than the polynomial function x^5, which shows

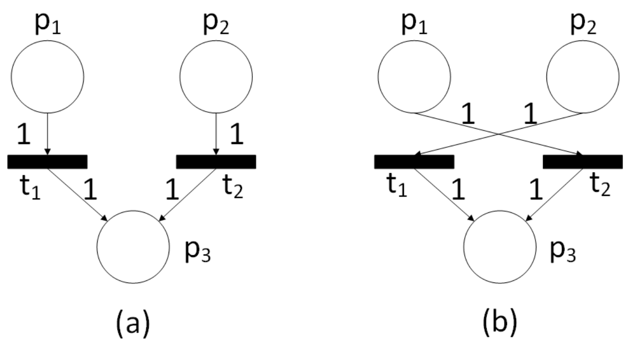

Fig. 5.1. Reconstructed Optimal Petri Net Structures.

its excellence. The simulation is operated in a laptop equipped with a 2.50 GHz processor, 8GB RAM. The simulation environment is based on Matlab R2013a.

Table 5.3.
Computational Cost in terms of the Length of the Observation Sequence.

Length	3	4	5
Time cost (sec)	0.00877	0.04401	1.08690
Length	6	7	8
Time cost (sec)	30.47965	339.94708	14582.21290

5.5 Application in a Traffic-Related Scenario

The traditional Petri net structure is also very suitable for modeling the traffic network. Therefore, we would like to apply the proposed algorithm to a scenario involving the traffic network. The problem could be formulated as follows: Suppose we have several stations that could let vehicles stay, based on observation of the vehicle number change in each station, the optimal traffic network with the simplest structure could be obtained. In this case, these stations could be represented as the places in the Petri net, and vehicles could be denoted by the tokens. We would like to construct the Petri net structure based on the observation of token change in each place. As a practical application, limited by the fea-

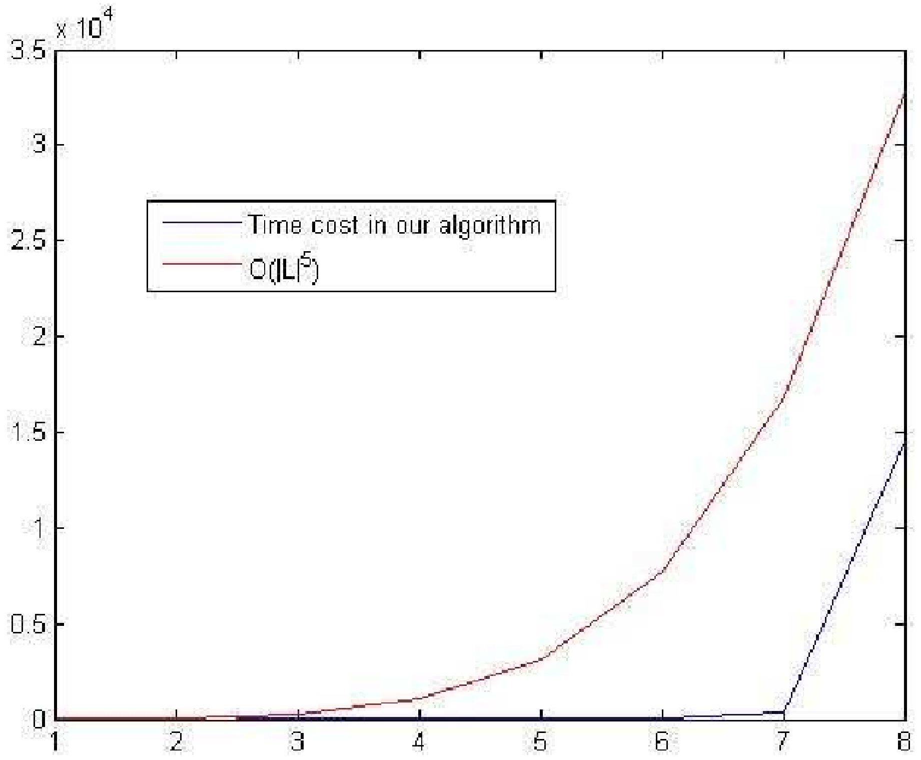

Fig. 5.2. The Growing Trend of the Computational Cost of the Proposed Algorithm in terms of the Observation Length.

tures of the Petri net structure, there will be several constraints in the observation sequence, which are listed as: C1) The number of token changes in each place can only be one in one step. C2) The number of increasing arcs and decreasing arcs should always be the same. C3) Because of C2, the sum of the length of observation sequences for all places should be even.

These constraints are easy to be understood. Since we don't consider the sink and source transitions, the represented traffic network will have a constant number of vehicles. Moreover, each token denotes one vehicle and the one transition firing represents the movement of a vehicle between two stations. Therefore, limited by the mechanism of the Petri net, each time only one token can be transited by firing a transition, which is one token increased, and one token decreased in two places reflected in the observation sequence. In that case, if we want a legal solution from the observation sequence, there should always be the same amount of token increasing and token decreasing, so that they can make token change pairs that denote the vehicle movement. Otherwise, the unpaired token change will be illegal in our case. We use the following observation sequence as an example:

$$\begin{aligned} s_1: &\quad 2 \rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow 2 \rightarrow 3 \\ s_2: &\quad 1 \rightarrow 2 \rightarrow 1 \\ s_3: &\quad 1 \rightarrow 2 \rightarrow 1 \rightarrow 0 \\ s_4: &\quad 1 \rightarrow 2 \rightarrow 1 \rightarrow 0 \rightarrow 1 \end{aligned}$$

which implies that the Petri net should be four places with the initial marking $Mk_{initial}$ and final marking Mk_{final} to be:

$$Mk_{initial} = \begin{bmatrix} 2 \\ 1 \\ 1 \\ 1 \end{bmatrix}; \; Mk_{final} = \begin{bmatrix} 3 \\ 1 \\ 0 \\ 1 \end{bmatrix}. \tag{5.1}$$

Applying the algorithm, we could obtain 93 legal marking sequences as candidates. Keeping only the ones with the least number of transitions (i.e., the ones with incident matrices that have most zeros), we have three different Petri net structures as the result, shown in Fig. 5.3, Fig. 5.4 and Fig. 5.5, with the corresponding marking sequences and transition sequences listed (Equation. 5.2 and 5.5 for Fig. 5.3, Equation. 5.3 and 5.6 for Fig. 5.4 and Equation. 5.4 and 5.7 for Fig. 5.5):

$$t_3 \rightarrow t_5 \rightarrow t_4 \rightarrow t_4 \rightarrow t_1 \rightarrow t_2 \rightarrow t_5 \tag{5.2}$$

$$t_3 \rightarrow t_4 \rightarrow t_1 \rightarrow t_5 \rightarrow t_2 \rightarrow t_3 \rightarrow t_2 \tag{5.3}$$

$$t_4 \rightarrow t_3 \rightarrow t_5 \rightarrow t_2 \rightarrow t_1 \rightarrow t_4 \rightarrow t_1 \tag{5.4}$$

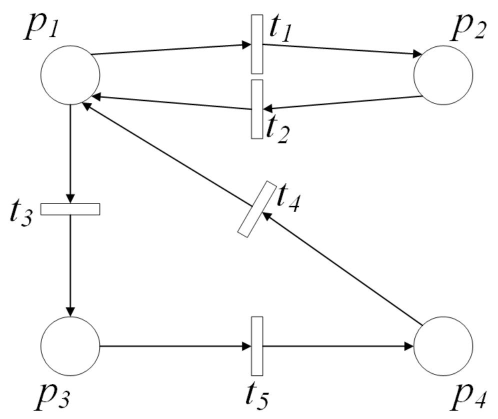

Fig. 5.3. The Constructed Petri Net Structure 1.

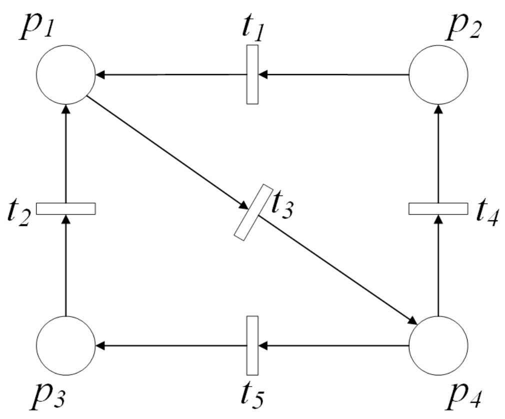

Fig. 5.4. The Constructed Petri Net Structure 2.

$$\begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\2\\1\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\1\\2\end{bmatrix} \rightarrow \begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}3\\1\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}2\\2\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}3\\1\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}3\\1\\0\\1\end{bmatrix} \tag{5.5}$$

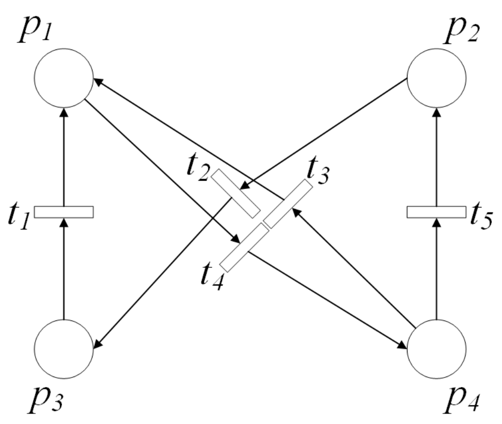

Fig. 5.5. The Constructed Petri Net Structure 3.

$$\begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\1\\2\end{bmatrix} \rightarrow \begin{bmatrix}1\\2\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}2\\1\\2\\0\end{bmatrix} \rightarrow \begin{bmatrix}3\\1\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}3\\1\\0\\1\end{bmatrix} \tag{5.6}$$

$$\begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}1\\1\\1\\2\end{bmatrix} \rightarrow \begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}2\\2\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}2\\1\\2\\0\end{bmatrix} \rightarrow \begin{bmatrix}3\\1\\1\\0\end{bmatrix} \rightarrow \begin{bmatrix}2\\1\\1\\1\end{bmatrix} \rightarrow \begin{bmatrix}3\\1\\0\\1\end{bmatrix} \tag{5.7}$$

5.6 Solution Update

Since the observation sequence can be continuously growing, in this section, we would like to develop an algorithm to check whether the reconstructed optimal Petri net structure(s) would be consistent with newly provided observations. The consistency check algorithm is shown in Algorithm 6. The incident matrix (matrices) *In* that is (are) identified from Algorithm 5 has (have) been used as the input of this algorithm. Based on the values of *In*, we could solve the state equation to calculate the difference of markings. Because we have knowledge of the current marking, it is possible to obtain future markings to see if they are consistent with the new observations. The detailed consistency checking algorithm is shown as follows, which takes the solution from the previous algorithm as the input and output *YES* or *No* as the result of the consistency check.

Algorithm 6
Algorithm of Consistency Checking

```
Input:
    Optimal incident matrices $In$ obtained from Algorithm 5;
    New observed sequences with largest length $L'_{max}$ and the final marking $Mk_k$,
Output:
    Whether $In$ matches the newly observed sequences.
1: Load $In$ matrix (matrices) and the new observed sequences $S_1$.
2: Compute the bounds of the number of transition firings.
3: Try each possible firing to check whether the optimal $In$ matrix (matrices) satisfies
   (satisfy) the state equation.
4: If there is one transition firing sequence that let the $In$ to be consistent with the obser-
   vation, output YES.
5: Otherwise, output NO.
```

The computational complexity of this algorithm is obtained as follows: Firstly, the scanning of the marking evolution sequences runs in the constant time since the longest length of the new observed sequences, i.e., L'_{max}, is treated as a constant. Then, based on the results in [69], the complexity of obtaining all possible marking sequences is upper bounded by $O(m^{L'_{max}})$, which is polynomial in terms of m because L'_{max} is treated as a constant. Secondly, solving the state equation has complexity $O(n^3)$, which is also constant since n is treated as constant. Thus, the total computational complexity is $O(m^{L'_{max}})$, which is polynomial in terms of m.

5.6.1 Illustrative Example

We would like to use a simple example to illustrate Algorithm 6. Recall that we have obtained two optimal incident matrices In_1 and In_2 from Algorithm 5. We use In_1 for the illustration in this example. The process is shown below.

$$In_1 = \begin{bmatrix} -1 & 0 \\ 0 & -1 \\ 1 & 1 \end{bmatrix}.$$

Suppose we are given the following new observed sequences:

$$\begin{aligned} s_1 &: \ 0 \to 1 \\ s_2 &: \ 0 \to 1 \to 2 \\ s_3 &: \ 3 \to 2 \to 1 \to 0 \end{aligned}$$

From the newly generated observation sequences, we could obtain the bounds of the number of transition firings Fa', where $3 \leq Fa' \leq 3$, i.e., $Fa' = 3$. Thus, it indicates that there is only one loop for Fa'. Scan the new observed sequences, we can obtain three different markings that are as follows:

$$Mk_1 = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}, Mk_2 = \begin{bmatrix} 0 \\ 1 \\ 2 \end{bmatrix}, Mk_3 = \begin{bmatrix} 1 \\ 1 \\ 2 \end{bmatrix}.$$

In the next step, we calculate potential future markings by solving the state equation with the incident matrix In_1. The results are shown as follows:

$$Mk_{11} = \begin{bmatrix} -1 \\ 0 \\ 4 \end{bmatrix}, Mk_{12} = \begin{bmatrix} 0 \\ -1 \\ 4 \end{bmatrix}.$$

We can see that there are negative entries in these results, which yields that the incident matrix In_1 is not consistent with the new observed sequences. In that case, for further analysis, we need to re-identify the optimal Petri net structures with new observations by applying Algorithm 5.

5.7 Summary

In this chapter, we developed an approach to reconstruct optimal Petri net structure(s) from the asynchronous observations of the token change sequence with finite length in each place. The optimality is in terms of the least scale, which has the least transitions, the most sparse incident matrix, and is consistent with the given observations. An algorithm has been developed to find such structure(s). Moreover, we developed another algorithm to check the consistency of the identified Petri net structure with newly generated observations with finite-length, for scalability.

6. TRAFFIC VOLUME CONTROL BASED ON OBSERVATION OF LABELED PETRI NET

With the rapid development of autonomous vehicle techniques, traffic planning and traffic volume control become critical issues. With reasonable traffic control, the problem of traffic jams and collisions could be significantly relieved. In this chapter, a traffic volume control algorithm based on the labeled Petri net will be introduced. Firstly, the traffic network is modeled with the Petri net structure, where places represent the road segments and transitions represent the connection between them. Labels are assigned to transitions based on their heading directions. Then the control algorithm is introduced. Different from some existing control strategies which aim to avoid collision and don't allow multiple tokens appearing in the same place, we would like to keep the traffic volume within a certain level and reduce congestion. Traffic capacities have been assigned to the places that represent the road segments. A controller is also applied to keep the number of tokens smaller than the values of capacities. The existence of unobservable transitions also generates more possibilities.

This chapter is organized as follows: Firstly, the proposed method with a simple example will be described. In the next section, the detailed algorithm and the complexity analysis will be provided. An illustrative example will be given then to show the performance of the proposed algorithm.

6.1 Problem Formulation

Inspired by the study in [102], which focused on traffic network controller design to avoid the collision, the objective in this chapter will also be related to traffic network control. A place in the Petri net represents a road segment, and a transition represents an intersection or a junction between road segments. Each transition has been assigned an alphabet label or empty label ε. The same label could be shared between multiple transitions. Different from the former study who divided the transition set into the controllable part and the uncontrollable part, we are considering a more challenging task, which is to treat some of the transitions to be unobservable. In that case, we can neither observe nor control the events related to some of the transitions. Another difference from the state-of-art studies, we don't limit our work to collision avoidance to reflect a more realistic situation, since in real work each road segment should have the capacity to contain a certain amount of vehicles. Before we get into it any deeper, we would like to mention some preliminary assumptions that we will need in this chapter: A1) All unobservable transitions are *contract-free*. A2) Unobservable transition firing will be considered before analyzing each observed label.

Assumption A1 ensures that our method will not be stuck in loops of firing unobservable transitions since the observation sequences only cannot limit the number of the firing of unobservable transitions. Assumption A2 helps to reduce the confusion caused by unobservable transition firings. A firing sequence corresponding to one label should either consist of one and only one observable transition or end up with an observable transition.

Based on these conditions, we could transfer a road network into a Petri net structure. A simple road network is shown in Fig. 6.1, which is consisted of a three-directional intersection and a road junction, which connected with two one-directional small junctions.

It is clear in Fig. 6.1, we can observe and control the junction between road segments 1 and 2, and the intersection between road segments 3, 4, and 5 with cameras and traffic signs. But, the two small junctions between the two main roads, which could be roads in a residential area or small country roads, normally would not have any cameras or signs. In this road network, when traffic control is needed, we could block certain lanes in the intersection and junction by changing the duration of the traffic lights or use temporary signs. Besides, since each road segment could allow a certain amount of traffic within it, it is not realistic if we define a collision to be two vehicles that appear in the same road segment. Hence, we need a term to define the capacity of a road segment.

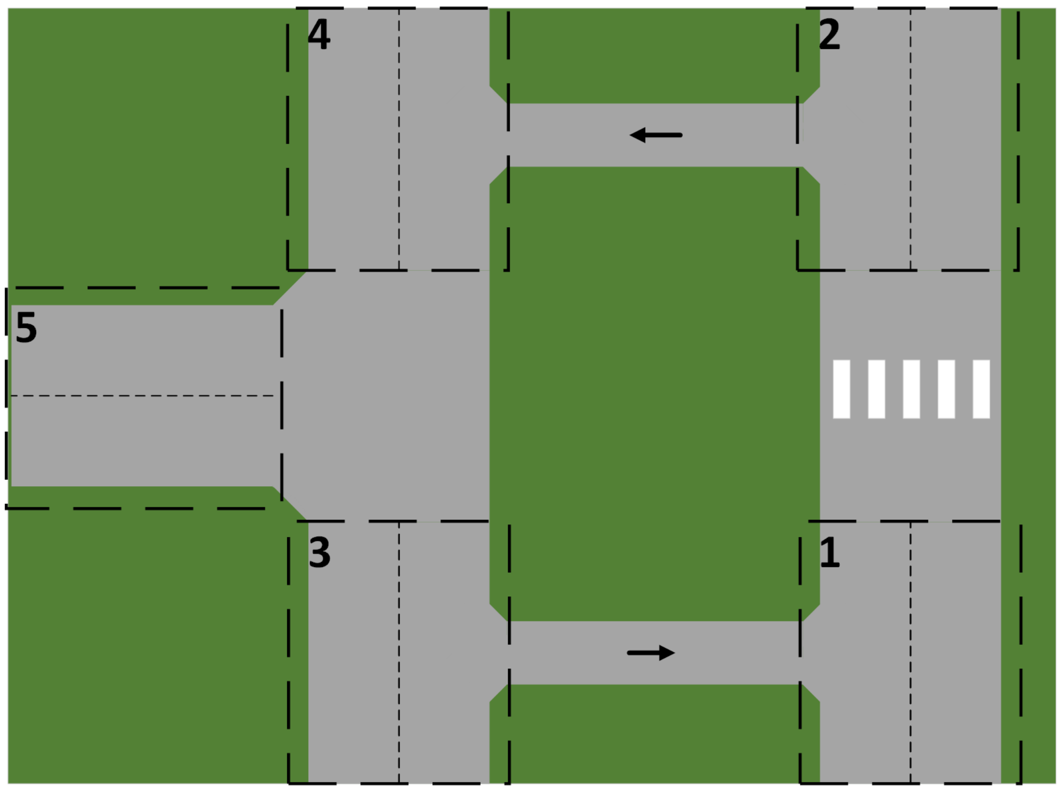

Fig. 6.1. A Simple Traffic Network.

Definition 14: The capacity of each road segment is defined with a column vector ζ, called the *capacity vector*:

$$\zeta = \begin{bmatrix} c_1 \\ c_2 \\ \ldots \\ c_i \\ \ldots \\ c_m \end{bmatrix}, i = 1, 2..., m \tag{6.1}$$

Where ζ is a m-dimensional vector and each positive entry corresponds to one road segment.

Therefore, the constraints of our model are adjustable by changing the capacity of each road segment, which will be more flexible based on needs. Known the information above, we could model the road network in Fig. 6.1 with the Petri net structure in Fig. 6.2 below:

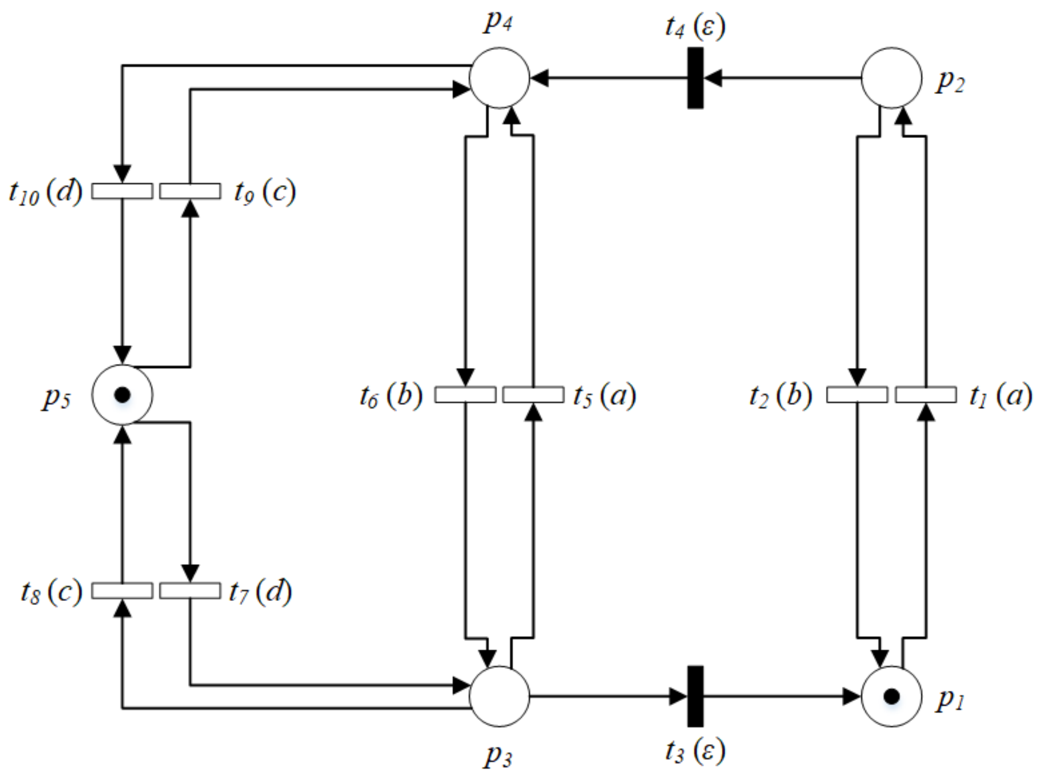

Fig. 6.2. The Petri Net Structure Generated from the Road Network in Fig. 6.1.

Reflecting all the terms in the road network into the Petri net, the objective is to avoid exceeding token capacity by enabling/disabling each controllable transition and obtain all legal markings. The initial marking and an observed label sequence are given. Following the observed label sequence, in each step, we consider the movement of only one vehicle, in order to meet the rules of the Petri net structure. Thus we could conclude all the constraints about the transition sequence δ, observed label sequence θ and the new generated marking Mk' in our problem: C1) The transition sequence δ must satisfy the observed label sequence θ. C2) Mk' is reachable from the initial marking Mk_0 through firing δ. C3) If a reachable marking Mk' has an entry larger than the corresponding entry in ζ, this Mk' is illegal. C4) If a reachable marking Mk' enables an unobservable transition t_u, which yields another marking Mk'' by firing t_u, and Mk'' has an entry larger than the corresponding entry in ζ, this Mk' is also illegal.

Definition 15: The controller for the Petri net is defined with a column vector η, called the *controller vector*:

$$\eta = \begin{bmatrix} e_1 \\ e_2 \\ \dots \\ e_i \\ \dots \\ e_n \end{bmatrix}, i = 1,2...,n \tag{6.2}$$

Where η is a n-dimensional vector with only zero and one entries.

The controller η we used to achieve these constraints could be considered as an n-dimensional binary vector, whose entries correspond to the transitions in the Petri net. When an entry is set to be one, the corresponding transition could be enabled by the tokens in the net. Otherwise, if the entry is set to be zero, the transition cannot be fired at any time. In this way, we could block or release the transitions depending on whether the constraints are satisfied.

In order to show how the method works in a more intuitive way, we will use the following example to show the idea.

Example 16: Consider the traffic network represented by a Petri net structure in Fig. 6.2, which is generated from the road network in Fig. 6.1. Places p_1 to p_5 represent five different road segments. Transitions t_1 and t_2 form the junction between p_1 and p_2. Transitions t_5 to t_{10} form the three-directional intersection between p_3, p_4, and p_5. t_3 and t_4 are unobservable transitions represent the small junctions connecting the two parts. The labeling function follows the rule that the same directions share the same label. For example, since t_1 and t_3 are both going straight towards the north, they share label a. Because t_8 and t_9 are both turning left, they share label c. Hence we have: $Label(t_1,t_5) = a$, $Label(t_2,t_6) = b$, $Label(t_8,t_9) = c$, $Label(t_7,t_{10}) = d$, and $Label(t_3,t_4) = \varepsilon$.

Initially, we insert one token to places p_1 and p_5, which means there is initially one vehicle in each of these two road segments. Suppose the observation label sequence is $\theta = abda$, we could have the following Table 6.1 shows all reachable markings:

Table 6.1.
Reachable Markings Based on the Observed Sequence Without Constraints.

Observed Label(s)	Transition Sequence	Markings
a	t_1	$[0,1,0,0,1]^T$
	t_5	None
ab	t_1t_2	$[1,0,0,0,1]^T$
	$t_1t_4t_6$	$[0,0,1,0,1]^T$
abd	$t_1t_2t_7$	$[1,0,1,0,0]^T$
	$t_1t_2t_{10}$	None
	$t_1t_4t_6t_7$	$[0,0,2,0,0]^T$
	$t_1t_4t_6t_{10}$	None
abda	$t_1t_2t_7t_1$	$[0,1,1,0,0]^T$
	$t_1t_2t_7t_5$	$[1,0,0,1,0]^T$
	$t_1t_4t_6t_7t_3t_1$	$[0,1,1,0,0]^T$
	$t_1t_4t_6t_7t_5$	$[0,0,1,1,0]^T$

There is something we need to emphasize in Table 6.1. 1) *None* in the *Markings* column means that the corresponding transition sequence cannot be enabled to generate a new marking. 2) When transitions t_3 or t_4 appear, it means that transition sequence could generate a new marking through firing unobservable transitions. 3) The transition sequence $t_1t_4t_6t_7$ generates a new marking $[0,0,2,0,0]^T$. If we follow the constraint in [102], this is an illegal marking, which will lead to a collision. However, in our case, if we set the capacity of place p_3 to be no smaller than two, it is a legal marking. That is to say, if the capacity of a place is set to be one, our problem is yielded to a normal collision avoidance problem. 4) Some of the generated markings could enable unobservable transitions. For instance, $[1,0,1,0,0]^T$ after firing $t_1t_2t_7$ will enable t_3. Since we cannot control t_3, if the capacity of p_1 is one, we must block t_7 before it is fired, which means the controller vector has the following value:

$$\eta = \begin{bmatrix} 1 \\ 1 \\ \cdots \\ 1 \\ 0 \\ 1 \\ \cdots \\ 1 \end{bmatrix} \begin{array}{l} \\ \\ \\ \\ \leftarrow 7^{th}\ entry \\ \\ \\ \\ \end{array} \tag{6.3}$$

If we set the capacity vector following the values in Eqn. 6.4, the logic of the controller could be applied to each step of firing. After removing all illegal markings, we can have the following reachable markings in Table 6.2:

$$\zeta = \begin{bmatrix} 1 \\ 1 \\ 2 \\ 3 \\ 3 \end{bmatrix} \tag{6.4}$$

Where each entry corresponds to the capacity of each place, and remove those that could lead to exceeding capacity:

6.2 Descriptions of Proposed Algorithms

In this section, we will explain our recursive algorithm in detail. We are given the initial marking Mk_0 and the observed label sequence θ as input and would like to have all the legal reachable markings as output. In each iteration, all transitions that fit current label are enumerated, including the observable transition $t = \{t \in Tr_o | Label(t) = l_{current}\}$ and

Table 6.2.
Reachable Markings Based on the Observed Sequence with Constraints.

Observed Label(s)	Transition Sequence	Markings
a	t_1	$[0,1,0,0,1]^T$
ab	t_1t_2	$[1,0,0,0,1]^T$
	$t_1t_4t_6$	$[0,0,1,0,1]^T$
abd	$t_1t_4t_6t_7$	$[0,0,2,0,0]^T$
abda	$t_1t_4t_6t_7t_3t_1$	$[0,1,1,0,0]^T$
	$t_1t_4t_6t_7t_5$	$[0,0,1,1,0]^T$

the unobservable $t \in Tr_u$. New reachable markings will be calculated from these transitions with two steps of examines applied. Firstly, compare each marking with the capacity vector ζ element-wisely. Secondly, check whether any unobservable transitions $t_u \in Tr_u$ are enabled and if so, fire these unobservable transitions and generate new parent markings to repeat the first examine to the newly generated marking. Only markings that are element-wisely smaller than ζ will be kept for the calculation for the following iterations. The detailed algorithm has been provided below.

Now we could analyze the complexity of the proposed algorithm. Each marking stored in the list *tree* (except the leaf nodes in the last layer) will be used as parent marking to generate new markings. The calculation has computational complexity to do that is upper bounded by $O(n^3)$, which is the complexity of solving the state equation. Besides, since we are considering the unobservable transitions, before each observed label, there could be multiple unobservable transitions fired. On the other hand, duplication may happen if we enumerate all reachable markings since different firing sequences could lead to the same marking in the same layer. Based on the results in [82] and [103], if the observation label sequence has length k the number of consistent markings is upper bounded by $O((1+a_1+a_2k)^{n-n_u}(1+c_1+c_2k)^{n_u})$, if the structure of the unobservable subnet is *structurally bounded*, where a_1, a_2, c_1 and c_2 are parameters depending on the Petri net structure, label

Algorithm 7
Capacity exceeding avoidance based on initial marking and label observation

Input: The incident matrix In of a labeled Petri net structure $PN_L = (Pl, Tr_o, Tr_u, Ar, We, Label, \delta)$, label observation sequence θ, the initial marking $Mk_0 = [Mk_{0\text{-}1}, Mk_{0\text{-}2} ..., Mk_{0\text{-}m}]^T$ and the capacity vector ζ.

Output: The set of legal reachable markings $Tree$.

```
1: $Tree = \{Mk_0\}$, $Parent = \{\phi\}$
2: **for** $i = 1$ to $|\theta|$ **do**
3: $l_{current} = \theta(i)$
4: $Tr_{fsbl} = t|t \in Tr_{l_{current}}$
5: **for** $j = 1$ to $|Tr_{fsbl}|$ **do**
6: $t = Tr_{fsbl}(j)$
7: $vf = [0, 0, ...1, ..., 0]^T$ , the entry corresponds to $t$ equals one.
8: **for** $k = 1$ to $|Tree(i, :)|$ **do**
9: $Mk_{parent} = |Tree(i, k)|$
10: Call Algorithm 2 with input $In$, $Mparent$, $\zeta$, and obtain the additional parent marking
    set $Mk_u$ through firing unobservable transitions.
11: **for** Each $Mk \in Mparent \cup Mk_u$ **do**
12: Call Algorithm 3 with input $In$, $Mk$, $t$, $vf$, $Tree$, $\zeta$.
13: **end for**
14: **end for**
15: **end for**
16: **end for**
17: Output $Tree$.
```

Algorithm 8
Loop for algorithm 1 to calculate new markings through unobservable transitions

```
Input: The incident matrix $In$ of a labeled Petri net structure $PN_L = (Pl, Tr_o, Tr_u, Ar, We, Label, \delta)$, the current parent marking $Mparent$ and the capacity vector $\zeta$.
Output: The new legal marking set $Mk_u$.
1: $Mk_u = \phi$
2: for Each $t_u \in Tr_u$ do
3:     Form a $v_u = [0, 0, ...1, ..., 0]^T$ , the entry corresponds to $t_u$ equals one.
4:     if $Mk_{parent}$ enables $t_u$ then
5:         Calculate new marking $Mk_{childu}$ with equation $Mk_{childu} = Mk_{parent} + In \cdot v_u$
6:         Check the validation of the legality of $Mk_{childu}$ by comparing it with $\zeta$ and check duplication with the previous marking in $Mk_u$.
7:         if $Mk_{childu}$ is a legal marking then
8:             Save it to $Mk_u$.
9:         end if
10:    end if
11: end for
```

function, and the token distribution in the initial marking. Since all unobservable transitions in our case are assumed to be *contact-free*, the condition is satisfied. Therefore, the total complexity can be calculated as:

$$O((1 + a_1 + a_2 k))^{n-n_u}(1 + c_1 + c_2 k)^{n_u})n^3) \tag{6.5}$$

We can see that the algorithm has polynomial complexity related to the length of label observation l, label function, and the number of tokens in the net, but exponential to the Petri net structure.

Algorithm 9
Main loop for algorithm 1 to calculate new markings

Input: The incident matrix In of a labeled Petri net structure $PN_L = (Pl, Tr_o, Tr_u, Ar, We, Label, \delta)$, current parent marking Mk, current observable transition t, current firing vector vf, the legal marking set $Tree$ and the capacity vector ζ.

Output: A new legal marking Mk'.

```
1: if $Mk$ enables $t$ then
2: Calculate new marking $Mk_{child}$ with equation $Mk_{child} = Mk + In \cdot vf$
3: Check the validation of the legality of $Mk_{child}$ by comparing it with $\zeta$ and check duplication with previous markings in the same layer of $Tree$.
4: if $Mk'$ is a legal marking then
5: Save it to the first available slot in $Tree(i+1,:)$.
6: end if
7: end if
```

6.3 Illustrative Example and Simulation Result

In this section, we will use an illustrative example to show the performance of the proposed algorithm. Consider the Petri net structure in Fig. 6.3. The Petri net structure represents a road network that consists of four three-directional intersections and one four-directional intersection. Between every two three-directional intersections, there is also a small one-way junction connecting them. In the figure we can see that the four three-directional intersections are formed with place and transition sets shown in Eqn. 6.6 to Eqn. 6.9.

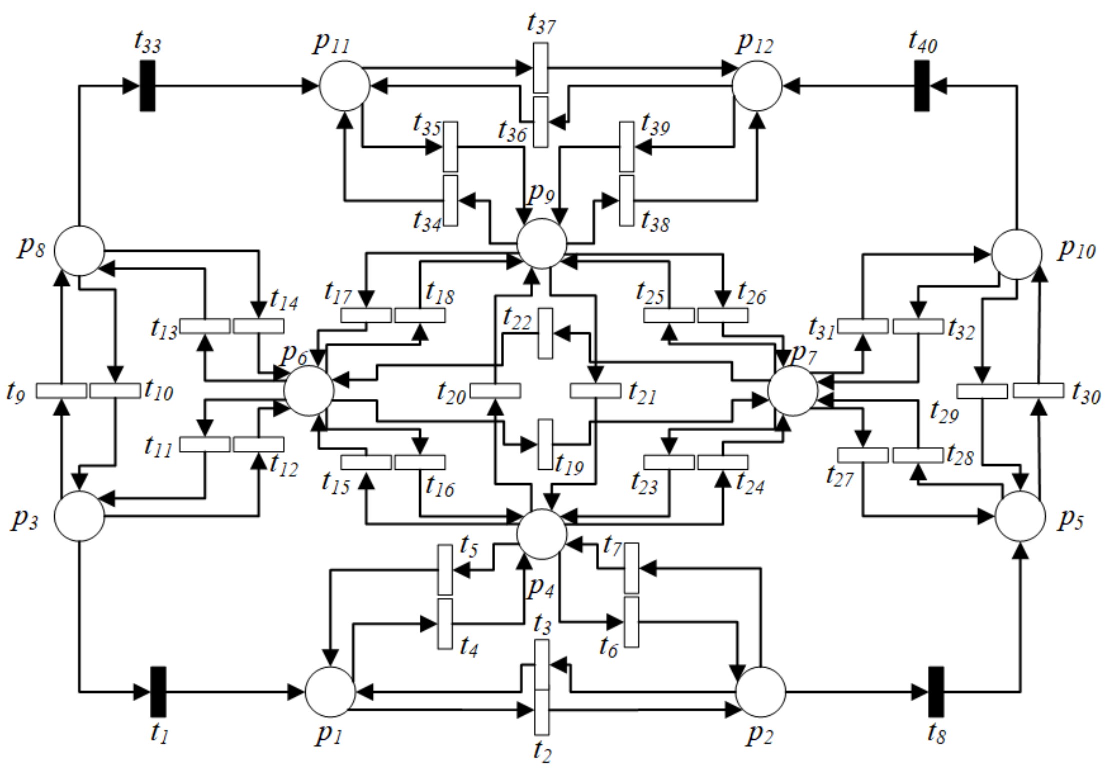

Fig. 6.3. The Petri Net Structure for Illustration.

$$bottom = \{p_1, p_2, p_4, t_2, t_3, t_4, t_5, t_6, t_7\} \tag{6.6}$$

$$left = \{p_3, p_6, p_8, t_9, t_{10}, t_{11}, t_{12}, t_{13}, t_{14}\} \tag{6.7}$$

$$right = \{p_5, p_7, p_{10}, t_{27}, t_{28}, t_{29}, t_{30}, t_{31}, t_{32}\} \tag{6.8}$$

$$top = \{p_9, p_{11}, p_{12}, t_{34}, t_{35}, t_{36}, t_{37}, t_{38}, t_{39}\} \tag{6.9}$$

Similarly, the four-directional intersection is formed by set:

$$middle = \{p_4, p_6, p_7, p_9, t_{15}, t_{16}, t_{17}, t_{18}, t_{19}, t_{20}, t_{21}, t_{22}, t_{23}, t_{24}, t_{25}, t_{26}\}$$

The four one-way junctions are represented with unobservable transitions t_1, t_8, t_{33}, and t_{40}. Besides, the labeling function is defined as follows, from Eqn. 6.10 to Eqn. 6.15. The assignment of the labels is based on the directions of the transitions' input and output arcs. For example, since t_9 has its input and output arcs straightly upwards (representing the direction of the road segments are north-wards), it shares the label a with t_{20} and t_{30} that also have the same direction. Transition t_4 represents the connection of road segments that are leading the traffic turning left. Hence, it shares the label e with other transitions that have the same condition.

$$Label(t_9, t_{20}, t_{30}) = a \tag{6.10}$$

$$Label(t_{10}, t_{21}, t_{29}) = b \tag{6.11}$$

$$Label(t_2, t_{19}, t_{37}) = c \tag{6.12}$$

$$Label(t_3, t_{22}, t_{36}) = d \tag{6.13}$$

$$Label(t_4, t_6, t_{11}, t_{14}, t_{15}, t_{18}, t_{23}, t_{26}, t_{28}, t_{31}, t_{34}, t_{39}) = e \tag{6.14}$$

$$Label(t_5, t_7, t_{12}, t_{13}, t_{16}, t_{17}, t_{24}, t_{25}, t_{27}, t_{32}, t_{35}, t_{38}) = f \tag{6.15}$$

At last, we also assign the *observation label sequence (θ)*, the *capacity vector (ζ)* and the *initial marking (Mk_0)* of this Petri net to be:

$$\theta = aedbefbadc, \tag{6.16}$$

$$\zeta = \begin{bmatrix} 5 & 5 & 6 & 10 & 6 & 10 & 10 & 6 & 10 & 6 & 5 & 5 \end{bmatrix}^T, \tag{6.17}$$

$$Mk_0 = \begin{bmatrix} 2 & 4 & 0 & 6 & 3 & 7 & 7 & 3 & 6 & 2 & 1 & 4 \end{bmatrix}^T. \tag{6.18}$$

Thus, the objective is to find the reachable markings at each observation steps following the observation labels from the given initial marking, and obey the constraints of the capacity vector at the same time. Our simulation has been done with Matlab 2019b and the results have been provided below in Table 6.3 and 6.4.

The data in Table 6.3 shows the number of all legal markings consistent with the observed label sequences while the data in Table 6.4 shows the number of the parent markings via unobservable transitions that are used to calculate the legal markings in the next layer (that is the reason that the last layer of Table 6.4 is N/A). We can see that more than one-third of the parent markings generation involving unobservable transition sequence firing.

Table 6.3.
The Number of Legal Reachable Markings in each Layer.

#Layer	Observed Label Sequence	#Legal Markings
1	a	2
2	ae	24
3	aed	72
4	aedb	180
5	aedbe	1,135
6	aedbef	10,053
7	aedbefb	17,775
8	aedbefba	24,088
9	aedbefbad	46,673
10	aedbefbadc	82,618

We can see the growing trends of the two sets of data with the following plots in Fig. 6.4 and 6.5.

We can see that in both of the two plots, the curves of the number of the markings have a much slower growing speed than the polynomial function $y = x^5$, which can prove the conclusion in the complexity analysis.

6.4 Summary

In this chapter, we proposed an approach to model the traffic network in a certain area with the labeled Petri net structure and then designed a control algorithm to limit traffic volume and reduce the collision rate. To meet the real traffic network conditions, the Petri net structure that we are using contain both observable and unobservable transitions, where observable transitions represent intersections and junctions on main roads, while unobservable transitions represent connections between main roads like residential areas or country roads that normally cannot be observed with monitoring devices. By going through

Table 6.4.
The Number of Legal Reachable Markings Via Unobservable Transitions in each Layer.

#Layer	Observed Label Sequence	#Legal Marking via Unobservable Transitions
1	a	0
2	ae	0
3	aed	0
4	aedb	0
5	aedbe	84
6	aedbef	3,081
7	aedbefb	6,612
8	aedbefba	10,350
9	aedbefbad	28,349
10	aedbefbadc	N/A

unobservable paths, many more possibilities could appear, which makes the problem more challenging and interesting. Also, different from other research works that studied collision avoidance only, we assign the road segments (represented by places in the Petri net) with traffic volume capacities, which are positive integer numbers and form the *capacity vector*. In that case, each road segment could have multiple vehicles at the same time, which is more realistic than other existing studies. In our algorithm description and illustrative example sections, we presented our approach in detail and have proved that it is reliable and feasible to be applied to the traffic system.

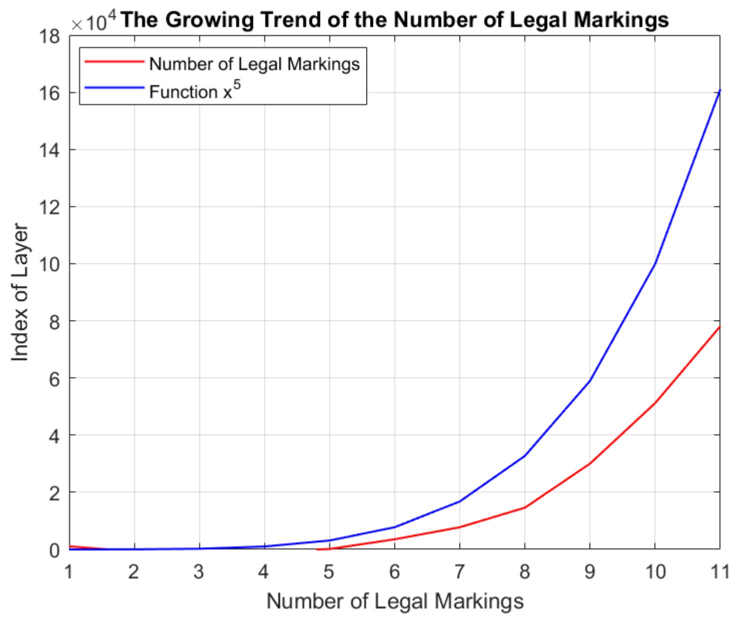

Fig. 6.4. The Growing Trend of the Number of Legal Markings in Each Layer.

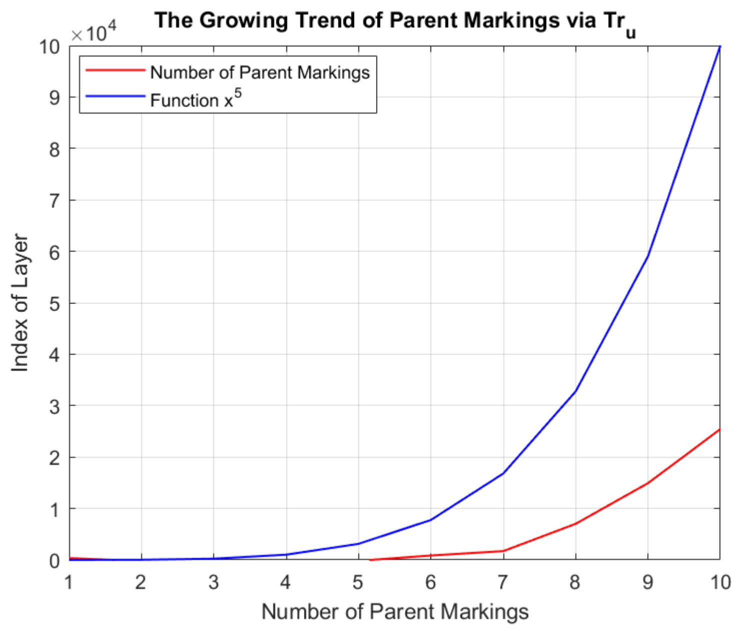

Fig. 6.5. The Growing Trend of the Number of Parent Markings Via Unobservable Transitions.

7. CONCLUSIONS AND FUTURE WORK

In this dissertation, many problems involving traffic planning and control applying the Petri net structure have been studied. Firstly, two types of Petri nets have been utilized for traffic network modeling. The first one is called the Probabilistic Petri net, whose transitions have multiple output arcs with probabilities assigned, representing different options of token transiting. This model has been used in a highway segment, which has been discretized into blocks. The places that representing those blocks are connected based on an open-source database obtained from the Internet. The probabilities are calculated from the number of vehicles that traveled from place to place. After that, the probabilities are converted into rewards by taking logarithm calculation, to make the values proportional to the safety level. Part of the generated Probabilistic Petri net structure has been shown, which could be useful for analyzing the dynamic of certain traffic flows.

The second one is the transitional Petri net structure that is used to discretize the highway traffic network in a larger area. We used the highway network around the metropolitan area of Indianapolis as an example. The highway network consists of highway or freeway (Interstate-65, Interstate-70, Interstate-465, and US-31) and major local roads (e.g. Meridian Street, Michigan Road, Allisonville Road, Binford Boulevard...). The scenario that has been studied is set to be a daily driving case, expressed as follows:

Suppose we start at home (the starting place) and would like to go to school (the finishing place). During the way, we want to visit some other $destinations\ of\ interest (DOI)$, such as a supermarket, a McDonald's, or a gas station. Since there are multiple choices for these *DOIs* around the city, we could represent those *DOI* that we may potentially visit as places in Petri net and connect them with transitions based on the connection relationship of the highway network. When traveling between the places, time will be consumed. In the discretized Petri net, time is also discretized into time units, which is represented as tokens. We set the time unit equaling to one minute in real-world, rounded-up if less than one minute. The weights of the arcs connecting places and transitions show how many time units will be consumed. The weights depend on the distance between two *DOI* as well as the traffic condition on this path. Therefore, the highway network has been converted into a discrete event system modeled by the Petri net. In this Petri net, place set *Pl* represents *destinations of interest DOI*. Transition set Tr represents the possible traveling path between each two *DOI*. Arc weights represent the time costs, which are determined by the distance and traffic condition. Tokens represent the time units and will be consumed when firing a transition.

After we have obtained the Petri net that modeling the highway traffic network, we further developed two algorithms that are used to optimize the structure and initial marking of the Petri net. Both of the two algorithms need external observations as input.

We first proposed an algorithm to optimize the initial marking of a labeled Petri net. The external observation given for this algorithm is a sequence of labels. Each label is assigned to a subset of the transition set of the Petri net. The goal is to find the minimum initial marking(s) that could fire a transition sequence that satisfies the given label sequence. The challenging part is that unobservable transitions are considered in our study. To simplify the problem, we assume that the unobservable transitions are contact-free. Besides, before each label observation, there will be one and only one unobservable transition fired. With these assumptions, we are able to run the algorithm in polynomial complexity related to the length of the observation sequence. An illustrative example has been provided to show and the performance of the algorithm. The algorithm could work with polynomial time complexity and get all the possible solutions that satisfy the requirements. Even though, the algorithm still produces too much redundant information during the calculation. Hence, we proposed two heuristic algorithms that save a lot of computation resources but will only obtain some sub-optimal solutions. These sub-optimal solutions could also be a subset of the optimal ones. Since for practical use, we usually don't need all the optimal solutions with high cost, the sub-optimal solutions are acceptable for real-life use. To illustrate our algorithms, the illustrative example mentioned above has also been used to compare the performance of the heuristic algorithms and the main one.

Another algorithm has been developed to reconstruct the structure of the Petri net, which aims to improve the transition efficiency when a token change sequence is observed. The minimum transition number and connection between places and transitions are the objectives. This algorithm treats the length of an observation sequence as constant when we take it as input, which makes it could generate an optimized Petri net structure from given observation sequences in polynomial time. Then, we provided an example to evaluate the performance of this algorithm. After the analysis and verification with the Matlab code, we could see this algorithm could work properly.

The proposed algorithm has also been applied to a practical traffic scenario, which consisted of a certain number of stations that could let vehicles stay. These stations are represented with places in the Petri net and vehicles are represented with the tokens. The goal is to simplify the structure of the Petri net, i.e., the vehicle transiting network, based on the observed token change sequences in the places. Because of the practicality of the vehicle moving, several constraints of this application are emphasized to make the method realistic. In the illustrative example, a traffic scenario with four stations has been evaluated, with three optimized results selected from ninety-three candidates. The corresponding marking sequences and transition firing sequences have also been provided.

The first algorithm has been applied to the second traffic modeling scheme that is mentioned above, which represents the traffic network around the metropolitan area of Indianapolis with a traditional Petri net structure. In the built Petri net, different labels could be assigned to the transitions depending on the *DOI* types that the transitions represent. The labeling function shows the sets of different *DOI*s and the mapping relationship between the transitions and the *DOI* sets and the observed label sequence denotes the types and order of the visited *DOI*s during the path from the starting point to the destination. However, in the realistic situations, the order of visiting different *DOI*s normally cannot be fixed. We can only limit the number of each label that appears in the observed sequence. Therefore, the observed label sequence in the original algorithm has been turned into an observed label set, which contain a fixed number of each label, and could generate multiple label sequences with different permutations. The way to find all the combinations has been provided in the corresponding chapter. By applying the second algorithm to each one of these label sequences, it is possible to obtain an optimal route for the observed label set that can visit all the required *DOI*s and reach the final destination with the least time cost.

In the next chapter, the traditional Petri net structure has been used to model another type of traffic network, with further traffic control work being done. The traffic network contains road segments, which are connected with intersections (with monitoring devices and more than two directions), junctions (with monitoring devices and two directions), and small connections (without monitoring devices). In the Petri net structure that is used to model this traffic network, the places are used to represent the road segments. The observable connections, including intersections and junctions, are represented with observable transitions, while the unobservable connections are represented with unobservable transitions. The objective is to control the network and limit the traffic volume within a certain level following the given observed label sequence. Each of the places has been assigned a number representing the maximum tokens that it can contain. The controller proposed in this chapter is a binary vector with the length of the number of transitions, which enable and disable the transitions to limit the movement of the tokens, to make sure that the number of tokens in each place will not exceed its capacity limit. Meanwhile, because of the existence of the unobservable transitions, more possibilities of the firing sequences have been considered. Before each observed label, all enabled unobservable transitions could also generate many reachable markings that could be potential legal. Moreover, the unobservable transitions have also affected the judgment criteria of the legality for each reachable marking. For each reachable marking, it is necessary to check one more step of firing, if it enables any unobservable transitions. If the marking generated from the unobservable transition firing is illegal, its parent marking is also illegal. At the end of the chapter, a simulation has been done with the results, which shows the feasibility of the proposed algorithm.

To summarize, this dissertation focuses on traffic modeling, state and structure estimation, optimization with the Petri net structure based on external observations of the token changes or labels, and the traffic volume control within the traffic network modeled by labeled Petri net based on external observation. Three algorithms have been proposed for state/structure optimization or token movement control, which have been applied to the Petri nets that model different traffic networks with two modeling schemes for different purposes. These algorithms and modeling schemes are proved to be reliable and feasible, which could be applied to many more traffic scenarios in realistic situations.

There could be extensions in many possible directions. For the traffic modeling part, the probabilistic model can be used to investigate path planning algorithms for collision avoidance on the highway. It is also interesting to explore more traffic datasets to make the model more realistic. For the Petri net structure reconstruction part, the external observation could be expanded to token change sequences with infinite length. The algorithm needs to be updated to reduce complexity. For the minimum initial marking estimation part, other special structures of the unobservable subnet could be considered. For that purpose, the way to reduce the complexity also needs to be developed. When applying the algorithm to the traffic network application, the traffic network could be further distributed and more local roads could be included. A better data source could be used for higher accuracy. Also, because we are applying many external data sources to our research, the reliability of these data is also an aspect that must be considered. Therefore, the fault tolerance for noisy data processing and data robustness is also a potential research direction in the future. For the traffic volume controlling part, a more complex controller could be developed instead of a binary vector, in order to implement more flexible controlling operations. Besides that, when modeling the traffic network, we could consider the vehicles that are entering and leaving the area-of-interest we are considering. Representing the scenario with the Petri net structure, it will be the change of the total number of tokens in the Petri net. These extensions will make the problems more challenging but also more applicable to real applications.

REFERENCES

REFERENCES

[1] J. Tan, C. Xu, L. Li, F.-Y. Wang, D. Cao, and L. Li, “Guidance control for parallel parking tasks,” *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 301-306, January 2020.

[2] X. Zhao, K. Yan, H. Mo, and L. Li, “Type-2 fuzzy control for driving state and behavioral decision of unmanned vehicle,” *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 1, pp. 178-186, January 2020.

[3] B.-L. Ye, W. Wu, K. Ruan, L. Li, T. Chen, H. Gao, and Y. Chen, “A survey of model predictive control methods for traffic signal control,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 623-640, May 2019.

[4] R. Tian, K. Ruan, L. Li, J. Le, J. Greenberg, and S. Barbat, “Standardized evaluation of camera-based driver state monitoring systems,” *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 716-732, May 2019.

[5] S. Chien, M. T. Moury, G. Widmann, W. Kosiak, L. Li, and Y. Chen, “Performance measurement of vehicle Crash Imminent Braking systems”, in Proc. *2011 IEEE International Conference on Vehicular Electronics and Safety,* pp. 107-112, Beijing, China, July 2011.

[6] A. Lopez, R. Sherony, S. Chien, L. Li, Q. Yi, and Y. Chen, “Analysis of the braking behaviour in pedestrian automatic emergency braking”, in Proc. *2015 IEEE 18th International Conference on Intelligent Transportation Systems,* pp. 1117-1122, Canary Islands, Spain, September 2015.

[7] D. Shen, Q. Yi, L. Li, S. Chien, Y. Chen, and R. Sherony, “Data Collection and Processing Methods for the Evaluation of Vehicle Road Departure Detection Systems”, in Proc. *2018 IEEE Intelligent Vehicles Symposium (IV),* pp. 1373-1378, Changshu, Suzhou, China, July 2018.

[8] S. Chien, L. Li, A. Maina Ari, A. Meadows, H. Banvait, Y. Chen, M. T. Moury, and G. Widmann, “A new scoring mechanism for vehicle crash imminent braking systems”, *IEEE Intelligent Transportation Systems Magazine*, vol. 4, no. 4, pp. 17–29, Winter 2012.

[9] S. Chien, L. Li, and Y. Chen, “A novel evaluation methodology for combined performance of warning and braking in crash imminent braking systems”, *IEEE Intelligent Transportation Systems Magazine*, vol. 5, no. 4, pp. 62–72, Winter 2013.

[10] D. Good, K. Krutilla, S. Chien, L. Li, and Y. Chen, “Preliminary benefit analysis for pedestrian crash imminent braking systems”, in Proc. *2015 IEEE International Conference on Intelligent Transportation Systems*, pp. 1123–1128, Canary Islands, Spain, September 2015.

[11] A. Lopez, R. Sherony, S. Chien, L. Li, Y. Qiang, and Y. Chen, "Analysis of the braking behaviour in pedestrian automatic emergency braking", in Proc. *2015 IEEE International Conference on Intelligent Transportation Systems*, pp. 1117–1122, Canary Islands, Spain, September 2015.

[12] Q. Yi, S. Chien, J. Brink, W. Niu, L. Li, Y. Chen, C. Chen, R. Sherony, and H. Takahashi, "Development of bicycle surrogate for bicyclist pre-collision system evaluation", in *SAE Technical Paper* 2016-01-1447, https://doi.org/10.4271/2016-01-1447, 2016.

[13] A. Lopez, S. Chien, L. Li, Q. Yi, Y. Chen, and R. Sherony, "Certainty and critical speed for decision making in tests of pedestrian automatic emergency braking systems", *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 6, pp. 1358–1370, June 2017.

[14] Q. Yi, S. Chien, J. Brink, Y. Chen, L. Li, D. Good, C.-C. Chen, and R. Sherony, "Mannequin development for pedestrian pre-collision system evaluation", in Proc. *17th IEEE International Conference on Intelligent Transportation Systems*, pp. 1626–1631, October 2014.

[15] N. Kaempchen, B. Schiele, and K. Dietmayer, "Situation assessment of an autonomous emergency brake for arbitrary vehicle-to-vehicle collision scenarios", *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 678–687, December 2009.

[16] J. Stellet, P. Vogt, J. Schumacher, W. Branz, and J. M. Zollner, "Analytical derivation of performance bounds of autonomous emergency brake systems", in Proc. *2016 IEEE Intelligent Vehicles Symposium*, pp. 220–226, Gothenburg, Sweden, June 2016.

[17] C. Chao and X. Qin, "Research of vehicle automatic emergency braking system evaluation methods", in Proc. *IET International Conference on Intelligent and Connected Vehicles*, pp. 1–9, Chongqing, China, September 2016.

[18] G. Savino, M. Pierini, M. Rizzi, and R. Frampton, "Evaluation of an autonomous braking system in real world PTW crashes", *Traffic Injury Prevention*, vol. 14, no. 5, pp. 532–543, 2013.

[19] J. Kim, W. Jung, S. Kwon, and Y. Kim, "Performance test of autonomous emergency braking system based on commercial radar", in Proc. *2016 IIAI International Congress on Advanced Applied Informatics*, pp. 1211–1212, Kumamoto, Japan, July 2016.

[20] H. Kopetz and S. Poledna, "Autonomous emergency braking: A system-of-systems perspective", in Proc. *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop*, pp. 1–7, Budapest, Hungary, June 2013.

[21] F. Jomrich, J. Schmid, S. Knapp, A. Höß, R. Steinmetz, and B. Schuller, "Analysing communication requirements for crowd sourced backend generation of HD Maps used in automated driving," in Proc. *2018 IEEE Vehicular Networking Conference*, pp. 1-8, Taipei, Taiwan, 2018.'

[22] H. Chu, L. Guo, B. Gao, H. Chen, N. Bian, and J. Zhou, "Predictive cruise control using high-definition map and real vehicle implementation," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 11377-11389, December. 2018.

[23] M. O. Tas, H. S. Yavuz, and A. Yazici, ”Updating HD-maps for autonomous transfer vehicles in smart factories,” in Proc. *2018 IEEE International Conference on Control Engineering & Information Technology*, pp. 1-5, Istanbul, Turkey, 2018.

[24] A. Yaemjaem, N. Sutthisangiam, A. Sompan, N. Sa-ngiam, and P. Jindasee, ”Development of high-definition 3D mapping system for water resources management,” in Proc. *2018 IEEE International Conference on Advanced Informatics: Concept Theory and Applications*, pp. 219-223, Krabi, Thailand, 2018.

[25] A. Borkar, M. Hayes, and M. T. Smith, ”An efficient method to generate ground truth for evaluating lane detection systems,” in Proc. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1090-1093, Dallas, TX, 2010.

[26] A. Borkar, M. Hayes, and M. T. Smith, ”A novel lane detection system with efficient ground truth generation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 365-374, March 2012.

[27] R. Marc, G. Dominique, and P. Evangeline, ”Generator of road marking textures and associated ground truth applied to the evaluation of road marking detection,” in *Proc. 2012 IEEE International Conference on Intelligent Transportation Systems*, pp. 933-938, Anchorage, AK, USA, 2012.

[28] V. John, A. Boyali, H. Tehrani, K. Ishimaru, M. Konishi, Z. Liu, and S. Mita, “Estimation of steering angle and collision avoidance for automated driving using deep mixture of experts,” *IEEE Transactions on Intelligent Vehicles,* vol. 3, no. 4, pp. 571–584, December 2018.

[29] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, “Collision avoidance and stabilization for autonomous vehicles in emergency scenarios,” *IEEE Transactions on Control Systems Technology,* vol. 25, no. 4, pp. 1204–1216, July 2017.

[30] I. Shim, J. Choi, S. Shin, T. Oh, U. Lee, B. Ahn, D. Choi, D. H. Shim and I. Kweon, “An Autonomous Driving System for Unknown Environments Using a Unified Map,” *IEEE Transactions on Intelligent Transportation Systems,* vol. 16, issue. 4, pp. 1999–2013, Aug 2015.

[31] M. M. Mekker, S. M. Remias, M. L. McNamara, and D. M. Bullock, “Characterizing interstate crash rates based on traffic congestion using probe vehicle data”, in *Transportation Research Board 95th Annual Meeting*, Washington DC, United States, January 2016.

[32] C. M. J. Tampere, S. P. Hoogendoorn, and B. van Arem, “Continuous traffic flow modeling of driver support systems in multiclass traffic with intervehicle communication and drivers in the loops”, *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, no. 4, pp. 649-657, December 2009.

[33] C. Nowakowski, D. Vizzini, S. Gupta, and R. Sengupta, “Evaluation of real-time freeway end-of-queue alerting system to promote driver situational awareness”, *Transportation Research Record*, no. 2324, pp. 37-43, 2012.

[34] R. Browne, and A. Byrne, “Highway 402 Queue Warning System (Wireless Long Haul Application)”, in *Annual Conference of the Transportation Association of Canada*, Toronto, Ontario, Canada, 2008.

[35] A. M. Khan, “Intelligent Infrastructure-Based Queue-End Warning System For Avoiding Rear Impacts”, *IET Intelligent Transport Systems*, vol. 1, issue. 2, pp. 138 - 143, June 2007.

[36] Y. Liu, W. Zhang, Z. Wang, and C. Chan, “DSRC-based end of queue warning system”, *IEEE Intelligent Vehicles Symposium*, Los Angeles, CA, USA, 11-14 June 2017.

[37] Y. Jiang, “Estimation of Traffic Delays and Vehicle Queues at Freeway Work Zones”, in *Transportation Research Board 80-th Annual Meeting*, Washington DC, United States, January 7-11, 2001.

[38] Z. Amini, R. Pedarsani, A. Skabardonis and P. Varaiya, “Queue-Length Estimation Using Real-Time Traffic Data”, in *International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, Brazil, Nov. 1-4. 2016.

[39] G. L. Ullman, M. A. Brewer, J. E. Bryden, M. O. Corkran, C.W. Hubbs, A. K. Chandra, and K. L. Jeannotte, “Enforcement Effectiveness for Queue-End Protection”, in *Traffic Law Enforcement in Work Zones: Phase II Research, Chapter 3*, pp. 35-57, 2013.

[40] C. G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems,* New York: Springer US, 2008, pp. 223–267.

[41] T. Murata, “Petri nets: Properties, analysis and applications,” *Proceeding of the IEEE,* vol. 77, no. 4, pp. 541–580, April 1989.

[42] K. Ruan, W. Wu and L. Li, ”Reconstruction of unknown Petri net structures from asynchronous observations of token change sequences,” in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Fort Worth, TX, 2016, pp. 1049-1054.

[43] K. Ruan, L. Li and W. Wu, ”Minimum Initial Marking Estimation in Labeled Petri Nets With Unobservable Transitions,” *IEEE Access*, vol. 7, pp. 19232-19237, 2019.

[44] C. Ciufudean and C. Buzduga, ”Petri net model for energy sparing in railway traffic,” in *2016 IEEE International Power Electronics and Motion Control Conference (PEMC)*, Varna, 2016, pp. 839-842.

[45] L. Qi, M. Zhou and W. Luan, ”Modeling and control of urban road intersections with incidents via timed Petri nets,” in *2015 IEEE 12th International Conference on Networking, Sensing and Control*, Taipei, 2015, pp. 185-190.

[46] X. Zhu, ”A Petri-net modeling approach for airport apron traffic dynamics,” in *2017 36th Chinese Control Conference (CCC)*, Dalian, 2017, pp. 2332-2337.

[47] U. Bhanja, ”Urban Traffic Flow Optimization using Intelligent Techniques,” in *2018 IEEE 13th International Conference on Industrial and Information Systems (ICIIS)*, Rupnagar, India, 2018, pp. 1-6.

[48] A. Di Febbraro, D. Giglio and N. Sacco, ”A Deterministic and Stochastic Petri Net Model for Traffic-Responsive Signaling Control in Urban Areas,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 510-524, Feb. 2016.

[49] R. Aziz, M. Kedia, S. Dan, S. Sarkar, S. Mitra and P. Mitra, “Segmenting Highway Network Based on Speed Profiles ,” in *IEEE 18th International Conference on Intelligent Transportation Systems,* Las Palmas, Spain, September. 15-18 2015.

[50] Y. Guan, S. E. Li, J. Duan, W. Wang, and B. Cheng. ”Markov probabilistic decision making of self-driving cars in highway with random traffic flow: A simulation study.” *Journal of Intelligent and Connected Vehicles*, vol. 1, no. 2, pp. 77–84, August 2018.

[51] S. Contreras, P. Kachroo, and S. Agarwal, “Observability and sensor placement problem on highway segments: A traffic dynamics-based approach,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 848–858, March 2016.

[52] J. Brown, M. Romero, and A. P. Tarko. ”Discretization of Road Networks for Safety Evaluation with Consideration of Intersection Impact Zones.” *Transportation Research Record*, 2280.1 (2012): 135-144.

[53] G. Gentile, K. Nöekel, and L Meschini. ”Time and space discretization in dynamic traffic assignment models.” in Proc. of *the First International Symposium on Dynamic Traffic Assignment–DTA*, 2006. 2006.

[54] T. Nishi and R. Maeno, ”Petri Net Decomposition Approach to Optimization of Route Planning Problems for AGV Systems,” *IEEE Transactions on Automation Science and Engineering*, vol. 7, no. 3, pp. 523-537, July 2010.

[55] C. Tolba, D. Lefebvre, P. Thomas and A. El Moudni, ”Continuous Petri nets models for the analysis of traffic urban networks,” in *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, Tucson, AZ, USA, 2001, pp. 1323-1328 vol.2.

[56] Li-Guo Zhang, Zhen-Long Li and Yang-Zhou Chen, ”Hybrid petri net modeling of traffic flow and signal control,” in *2008 International Conference on Machine Learning and Cybernetics*, Kunming, 2008, pp. 2304-2308.

[57] B. Huang, C. Zhao and Y. Sun, ”Modeling of Urban Traffic Systems Based on Fluid Stochastic Petri Nets,” in *2008 Fourth International Conference on Natural Computation*, Jinan, 2008, pp. 149-153.

[58] X. Lu, M. Zhou, A. C. Ammari, and J. Ji, “Hybrid Petri nets for modeling and analysis of microgrid systems,” *IEEE/CAA Journal of Automatica Sinica*, vol. 3, no. 4, pp. 349–356, October 2016.

[59] L. Qi, M. Zhou, and W. Luan, “Emergency traffic-light control system design for intersections subject to accidents,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 1, pp. 170–183, January 2016.

[60] N. Ran, H. Su, and S. Wang, “An improved approach to test diagnosability of bounded Petri nets,” *IEEE/CAA Journal of Automatica Sinica*, vol. 4, no. 2, pp. 297–303, April 2017.

[61] F. Yang, N. Wu, Y. Qiao, and R. Su, “Polynomial approach to optimal one-wafer cyclic scheduling of treelike hybrid multi-cluster tools via Petri nets,” *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 270–280, January 2018.

[62] B. Huang, M. Zhou, Y. Huang, and Y. Yang, “Supervisor synthesis for FMS based on critical activity places,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, to appear, DOI: 10.1109/TSMC.2017.2732442, Volume: 49, Issue: 5 , May 2019.

[63] M. P. Cabasino, A. Giua, and C. Seatzu, “Identification of Petri nets from knowledge of their languages,” *Discrete Event Dynamic Systems*, vol. 17, no. 4, pp. 447–474, December 2007.

[64] M. P. Cabasino, A. Giua, and C. Seatzu, “Linear programming techniques for the identification of place/transition nets,” in Proc. *IEEE International Conference on Decision and Control*, pp. 514–520, Cancun, Mexico, December 2008.

[65] M. Dotoli, M. P. Fanti, and A. M. Mangini, “Real time identification of discrete event systems using Petri nets,” *Automatica*, vol. 44, no. 5, pp. 1209–1219, May 2008.

[66] M. Dotoli, M. P. Fanti, A. M. Mangini, and W. Ukovich, “Identification of the unobservable behaviour of industrial automation systems by Petri nets,” in *Control Engineering Practice*, vol. 19, no. 9, pp. 958–966, September 2011.

[67] M. P. Cabasino, A. Giua, C. N. Hadjicostis, and C. Seatzu, “Fault model identification and synthesis in Petri nets,” *Discrete Event Dynamic Systems*, vol. 25, no. 3, pp. 419–440, September 2015.

[68] A. P. Estrada-Vargas, E. Lopez-Mellado, and J.-J. Lesage, “A blackbox identification method for automated discrete-event systems,” *IEEE Trans. Antomation Science and Engineering*, vol. 14, no. 3, pp. 1321-1336, July 2017, doi: 10.1109/TASE.2015.2445332.

[69] L. Li and C. N. Hadjicostis, “Reconstruction of transition firing sequences based on asynchronous observations of place token changes,” in Proc. *46th IEEE Conference on Decision and Control*, pp. 1898–1903, New Orleans, LA, December 2007.

[70] J. Yan, L. Li, and D. S. Kim, “Reconstruction of event sequences based on asynchronous observations in sensor networks,” in Proc. *ACM Intl. Conference on Ubiquitous Information Management and Communication*, Kota Kinabalu, Malaysia, January 2013.

[71] M. Qin, Z. Li, M. Zhou, M. Khalgui and O. Mosbahi, “Deadlock prevention for a class of Petri nets with uncontrollable and unobservable transitions,” *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, vol. 42, issue. 4, pp. 727–738, May 2012.

[72] P. Bonhomme, “Marking estimation of P-time Petri nets with unobservable transitions,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 3, pp. 508–518, March 2015.

[73] F. Arichi, B. Cherki, and M. Djemai, ”State and firing sequence estimation of Petri net application to manufacturing systems,” in Proc. *IEEE 2013 International Conference on Control, Decision, and Information Technologies*, pp. 608–613, Hammamet, Tunisia, December 2013.

[74] A. Giua, “Petri net state estimators based on event observation,” in Proc. *36th Intl. Conference on Decision and Control*, pp. 4086–4091, San Diego, CA, USA, December 1997.

[75] A. Giua and C. Seatzu, “Observability of place/transition nets,” *IEEE Transactions on Automatic Control*, vol. 47, no. 9, pp. 1424–1437, Sepetember 2002.

[76] D. Corona, A. Giua, and C. Seatzu, “Marking estimation of Petri nets with silent transitions,” in Proc. *43rd IEEE Intl. Conf. Decision and Control*, pp. 966–971, Atlantis, Paradise Island, Bahamas, December 2004.

[77] D. Corona, A. Giua, and C. Seatzu, “Marking estimation of Petri nets with silent transitions,” *IEEE Transactions on Automatic Control*, vol. 52, no. 9, pp. 1695–1699, September 2007.

[78] M. P. Cabasino, C. N. Hadjicostis, and C. Seatzu, “Initial marking estimation in labeled Petri nets in a probabilistic setting,” in Proc. *53rd IEEE Intl. Conference on Decision and Control*, pp. 6725–6730, Los Angeles, CA, USA, December 2014.

[79] M. P. Cabasino, C. N. Hadjicostis, and C. Seatzu, “Probabilistic marking estimation in labeled Petri nets,” *IEEE Transactions on Automatic Control*, vol. 60, no. 2, pp. 528–533, February 2015.

[80] D. Kiritsis, K. P. Neuendorf, and P. Xirouchakis, “Petri net techniques for process planning cost estimation,” *Advances in Engineering Software*, vol. 30, no. 6, pp. 375–387, June 1999.

[81] M. Yamauchi and T. Watanabe, “A heuristic algorithm for the minimum initial marking problem of Petri nets,” in Proc. *1997 IEEE Intl. Conference on Systems, Man, and Cybernetics*, pp. 245–250, 1997.

[82] Y. Ru and C. N. Hadjicostis, “State estimation in discrete event systems modeled by labeled Petri nets,” in Proc. *45th IEEE Intl. Conference on Decision and Control*, pp. 6022–6027, San Diego, CA, USA, December 2006.

[83] L. Li and C. N. Hadjicostis, “Minimum initial marking estimation in labeled Petri nets,” in Proc. *2009 American Control Conference*, pp. 5000–5005, St. Louis, MO, USA, June 2009.

[84] L. Li and C. N. Hadjicostis, “Minimum initial marking estimation in labeled Petri nets,” *IEEE Transactions on Automatic Control*, vol. 58, no.1, pp. 198–203, January 2013.

[85] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho and M. A. Perillo, ”Middleware to support sensor network applications,” *IEEE Network*, vol. 18, no. 1, pp. 6-14, January.-February. 2004.

[86] H. Jameel, U. Kalim, A. Sajjad, S. Lee and T. Jeon, ”Mobile-to-Grid Middleware: Bridging the Gap Between Mobile and Grid Environments”, *Lecture Notes in Computer Science*, 3470. 932-941. 10.1007/11508380_95, 2005.

[87] F. Arichi, H. Kebabti, B. Cherki and M. Djemai, ”Failure components detection in discrete event systems modeled by Petri net,” in *3rd International Conference on Systems and Control*, Algiers, 2013, pp. 224-229.

[88] D. Li, F. Xia, J. Luo and A. Yakovlev, ”Modelling Reversion Loss and Shoot-through Current in Switched-Capacitor DC-DC Converters with Petri Nets,” in *2019 29th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, Rhodes, Greece, 2019, pp. 69-74.

[89] F. Čapkovič, ”Modelling and Control of Complex Flexible Manufacturing Systems by Means of Petri Nets,” in *2018 IEEE 18th International Symposium on Computational Intelligence and Informatics (CINTI)*, Budapest, Hungary, 2018, pp. 000125-000130.

[90] K. Farah, K. Chabir and M. N. Abdelkrim, ”Colored Petri nets for modeling of networked control systems,” in *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, Sousse, Tunisia, 2019, pp. 226-230.

[91] Z. Ding, Y. Zhou, M. Jiang and M. Zhou, ”A New Class of Petri Nets for Modeling and Property Verification of Switched Stochastic Systems,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 45, no. 7, pp. 1087-1100, July 2015.

[92] L. Li and C. N. Hadjicostis, “Least-cost transition firing sequence estimation in labeled Petri nets with unobservable transitions,” *IEEE Transactions on Automation Science and Engineering*, vol. 8, no. 2, pp. 394–403, April 2011.

[93] Y. Emzivat, B. Delahaye, D. Lime, and O.H. Roux, ”Probabilistic time Petri nets”, *Kordon F., Moldt D. (eds) Application and Theory of Petri Nets and Concurrency. PETRI NETS 2016. Lecture Notes in Computer Science*, vol 9698. Springer.

[94] J.-M. Proth and X. Xie, ”Petri Nets: A Tool for Design and Management of Manufacturing Systems”, *New York: Wiley*, 1996.

[95] U.S. Department of Transportation’s (USDOT) Intelligent Transportation Systems (ITS) Joint Program Office (JPO), ”Active Transportation Demand Management (ATDM) Trajectory Level Validation”, https://data.transportation.gov/Automobiles/Active-Transportation-Demand-Management-ATDM-Traje/25r8-p3cy, November 2018.

[96] K, G, Arthur and K. Theresa M, ”Appendix B: B9. Plane and Spherical Trigonometry: Formulas Expressed in Terms of the Haversine Function”, *Mathematical handbook for scientists and engineers: Definitions, theorems, and formulas for reference and review (3 ed.)*, Mineola, New York, USA: Dover Publications, Inc. pp. 892—893, 2000.

[97] K. Ruan, Z. Yarmand, R. Tian, L. Li, Y. Chen, F. Li, and J. Sturdevant, ”Highway End-of-Queue Alerting System Based on Probe Vehicle Data”, in *Duffy V. (eds) Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management, Human Body and Motion. HCII 2019. Lecture Notes in Computer Science*, vol 11581. Springer, Cham

[98] Traffic Slowdown Database from Indiana Department of Transportation (INDoT), http://content.trafficwise.org/json/deltas.json. Last Accessed: May. 30, 2020.

[99] Google Map Platform, Web Services, Roads API, https://developers.google.com/maps/documentation/roads/speed-limits. Last Accessed: May. 30, 2020.

[100] E. W. Dijkstra, ”A note on two problems in connexion with graphs.” *Numerische mathematik* 1.1 (1959): 269-271.

[101] E. Aarts and J. K. Lenstra, ”Local Search in Combinatorial Optimization”, *John Wiley and Sons*, New York, NY, USA, 1997.

[102] J. Luo, Y. Wan, W. Wu and Z. Li, ”Optimal Petri-Net Controller for Avoiding Collisions in a Class of Automated Guided Vehicle Systems,” *IEEE Transactions on Intelligent Transportation Systems*, 30 August 2019, 10.1109/TITS.2019.2937058.

[103] Y. Ru and C. N. Hadjicostis, ”Bounds on the Number of Markings Consistent With Label Observations in Petri Nets,” *IEEE Transactions on Automation Science and Engineering*, vol. 6, no. 2, pp. 334-344, April 2009.

VITA

VITA

Keyu Ruan is currently enrolled as a Ph.D. student at Purdue University West Lafayette and he specializes in Electrical and Computer Engineering. Before that, he received the degree of M.S. in Electrical and Computer Engineering from Indiana University–Purdue University Indianapolis in 2015 and received the degree of B.S. in Electrical Engineering from Beijing Jiaotong University in 2010.

In the years of pursuing the M.S. and Ph.D. degrees, he has also been working in the research group called the Transportation Active Safety Institute (TASI) as a research assistant. He has been involving in many different academic and industrial projects related to autonomous vehicles and intelligent transportation during these years. Benefit from these experiences, he has the opportunities for the internships offered by vehicle OEMs like Ford and Aptiv.

Because of his education and project experiences, Keyu Ruan's research interests include control theory, discrete event systems, system modeling, active/passive safety systems, autonomous vehicles, and intelligent traffic/transportation.

Keyu Ruan's publications have been listed below:

[1] K. Ruan, W. Wu and L. Li, ”Reconstruction of unknown Petri net structures from asynchronous observations of token change sequences,” in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, Fort Worth, TX, 2016, pp. 1049-1054, doi: 10.1109/COASE.2016.7743519.

[2] K. Ruan, L. Li and W. Wu, ”Minimum Initial Marking Estimation in Labeled Petri Nets With Unobservable Transitions,” in *IEEE Access*, vol. 7, pp. 19232-19237, 2019, doi: 10.1109/ACCESS.2019.2894352.

[3] K. Ruan et al., ”A Novel Scoring Method for Pedestrian Automatic Emergency Braking Systems,” in *2019 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, Zhengzhou, China, 2019, pp. 128-133, doi: 10.1109/SOLI48380.2019.8955051.

[4] K. Ruan et al. ”Highway End-of-Queue Alerting System Based on Probe Vehicle Data,” in *Duffy V. (eds) Digital Human Modeling and Applications in Health, Safety, Ergonomics and Risk Management. Human Body and Motion. HCII 2019. Lecture Notes in Computer Science*, vol 11581. Springer, Cham.

[5] K. Ruan et al., ”A Novel Method for Ground-truth Determination of Lane Information through a Single Web Camera,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*, Las Vegas, NV, United States, 2020.

[6] K. Ruan et al., ”Highway Traffic Modeling Using Probabilistic Petri Net Models,” in *2020 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, Virtual Conference, 2020.

[7] R. Tian, K. Ruan, L. Li, J. Le, J. Greenberg and S. Barbat, ”Standardized evaluation of camera-based driver state monitoring systems,” in *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 716-732, May 2019, doi: 10.1109/JAS.2019.1911483.

[8] R. Tian, K. Ruan, L. Li, J. Le et al., ”Towards Standardized Performance Evaluation of Camera-Based Driver State Sensing Technologies,” in *SAE Technical Paper*, 2016-01-1500, 2016, https://doi.org/10.4271/2016-01-1500.

[9] H. Wei, L. Chen, K. Ruan, L. Li and L. Chen, ”Low-Rank Tensor Regularized Fuzzy Clustering for Multiview Data,” in *IEEE Transactions on Fuzzy Systems*, doi: 10.1109/TFUZZ.2020.2988841.

[10] B. Ye, W. Wu, K. Ruan et al., ”A survey of model predictive control methods for traffic signal control,” in *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 3, pp. 623-640, May 2019, doi: 10.1109/JAS.2019.1911471.

[11] B. Ye, H. Gao, L. Li, K. Ruan, W. Wu and T. Chen, ”A MILP-based MPC Method for Traffic Signal Control of Urban Road Networks *,” in *2019 Chinese Automation Congress (CAC)*, Hangzhou, China, 2019, pp. 3820-3825, doi: 10.1109/CAC48633.2019.8997474.

[12] R. Sherony, Q. Yi, S. Chien, J. Brink et al., ”Development of Bicycle Carrier for Bicyclist Pre-Collision System Evaluation,” in *SAE Technical Paper*, 2016-01-1446, 2016, https://doi.org/10.4271/2016-01-1446.