Hindawi
Security and Communication Networks
Volume 2022, Article ID 3288308, 18 pages
https://doi.org/10.1155/2022/3288308

Research Article

Formal Model and Analysis for the Random Event in the Intelligent Car with Stochastic Petri Nets and *Z*

Yang Liu, Yingqi Fan, Darong Huang, Bo Mi, and Liyuan Huang

Information Science and Engineering, Chongqing Jiaotong University, Chongqing 400074, China

Correspondence should be addressed to Yingqi Fan; fanyingqi@mails.cqjtu.edu.cn

Received 24 June 2022; Revised 15 September 2022; Accepted 20 September 2022; Published 13 October 2022

Academic Editor: Chen Chen

With the continuous development of science and technology, people's lifestyle becomes more and more intelligent, especially in intelligent transportation. However, running in a random environment, safety can be affected by various factors during operation. For an intelligent car, guaranteeing its safety in operation is important to the passengers in the vehicle. So, it is vital to verify the safety of the system of the smart car. This study proposes an integration formal method with stochastic Petri nets and Z (SPZN). Stochastic Petri nets can better simulate the occurrence of random events in the driving process of intelligent cars. With the advantages of the frame structure of Z language, the concurrent process and state before and after the system at different times can be better described. In addition, the frame structure of Z language can solve the problem of state explosion in Petri nets. Using this method, the random events that may occur during the operation can be formally modeled, and the subsequent behavior of the vehicle can be analyzed and predicted effectively. Using the reinforcement learning, the parameter λ in the stochastic Petri nets can be optimized, which can reduce the probability of bad states and ensure the stability and security of the system. Moreover, a case study of the intelligent car modeled by stochastic Petri nets and Z is given. The results show that it can improve the safety and effectiveness of the smart vehicle driving system.

1. Introduction

With the rise of the Internet of vehicle technology [1–5], smart cars are gradually integrated into people's lives. However, smart cars always operate in an environment full of random factors, such as pedestrians, traffic signs on the road, other driving vehicles, and changeable weather, and its safety is particularly important. Liu et al. [6] proposed an efficient communication method FedCPF. Compared with traditional federated learning, this method achieves more efficient communication, faster convergence, and higher test accuracy. In terms of communication, the security of the Internet of vehicles system is improved. Zhao et al. [7] proposed a digital twin-assisted storage strategy for satellite-terrestrial networks (INTERLINK), which leverages the digital twins (DTs) to map the satellite networks to virtual space for better communication. By enabling more reliable communication, the safety of smart cars is guaranteed. Soni et al. [8] developed a novel low-cost sensor system that improves safety in intelligent transportation systems by improving vehicle sensor systems. Martinez et al. [9] found that driving style plays an important role in driving safety and then used different algorithms to characterize and identify the driver's driving style. Through the analysis of the driver's driving style, the safety performance during the driving process can be improved. Colombo et al. [10] proposed a strategy to dynamically decompose the formal verification problem of a large road network, which can effectively prevent vehicle collisions and improve the safety of autonomous vehicles. To sum up, the researchers have improved the security of the Internet of vehicles from different aspects, but the above methods have not effectively solved the random events that occur in the driving process of smart cars.

The formal method is an effective method to prove system security, accessibility, and effectiveness, and it is not only widely used in computer hardware technology [11], software requirement verification [12–14], industrial

engineering [15], medicine [16], communication [17], and so on. In recent years, it has been increasingly used in the field of transportation. Qi et al. [18–20] used Petri nets to model intersections and designed strategies to solve the congestion problem. In addition, they also used time-delay Petri nets (TPNs) for intersection modeling, designed corresponding strategies to solve the accident-induced congestion problem, and proposed a method to classify driving behavior at intersections in congestion. Their work effectively improves the congestion at intersections and enhances the management and safety of urban traffic. In the study by Labadi et al. [21], stochastic Petri net (SPN) was used to model and analyze public bike sharing systems, and the results showed that SPN is very suitable for analyzing and simulating discrete-time systems. Moreover, there exist some researchers who have applied formal modeling to Internet of vehicles (IoVs). Liu et al. [22] proposed a formal model based on integration time Petri nets and Z (TPZN). By applying TPZN to IoVs, the behavior of IoVs can be accurately and formally described, and the model is validated with a case study, and the results show that the proposed approach can effectively improve the safety and intelligence of IoVs.

Although the above work has improved the security and intelligence of IoV, it still suffers from the problem of insufficient ability to describe random events. SPN has the ability to describe random events, but it still exists the problem of state explosion. To solve this problem, this study proposes a formal modeling approach combining stochastic Petri nets and Z (SPZN). SPZN consists of two parts, SPZN-SPN and SPZN-Z. SPZN-SPN defines the structure and flow of the overall model, and SPZN-Z abstracts the structure and describes the related constraints. Through the abstraction of SPZN-Z, the complexity of SPN can be effectively reduced, the number of states can be reduced, and the state explosion can be avoided. In addition, each transition in SPN has a transition implementation rate λ. λ can affect the stability and security of the system, so setting the value of λ so that the system has high stability and high security is an urgent problem to be solved. However, setting λ in practical problems is not an easy task, and Song and Sun et al. [23, 24] used assumed λ in their work and did not give a method to determine the values of λ. Therefore, in this study, we propose an optimization method for the transition implementation rate λ. We use the actor-critic algorithm in reinforcement learning to optimize λ for SPZN-SPN, which improves the stability and security of the system.

The rest of this study is organized as follows. Section 2 presents some of the basics required for this study. Section 3 introduces a formal modeling approach based on SPZN, refines the method, and proposes a transition implementation rate optimization method. Section 4 analyzes the model in terms of reachability, boundedness, and safety, respectively, and lists the advantages of the model. An example of a SPZN-based smart connected car system with random events is given in Section 5 to verify the effectiveness of the method in this study. Section 6 concludes the study and discusses future work.

2. Preliminaries

In this section, we review smart net cars, stochastic Petri nets, the relationship between stochastic Petri nets and Markov chains, Z language structures, and reinforcement learning.

2.1. Intelligent Connected Vehicle. With the rapid development of computer technology and artificial intelligence, traditional traffic system is gradually being replaced by the intelligent connected vehicle system that integrates people, vehicles, roads, and clouds, that is, "smart transportation" [25, 26]. As shown in Figure 1, this figure is a conceptual diagram of the vehicle-road-human-cloud integrated system. The relationship between the intelligent connected vehicle (ICV) and its various components is shown in Figure 2. It is not difficult to see that the ICV is mainly divided into autonomous vehicles and the Internet of vehicles. For the yellow part where smart transportation and the Internet of vehicles intersect in the figure, the part to which the smart connected car belongs. There is a technical architecture diagram, as shown in Figure 3. In the infrastructure submodule, sensors play an essential role. Usually, sensors are divided into three categories: distance, speed, and image. The distance sensor is mainly radar, and the image sensor is mainly a surveillance camera.

The related research on autonomous vehicles can be summarized into three aspects: autonomous driving in high-speed environments, autonomous driving in urban environments, and autonomous driving in special environments. Autonomous driving in high-speed environments is mainly applied to highways, and the difficulty lies in the safety of vehicles and passengers under high-speed driving. A lot of research is still needed to break through this difficulty. In the urban environment, the automatic driving speed is slow, the vehicle safety performance is high, the application is more extensive, and the prospect is better. However, it still has a complex urban environment and requires a more sensitive perception control algorithm. For autonomous driving in special environments, it is mainly used in harsh environments such as military operations. In this application scenario, we should first consider the reliability of vehicles in harsh environments.

For the Internet of vehicles, Ji et al. [26] proposed a new Internet of vehicle architecture. The purpose is to promote the rapid development of intelligent connected vehicles, improve the ability of road condition perception and traffic management and control, and lay the foundation for intelligent transportation. As shown in Figure 4, the architecture mainly consists of four layers: security authentication layer, data acquisition layer, edge layer, and cloud platform layer.

2.2. Stochastic Petri Net Extension. A stochastic Petri net is a kind of advanced Petri net that includes time factors and probability [27]. The main difference between the stochastic Petri net and the time Petri net is randomness. So, a

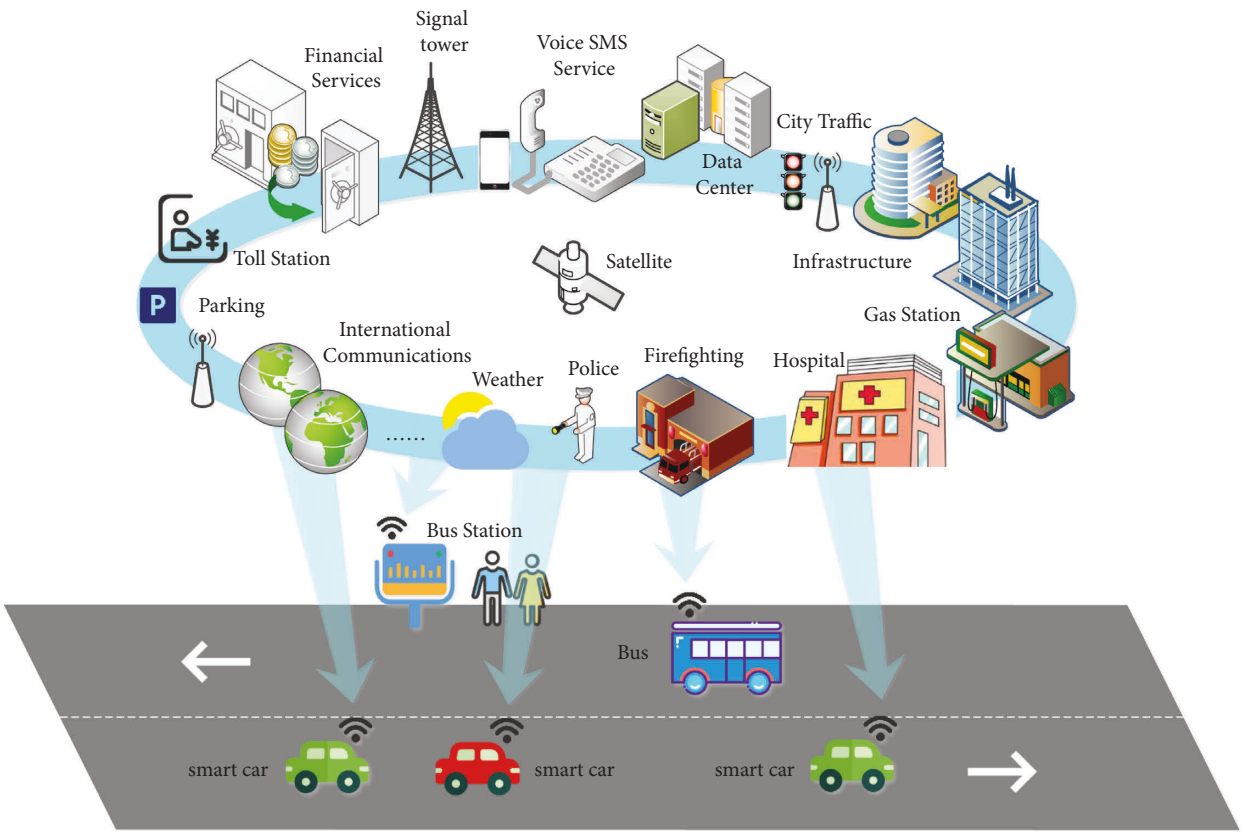
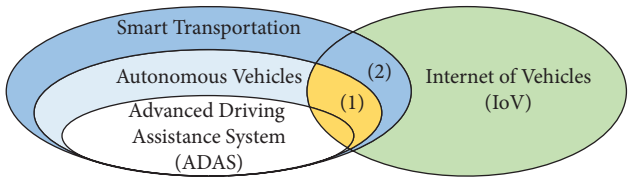


Figure 1: Conceptual diagram of the vehicle-road-human-cloud integrated system.



(1) Intelligent connected vehicle
(2) Collaborative intelligent traffic management

Figure 2: Relationship among ICV, IV, and IoV.

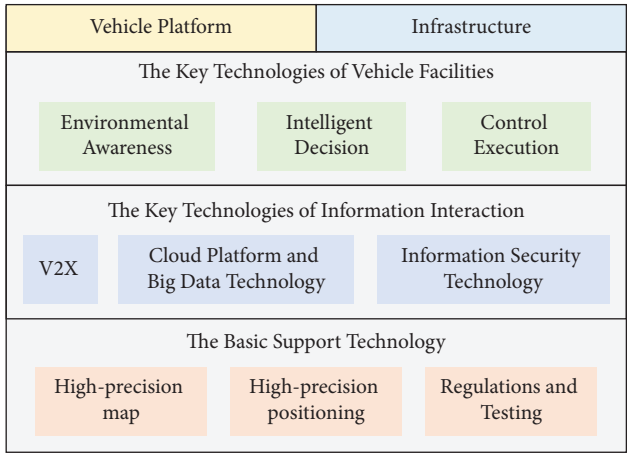


Figure 3: Framework of ICV technology.

stochastic Petri net is more suitable to describe the uncertain system. Furthermore, reachable marking graphs of stochastic Petri nets with finite places and transitions are isomorphic to one-dimensional continuous-time Markov [28, 29].

Stochastic Petri net is defined as 5-tuple $N = (P, T, F, M_0, \lambda)$:

(1) $P = \{p_1, p_2, \ldots, p_n\}$ is a finite and non-empty set of places

(2) $T = \{t_1, t_2, \ldots, t_n\}$ is a finite and non-empty set of transitions

(3) $F = \{P \times T\} \cup \{T \times P\}$ represents places to transitions and transitions to places

(4) M_0 is the initial state vector and represents the number of tokens in each place of the model

(5) λ represents the set of implementation rate of transition

Here, for $\forall t_i \in T$, $\lambda(t_i) = \lambda_i$ is a nonnegative real number, which represents the rate of occurrence when the transition t_i satisfies the occurrence conditions, and $1/\lambda_i$ represents the average firing delay or average service time.

$\forall t \in T$, $F_t(x) = 1 - e^{-\lambda_t x}$, represents the probability of t happen in x-time [30].

If there is $[M|t_j > M'$, then the transition t_j is triggered and the state is changed from M to M'. The token transfer rule is shown as follows:

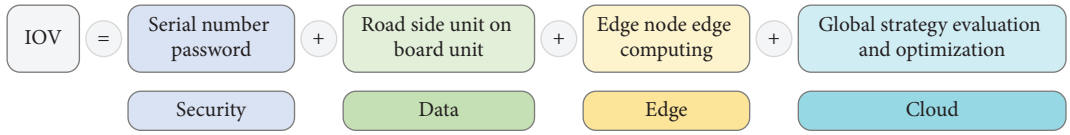


Figure 4: Main elements of the proposed architecture.

$$\forall p_i \in P, M'(p_i) = M(p_i) - \mathrm{Pre}(p_i, t_j) + \mathrm{Post}(p_i, t_j). \quad (1)$$

2.3. SPN to Markov Chain. For a stochastic Petri net, it has the rate of occurrence of transition, $\lambda_i \in \lambda$, where λ_i is a nonnegative real number; then, the time delay d_i of t_i is a random variable that is related to time and obeys exponential distribution. Because the exponential distribution has memoryless properties, if there is a stochastic Petri net, the reachable marking graph of the stochastic Petri net is isomorphic to a finite Markov chain and satisfies the following definition [30].

Definition 1. For a Markov chain that is isomorphic to a stochastic Petri net, where the Markov chain has n states, we define a transition matrix Q of $n \times n$ order. When i is not equal to j, if there is $t_k \in T$ that makes $M_i[t_k > M_j$, then we have

$$q_{ij} = \frac{d\left(1 - e^{-\lambda_k \tau}\right)}{d\tau}\Big|_{\tau=0} = \lambda_k, \quad (2)$$

else

$$q_{ij} = 0. \quad (3)$$

When i is equal to j, we have

$$q_{ij} = \frac{d\prod_k \left(1 - \left(1 - e^{-\lambda_k \tau}\right)\right)}{\mathrm{d}\tau}\Big|_{\tau=0} = \frac{d\left(e^{-\tau \sum_k \lambda_k}\right)}{\mathrm{d}\tau}\Big|_{\tau=0} = -\sum_k \lambda_k, \quad (4)$$

where λ_k is the average implementation rate of transition t_k and $\exists M' \in [M_0 >$ and $\exists t_k \in T$ that makes $M_i[t_k > M_j$.

Figure 5 is a simple case about the reachable marking graph isomorphic to the Markov chain. From the Petri net of Figure 5(a), the reachable marking graph of Figure 5(b) can be obtained, and the reachable marking graph is isomorphic to the Markov chain, so we can get the Markov chain shown in Figure 5(c). There are 5 states in the reachable marking graph, so there are also 5 states in the corresponding Markov chain.

By constructing the reachable marking graph of the stochastic Petri net, we can obtain that the reachable marking graph and its isomorphic Markov chain have n states, and the steady-state probability from state M_0 to state M_{n-1} is an n-dimensional vector P, and then, $P = (P(M_0), P(M_1) \cdots P(M_{n-1}))$, where $P(M_i)$ is the steady-state probability of marking M_i. According to the Markov process, there are the following equations. By solving this system of equations, we can obtain the steady-state probability $P(M_i)$ for each reachable state M_i.

$$\begin{cases} P \times Q = 0, \\ \sum_{i=0}^{n-1} P(M_i) = 1. \end{cases} \quad (5)$$

2.4. Z Frame Structure. Z language is a specification language mainly based on first-order predicate calculus. It is a functional language, which can easily express the state and behavior of things in mathematical symbols. As shown in Figure 6(a), in the Z language this structure is called a schema. It is the basic description unit and the basic structure of Z language. A pattern includes the name of the pattern (S), the declaration part (D), and the assertion part (P), and the assertion part is divided into pre-assertion and post-assertion. The framework is similar to a class in the C++ language, as shown in Figure 6(b). A class has a class name, attributes, and functions, which correspond to the name, declaration part, and assertion part of the schema, respectively.

In fact, Z language is just a set of prescribed mathematical symbols, and the "program" written in Z language is an abstract design of computer software or hardware system. Therefore, the content written in the Z language is not a computer program, nor is it a code that can be compiled to generate and can be run on a computer. The content written in Z language is not for the computer to run, but for human understanding and analysis. Using Z language, users can understand the modules, data types, and processes of the system so that the system can be analyzed, optimized, verified, tested, etc. In terms of data abstraction, Z language has a stronger descriptive ability than Petri net [31]. Using the Z language, Petri net state can be reduced and state explosion can be avoided.

2.5. Reinforcement Learning. In reinforcement learning, when the agent makes an action a according to the policy function $\pi(a|s)$ in the state s, the environment rewards the agent according to the action. We usually have two methods for artificial intelligence to intelligently control the agent: the first method is policy learning, and the second method is value learning.

Policy learning uses a neural network $\pi(a|s;\theta)$ to approximate the policy function $\pi(a|s)$, so the neural network is also called a policy network. The parameters θ in the policy network represent the weights, and they are updating and optimizing through the actions of the agent and the rewards given by the environment so that the policy network can achieve the desired result. It is represented by the policy gradient algorithm in policy learning, and the Monte Carlo method is the simplest among the many gradient algorithms.

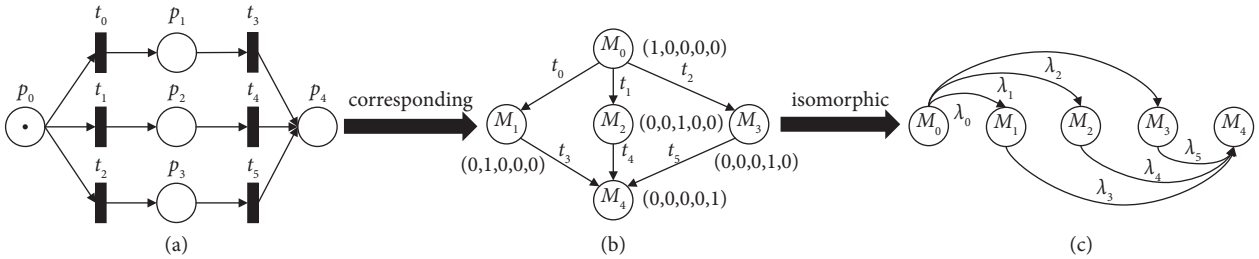


Figure 5: Reachable marking graph isomorphic Markov chain. (a) Petri net. (b) Reachable marking graph. (c) Markov chain.

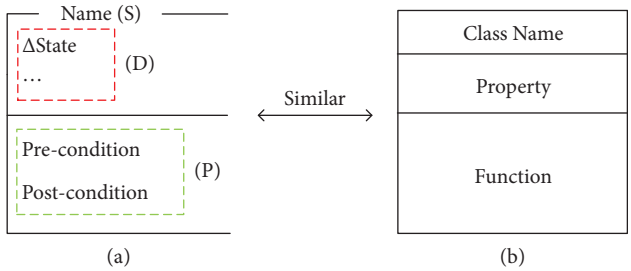


Figure 6: Structure of the Z language is similar to the structure of the C++. (a) The framework of Z language. (b) The C++ language class framework.

Value learning is also a method of approximating the value function $Q^*(s_t, a_t)$ using a neural network $Q(s, a; w)$ called a value network. s_t and a_t represent the state and action at time t, respectively, and s, a, and w represent the state, action, and weight of the value network, respectively. The value network is used to predict the score of the action in the current state, and then, the actual reward of the action is given according to the environment, and the gradient descent method is used to correct the future reward prediction. Common value learning algorithms include Q-learning algorithm and deep Q-network algorithm (DQN). Based on the Q-learning algorithm, the DQN algorithm combines the deep neural network, uses the neural network to approximate the value function, and uses the experience playback to train the entire process of reinforcement learning.

It should be noted that there are many learning methods in reinforcement learning. In this study, we use the actor-critic method for reinforcement learning. The method approximates the policy function and the value function using two neural networks, the policy network (actor) is equivalent to a gymnast, and the value network (critic) is equivalent to a gymnastic referee. In the initial state, athletes can only make random actions, and the referee can only make random evaluations based on the athletes' actions and current state. After the referee makes an action, the environment gives corresponding rewards, and the referee gradually becomes professional by catering to the environment. After the athlete makes an action, the referee evaluates it. By catering to the referee, the athlete's action gradually becomes standardized to achieve the purpose of optimizing the strategy function. Figure 7 depicts the working principle of the actor-critic method.

In the actor-critic method, a state value function $V_\pi(s) \approx \sum_a \pi(a|s) \times Q_\pi(s, a)$ is defined, where $\pi(a|s)$ is the policy function and $Q_\pi(s, a)$ is the value function, because two neural networks are used to approximate the policy function and the value function, respectively, so the policy network is $\pi(a|s; \theta)$, the value network is $q(s, a; w)$, and the state value function is as follows:

$$V_\pi(s) \approx \sum_a \pi(a|s; \theta) \times q(s, a; w). \tag{6}$$

By summing the products of the probability and value of each action, we can easily obtain the pros and cons of executing the policy function π in the current state s. The main steps of the method are shown in Figure 8. Among them, the temporal difference (TD) algorithm is an algorithm that stages the entire model training process, which is suitable for the idea of discounted returns in reinforcement learning and can be applied to value learning.

Comparing the above three reinforcement learning algorithms, their respective advantages and disadvantages are shown in Table 1 [32–34]. Although the representative algorithm DQN in value learning can store previous data through the experience pool and break the relationship between information, it can effectively solve the problem of complex state and action and the existence of a correlation between data. However, this method still has problems such as overfitting, low sample utilization, and unstable evaluation in the solution process. Compared with value learning, policy learning is simpler and has better convergence, but it still has shortcomings such as high algorithm variance, slow convergence speed, and difficulty in determining the learning step size. To further reduce the variance, the actor-critic algorithm is proposed, and the state value function is used as the baseline to predict the value of the bootstrapping method, which greatly reduces the variance, but it introduces deviation, making it a major drawback of the actor-critic algorithm.

3. Modeling with SPZN

To improve the abstraction ability and randomness ability in the intelligent networked automobile system, this study integrates the stochastic Petri net and the Z framework and proposes SPZN. Using the randomness of SPN and the abstraction ability of Z, the shortcomings of the system can be effectively improved. Compared with TPN, SPN, and PZN, SPZN can define and describe the system more efficiently.

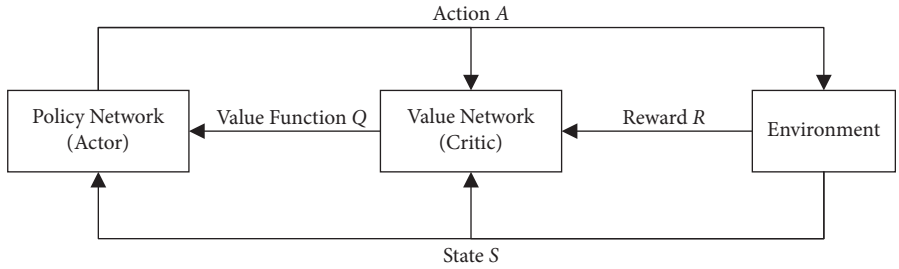


Figure 7: Principle of the actor-critic method.

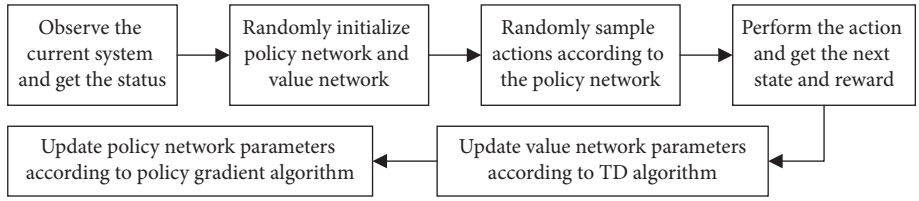


Figure 8: Main steps of the actor-critic method.

Table 1: Advantages and disadvantages of three reinforcement learning algorithms.

Category	Representative algorithm	Advantage	Disadvantage
Value learning	DQN	It can solve high-dimensional complex problems and is not easy to fall into local optimum	Overfitting, low sample utilization, instability, poor convergence
Policy learning	Monte Carlo method	High stability and strong convergence	Large variance, slow convergence, the easy local optimal solution
Value policy learning	Actor-critic	Small variance, fast training, and can solve continuous problems	Low learning efficiency, large deviation, and poor stability

3.1. SPZN

Definition 2. A SPZN is a tuple $(P, T, F, M_0, \lambda, Z_P, Z_T, S, C)$, where

(1) P is a set of the places, $P = \{p_1, p_2, \ldots, p_n\}$
(2) T is a set of the transitions, $T = \{t_1, t_2, \ldots, t_n\}$
(3) F is a set of the arcs, which links a place and a transition
(4) M_0 is the initial marking, which describes the initial state of the system
(5) λ is a set of the transition implementation rate, which is a concept with λ in SPN
(6) PN = (P, T, F, M_0) is a basic Petri net
(7) SPN = (P, T, F, M_0, λ) is a stochastic Petri net
(8) PZN = $(P, T, F, Z_P, Z_T, S, C)$ is a PZN
(9) Z_P is a set of the place based on Z
(10) Z_T is a set of the transition based on Z
(11) S: $P \longrightarrow Z_P$ is a set of the one-to-one map relationship between P and Z_P
(12) C: $T \longrightarrow Z_T$ is a set of the one-to-one map relationship between T and Z_T

To better express SPZN, the corresponding relationship between SPN and Z is shown in Figure 9. Among them, the implementation rate of transition in SPN can be used as a precondition for the occurrence of transition T, so corresponding to the pre-assertion in Z language, the implementation rate of transition can be reflected in the assertion. In Petri nets, the place is treated as an entity that does not have any action, so it has only attributes, corresponding to the declaration part in the Z language, whereas the transition is treated as an action, and the place is triggered by the transition, so the transition has properties and functions, corresponding to the declaration part and the assertion part in the Z language.

3.2. Model Refining.

The modeling process of an intelligent networked vehicle system with SPZN is shown in Figure 10. First, the vehicle's driving data, node device information, and other data from the smart car are obtained. While initializing the intelligent networked car system, the obtained node device information is abstracted, and the Z framework and node device information are established for the system node. The preconditions, post-conditions, input and output parameters are set, and an SPN model is established for the system information transmission process.

According to the above flowchart, the modeling of the system can be realized and the corresponding Markov chain can be constructed to calculate the steady-state probability. Taking measures to protect the vulnerable parts of the

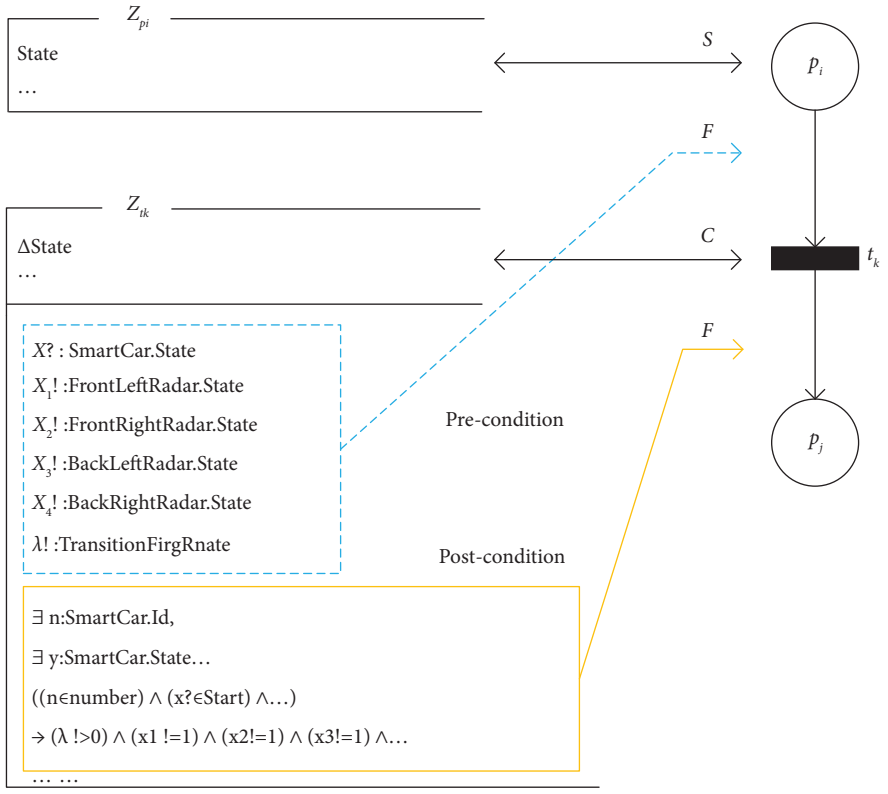


Figure 9: Relationship between SPN and Z in SPZN.

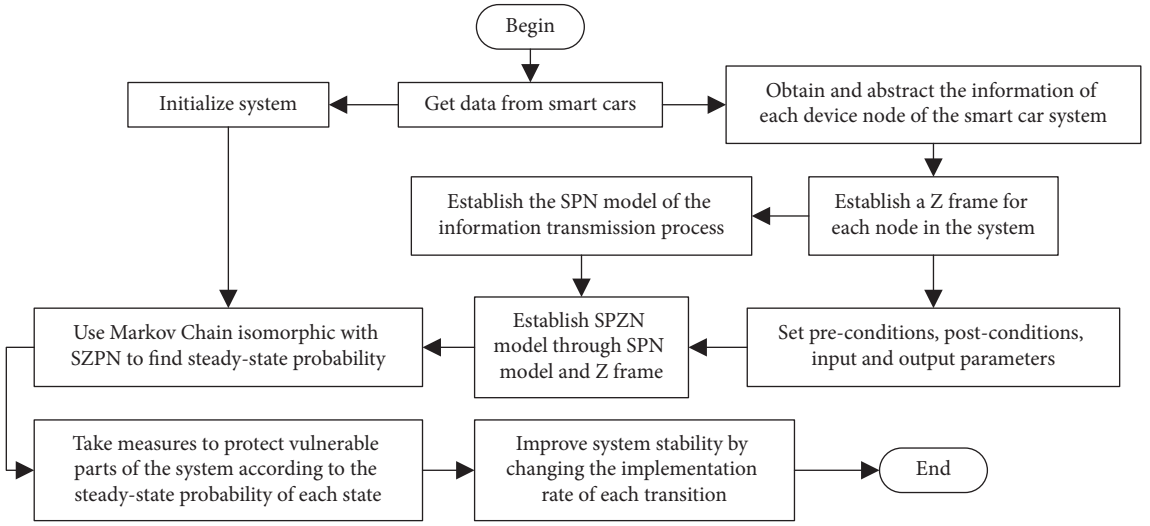


Figure 10: Flow chart of SPZN model establishment.

system based on the steady-state probability, the stability of the system can also be improved by changing the implementation rate of each transition.

3.3. Optimization Method of Parameter λ. In stochastic Petri nets, the transition implementation rate λ needs to be set for each transition. It is known through equation (5) in the Markov process that the transition implementation rate can affect the system steady-state probability. In practical problems, however, it is not easy to make reasonable assumptions about the rate of transition implementation based on improving system stability. In the papers of Song et al. [24] and Sun and Li [23], only the assumed transition implementation rate is used, and no method for determining the transition implementation rate is given. To this end, we propose an optimization method for the transition implementation rate λ, which uses the actor-critic algorithm to

Input: SPN, Environment, the Reward and Punishment Rules.
Output: λ.
Let the initial global time be t and the SPN Contains I Places and m transactions.
(1) The reachable marking graph and Markov Chain with n states Can be derived from the SPN.
(2) Random initialization transactions implementation rate $\lambda_t = \{\lambda_{t,1}, \lambda_{t,2} \ldots \lambda_{t,m}\}$ and let Action = $\{\lambda_{1^{++}}, \lambda_{1^{--}}, \cdots \lambda_{m^{++}}, \lambda_{m^{--}}\}$.
(3) The nth-order square matrix Q is introduced based on the Markov Chain obtained in Step 1 and Definition 1.
(4) According to the Markov Chain in Step 1, the steady-state probability vector $P_t = (P_t(M_0), \ldots P_t(M_{n-1}))$, P_t is an n-dimensional vector.
(5) According to equation (5), the vector P_t at the current moment t is calculated.
(6) Set the set of states $GS = \{g_1, g_2, \ldots, g_t\}$, the set of bad states $BS = \{b_1, b_2, \ldots, b_t\}$. By P_t then the steady-state probability vector of good states $P_{GS,t} = (p(g_{1,t}), p(g_{2,t}), \ldots, p(g_{x,t}))$, the steady-state probability vector for the bad state $P_{BS,t} = (p(b_{1,t}), p(b_{2,t}), \ldots, p(b_{y,t}))$.
(7) iteration = 0;
(8) While (iteration $\geq$ 100)
(1) Observe the current state s_t,i.e., the current steady-state probability P_t. Random initialize the policy network $\pi(\cdot|s_t; \theta_t)$ and the value network $q(s, a; w)$ and randomly sample an action a_t according to the policy network.
(2) Execute action a_t. The environment generates the next states $s_{t+1} = P_{t+1}$, according to action a_t. The reward R_t is calculated according to the reward and punishment rules.
(3) The policy network $\pi(\cdot|s_{t+1}; \theta_t)$ randomly samples an action $a_{t+1}\prime$ according to the current state s_{t+1}, but does not execute the action $a_{t+1}\prime$.
(4) From the value network $q(s, a; w), q_t = (s_t, a_t; w_t), q_{t+1} = (s_{t+1}, a_{t+1}\prime; w_t)$.
(5) Calculate TD error according to TD algorithm $\delta_t = q_t - (R_t + \gamma \cdot q_{t+1})$, and γ is the discount rate.
(6) Calculate the value network gradient $d_{w,t} = \partial q(s_t, a_t, w)/\partial w | w = w_t$.
(7) Update the value network, $w_{t+1} = w_t - \alpha \cdot \delta_t \cdot d_{w,t}$.
(8) Calculate the policy network gradient, $d_{\theta,t} = \partial \log \pi(a_t|s_t, \theta)/\partial\theta|\theta = \theta_t$.
(9).Update the policy network, $\theta_{t+1} = \theta_t + \beta \cdot q_t \cdot d_{\theta,t}$.
(10) t++ iteration++,
where α and β correspond to the learning rates in the value network and policy network, respectively.

Algorithm 1: Parameter λ optimization algorithm based on actor-critic.

train and optimize the transition implementation rate λ in SPZN-SPN to improve the stability of the system, as shown in Algorithm 1.

We initialize the global time t. If there exists an SPN containing l places and m transitions, then the implementation rate $\lambda = \{\lambda_1, \lambda_2, \ldots, \lambda_m\}$ is randomly initialized. The corresponding reachable marking graph and Markov chain containing n states can be obtained from the SPN. From Definition 1, equation (5), and the Markov chain, the steady-state probability vector of each state at the current moment t can be calculated, $P_t = (P_t(M_0), \ldots, P_t(M_{n-1}))$.

In the actor-critic algorithm, there are four elements: state, environment, action, and reward. The environment corresponds to equation (5), and calculating equation (5) yields the current state, i.e., the steady-state probability P_t of each state at the current moment t. We take changing the rate of transition implementation as the action in the algorithm, and for each transition implementation rate λ, there exist two actions, i.e., increasing and decreasing by an order of magnitude, so that for a stochastic Petri net with m transitions there are a total of $2 \times m$ actions. Action = $\lambda\{1++, \lambda_1--, \ldots, \lambda_m++, \lambda_m--\}$, where λ_i++ represents the action that adds one to the value of λ_i and λ_i-- represents the action that subtracts one from the value of λ_i. The reward should be used as part of the environment, but equation (5) does not have its function. Therefore, we redefined the reward and punishment rules of the environment as shown in equations (7) and (8).

The purpose of this algorithm is to increase the probability of stable states and decrease the probability of nonideal states, so we group the stable states into the set of good states $GS = \{g_1, g_2, \ldots, g_x\}$, and the undesirable states are grouped into the set of bad states $BS = \{b_1, b_2, \ldots, b_y\}$, and it should be noted that $GS \cap BS = \varnothing$ and $x + y \leq n$. By finding the steady-state probabilities of the corresponding states from P_t, we have the good state probability vector $P_{GS,t} = (P(g_{1,t}), \ldots, P(g_{x,t}))$ and the bad state probability vector $P_{BS,t} = (P(b_{1,t}), \ldots, P(b_{y,t}))$.

Because changing the rate of transition implementation may cause the probability of good states and bad states to increase or decrease simultaneously, it is stipulated that the probability of a good state at time $t + 1$ is higher than the probability of time t, and then, the positive feedback score R_t^+ given by the environment is greater and vice versa. The greater the probability of a bad state, the greater the negative feedback score R_t^- given by the environment and vice versa. If the probabilities of good states and bad states are the same, then the positive and negative feedback scores remain the same. At time t, the score R_t given by the environment is the difference between the positive feedback score and the

negative feedback score. The above is an explanation of equations (7) and (8) and the reward and punishment rules.

$$\begin{cases} P_{GS,t+1} - P_{GS,t} = (P_t(g_1), \ldots, P_t(g_x)), \\ P_{BS,t+1} - P_{BS,t} = \left(P_t(b_1), \ldots, P_t\left(b_y\right)\right), \end{cases} \tag{7}$$

$$\begin{cases} R_t^+ = P_t(g_1) + \cdots + P_t(g_x), \\ R_t^- = P_t(b_1) + \cdots + P_t\left(b_y\right), \\ R_t = R_t^+ - R_t^-. \end{cases} \tag{8}$$

4. Modeling Analysis

4.1. Reachability. Reachability is the most basic dynamic property of Petri nets, and all other properties must be defined by reachability, so whether a Petri net is reachable is crucial.

Definition 3. Let $\Sigma = (P, T, F, M_0)$ be a Petri net; if there is $\exists t \in T$ that makes $M[t > M'$, then M' is called directly reachable from M. If there are transition sequences $t_0, t_1, \ldots, t_{k-1}$ and marking sequences $M_0, M_1, \ldots, M_k$, such that $M_0[t_0 > M_1[t_1 > M_2 \cdots M_{k-1}[t_{k-1} > M_k]$, then M_k is said to be reachable from M_0. The set of all markings reachable from M_0 is denoted as $R(M_0)$.

By abstracting the above definitions, the algorithm for verifying the reachability of Petri nets as shown in Algorithm 2 can be obtained.

Generally speaking, to verify that a Petri net is reachable, it can be judged by constructing a reachable marking graph. As shown in the reachable marking graph in Figure 5(b), it can be seen that each state of the reachable marking graph is reachable, and there is no isolated state, so the Petri net has good reachability.

4.2. Boundedness and Safety. For any Petri net, we have Definition 4 and Definition 5 to verify its safety and boundedness.

Definition 4. Let $\Sigma = (P, T, F, M_0)$ be a Petri net; if there is a positive integer B such that $\forall M \in R(M_0)$: $M(p) \le B$, then the place p is called bounded, and the smallest positive integer B that satisfies this condition is called the bound of place p, denoted as $B(p)$.

$$B(p) = \min\{B | \forall M \in R(M_0): M(p) \le B\}. \tag{9}$$

When $B(p) = 1$, the place p is said to be safe.

Definition 5. Let $\Sigma = (P, T, F, M_0)$ be a Petri net; if any place is bounded, it is called a bounded Petri net.

$$B(\Sigma) = \max\{B(p) | p \in P\}. \tag{10}$$

We call $B(\Sigma)$ the bounds of Σ. When $B(\Sigma) = 1$, we call Σ is safe.

By abstracting Definition 4 and Definition 5, the algorithm for verifying the boundedness and security of Petri nets as shown in Algorithm 3 can be obtained. Among them, $S(p)$ and $S(\Sigma)$ are the security of the place p and Petri net, respectively. When $S = 0$, the object is unsafe, and when $S = 1$, the object is safe. For the Petri net shown in Figure 5(a), by observing its operation, it is not difficult to see that the places p_0 to p_4 are bounded, and their bounds are 1, so they are all safe. According to Definition 4 and Definition 5, the Petri net is safe and bound.

4.3. Advantage. For the recent research on the security of intelligent connected vehicles, Abu et al. [35] discussed what attack methods are available when attacking intelligent connected vehicles, what solutions can be adopted to defend against attacks, and the efficiency comparison of different solutions. Vijayarangam et al. [36] conducted a more in-depth discussion on the "data fascination attack," an attack method in intelligent connected vehicles, and proposed a model to detect data fascination attacks to enhance the security of connected vehicles. Additionally, the authors propose a model to reduce time in the case of traffic jams. With the above two models, ICVs can be prevented from data fascination attacks and the throughput efficiency can be improved.

The above two research studies are all proposed models and methods to improve the security of intelligent connected vehicles when they are attacked. They have not studied the security of intelligent vehicles during operation, but the SPZN model we propose can study the safety of smart car during operation. Since SPN has randomness that TPN does not have, it can vividly describe some random events. It is unavoidable that various random events occur during the vehicle's operation. Therefore, describing random events is the most crucial thing for smart cars. Notably, the best solution to random events is to prevent them in advance.

By adjusting the transiton implementation rate λ in SPN, in this study, an optimization method for λ is also proposed. Through reinforcement learning, the optimal λ value can be quickly found, which improves the security of the system.

Compared with SPN, SPZN also has a Z framework structure that SPN does not have, which can better describe an abstract system. Because the Z framework restricts the places and transitions, it can reduce the number of states in the system. The problem of state explosion in traditional Petri nets is effectively avoided. As shown in Table 2, the advantages of SPZN are more intuitively displayed.

5. A Case Study

To verify the effectiveness of our modeling method for the analysis and verification algorithm, in this section, we simplify the sensor control system of the smart car and only consider the situation of the speed control subsystem when the smart car is driving on a straight road with random events, such as shown in Figure 11. Suppose a smart car has four radar sensors, two video monitors, an acceleration controller, a deceleration controller, a brake, a front sensor subsystem, a rear sensor subsystem, an acceleration system, a deceleration system, and a braking system.

The first step of model building is to obtain the Z frame of each node. Due to the limited space, only the Z frame in part of the system model is given.

Input: $\sum$.
Output: $R(M_0)$.
if $\exists t \in T, M_0[t > M$
$M \in R(M_0), R(M_0) = \{M\}$
else if $\exists t_0, t_1, \cdots t_2 \in T, M_0[t_0 > M_1[t_1 > M_2 \cdots M_{k-1}[t_{k-1} > M_k$
$M_1, M_2, \ldots M_k \in R(M_0), R(M_0) = \{M_1, M_2, \ldots M_k\}$
else
$R(M_0) = \{\varnothing\}$.

Algorithm 2: Algorithm for verifying reachability of Petri nets.

Input: $\sum$.
Output: $B(p), B(\sum), S(\mathrm{p})$ and $S(\sum)$.
if $\forall M \in R(M_0)$: $M(p) \leq B$
$B(p) = \min\{B | \forall M \in R(M_0) M(p) \leq B\}$
if $B(p) = 1$
$S(p) = 1$
else
$S(p) = 0$
if $\forall p \in P, \exists B(p)$
$B(\sum) = \max\{B(p) | p \in P\}$
if $B(\sum) = 1$
$S(\sum) = 1$
else
$S(\sum) = 0$
else
$B(\sum) = 0$
else
$B(p) = 0$

Algorithm 3: Algorithms for verifying the boundedness and security of Petri nets.

Table 2: Difference between Z, PZN, TPZN, SPN, and SPZN.

	Framework	Dynamic	Randomness
Z	✓	×	×
PZN	✓	✓	×
TPZN	✓	✓	×
SPN	×	✓	✓
SPZN	✓	✓	✓

SmartCar

Id: number

Brand: T oyota, Honda, Ford, BYD, ...

Type: small, middle, large,

Version: VersionNumber

AnticollisionRadar: FrontLeftRa, FrontRightRa, BackLeftRa, BackRightRa, LeftRa, RightRa
Monitor: FrontMo, BackMo
System:RadarSystem, MonitorSystem, AcceleratingSystem, BrakeSystem, ...

... ...

State: Start, Stop, Acceleration, Deceleration, ...

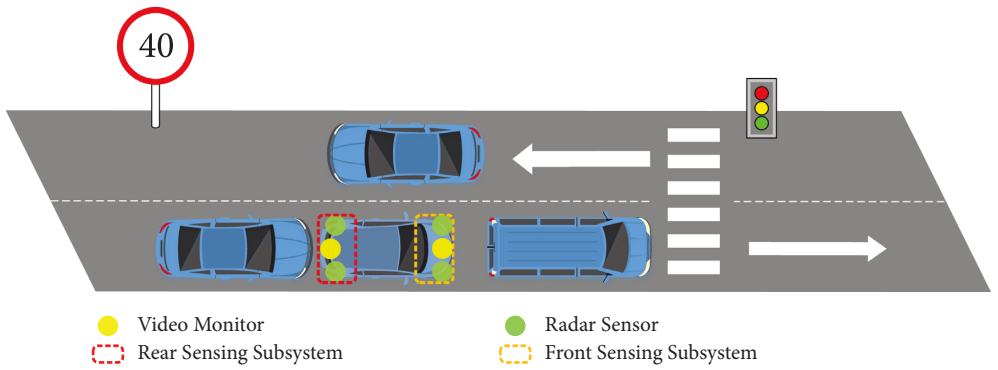


Figure 11: Illustration of smart car running in a case environment.

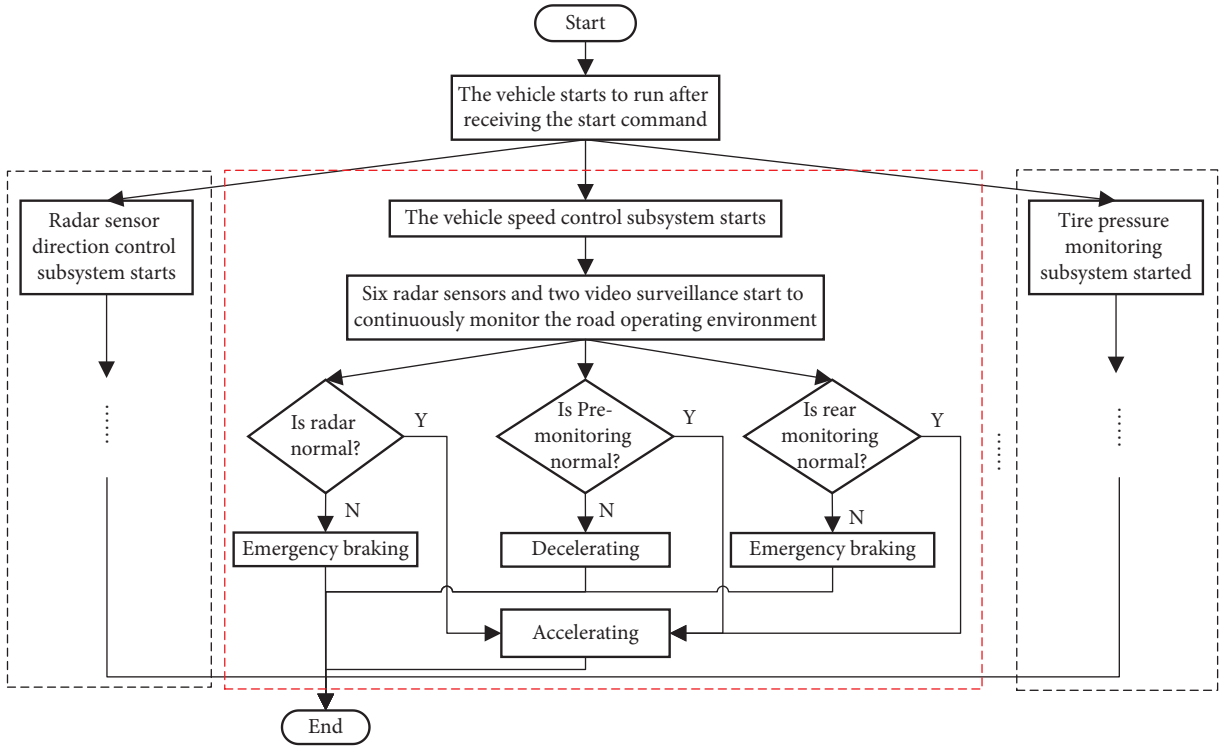


Figure 12: Flowchart in the case study.

The frame above shows the definition of the Z frame for smart cars, which is represented by Z_p in the system, and the blue dashed box in the picture defines the Z frame for the related equipment in the system. The following picture shows the definition of the Z frame in system. A certain node device is defined, and it is also an element in Z_p.

FrontLeftRa
Name: FrontLeftRadar
Type: AnticollisionRadar
Time: LocalTime, GlobalTime
Distance: SaftDistance, DangerDistance, LimitDistance
State: Close, Error, Activation

In the frame below, we have defined action in the system. It is also an element in Z_t, corresponding to a transition in SPZN.

START
ΔSmartCar, ΔFrontLeftRa, ΔFrontRightRa
ΔBackLeftRa, ΔBackRightRa, ΔFrontMo, ΔBackMo
. . . . . .
x?: SmartCar.State
x1!: FrontLeftRadar.State
x2!: FrontRightRadar.State
x3!: BackLeftRadar.State
x4!: BackRightRadar.State
λ!: StateTransitionProbility
. . . . . .

∃ n:SmartCar.Id, ∃ y:SmartCar.State . . .
((n ∈number)^(x?∈Start)^. . .)
→(λ!>0)^(x1!=1)^(x2!=1)^(x3!=1)^
. . .

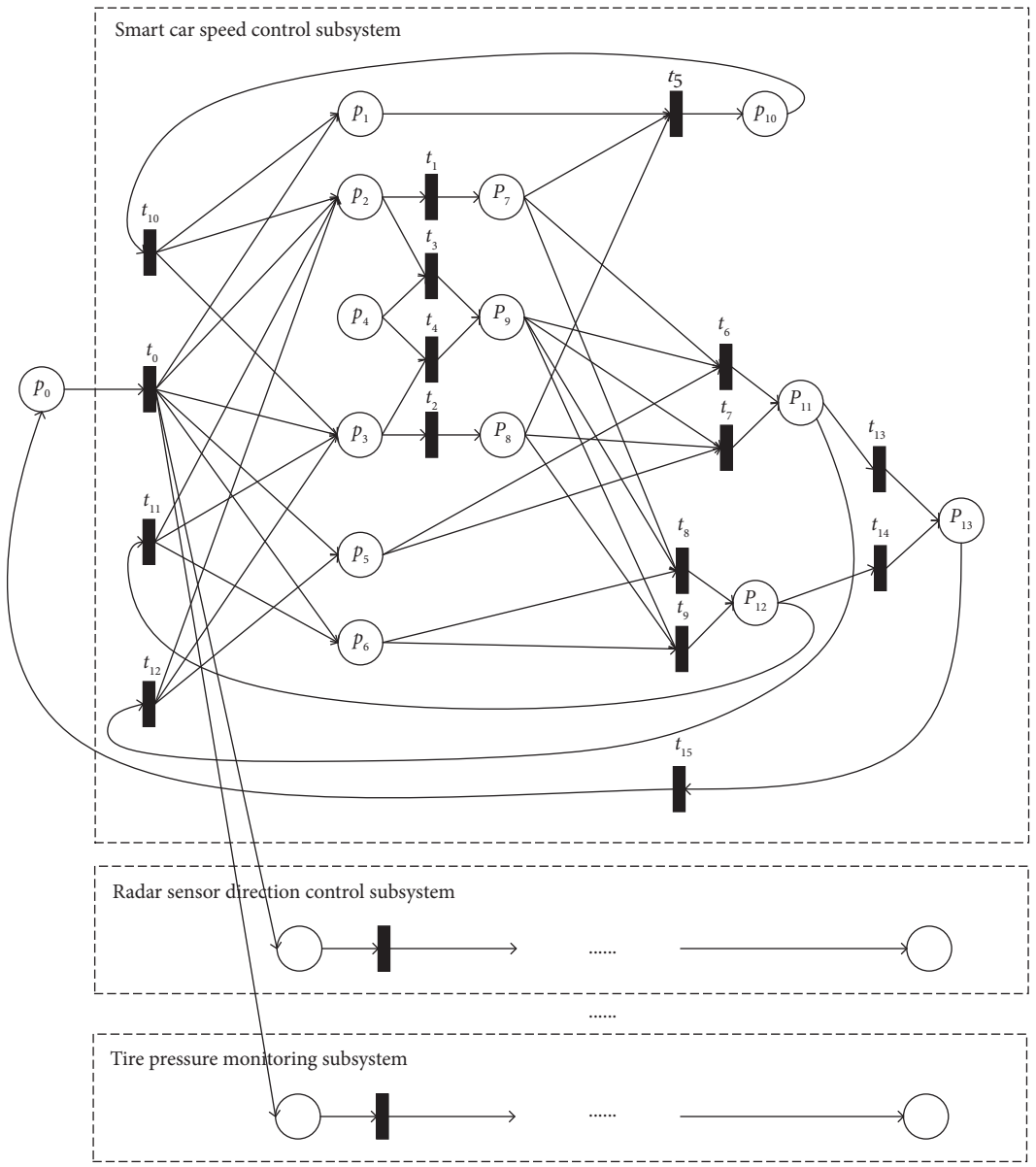


Figure 13: SPN model of the case study.

In the first step of model building, we obtained the Z frame structure of each node device, place, and transition. The SPN model of the intelligent vehicle speed control subsystem will be constructed in the next second step. Before constructing the SPN model, we analyzed the process of this subsystem and got the flow chart shown in Figure 12. The part of the constructed SPN model is shown in Figure 13.

In the intelligent vehicle speed control subsystem shown in Figure 13, the transition t_0 is triggered from the initial place p_0, indicating that the vehicle has been started. Then, the various sensors of the vehicle are monitored when the vehicle is started. If a random event occurs, corresponding processing will be performed according to the type of random event. There are usually three processing modes in the vehicle speed control subsystem: acceleration, deceleration, and braking. After processing random events, the system continues to monitor individual sensors. The meaning of each place and transition is shown in Table 3.

It can be seen from Figure 13 and Table 3 that there are mainly two situations in this case: the normal driving of the vehicle and the random event of the vehicle. Therefore, we will discuss the above two situations separately.

5.1. The Normal Driving Condition of the Vehicle. In this subsection, only the normal driving of the smart car is considered, and there will be no random events before and after the vehicle. Therefore, it is very easy to obtain the SPN model of the normal driving of the car as shown in Figure 14, and the meanings of the corresponding places and transitions in this figure are shown in Table 4.

Table 3: Meaning of each place and transition.

Place	Meaning
p_0	Initial state of the vehicle
p_1	Acceleration controller
p_2	Front sensor subsystem
p_3	Rear sensor subsystem
p_4	Random events
p_5	Deceleration controller
p_6	Automobile brake
p_7	Front sensor subsystem is normal
p_8	Rear sensor subsystem is normal
p_9	Warning message
p_{10}	Keep running
p_{11}	Deceleration
p_{12}	Brake
p_{13}	Vehicle damage
Transition	**Meaning**
t_0	Start
t_1	Processing normal information for front sensor subsystem
t_2	Processing normal information for rear sensor subsystem
t_3	Handling exception information for front sensor subsystem
t_4	Handling exception information for rear sensor subsystem
t_5	Accelerating
t_6	Slowdown caused by random events behind the vehicle
t_7	Slowdown caused by random events in front of the vehicle
t_8	Braking due to random events behind the vehicle
t_9	Braking due to random events in front of the vehicle
t_{10}	Continuous monitoring of vehicle status during normal driving
t_{11}	Slow down and continuously monitor vehicle status
t_{12}	Braking and continuous monitoring of vehicle status
t_{13}	Vehicle collides while decelerating
t_{14}	Vehicle collides while braking
t_{15}	Sending the vehicle to a repair shop for repair

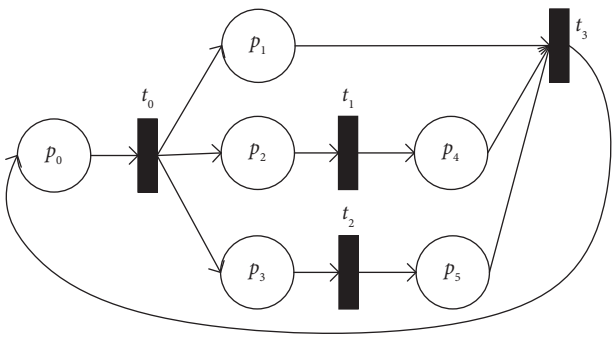


Figure 14: SPN model under normal driving of the vehicle.

It is easy to construct its reachable marking graph according to the constructed SPN model. Because the reachable marking graph of SPN is isomorphic with the Markov chain, the Markov chain can be used to calculate the steady-state probability of each possible state and use related mathematical methods to analyze the equilibrium state and change law of the system. Therefore, we constructed the isomorphic Markov chain as shown in Figure 15 from the reachable marking graph.

It can be seen from Figure 15 that the Markov chain contains 5 states, so we can obtain a 5×5-order transition matrix Q from Definition 1, as shown as follows. Assume that there is a steady-state probability vector $P = (P(M_0), P(M_1), \ldots, P(M_4))$, and the following equation can be obtained from equation (5):

$$Q = \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 & 0 & 0 \\ 0 & -\lambda_1 - \lambda_2 & \lambda_1 & \lambda_2 & 0 \\ 0 & 0 & -\lambda_2 & 0 & \lambda_2 \\ 0 & 0 & 0 & -\lambda_1 & \lambda_1 \\ \lambda_3 & 0 & 0 & 0 & -\lambda_3 \end{pmatrix},$$

$$\begin{cases} \lambda_0 P(M_0) = \lambda_3 P(M_4), \\ \lambda_0 P(M_0) = (\lambda_1 + \lambda_2) P(M_1), \\ \lambda_1 P(M_1) = \lambda_2 P(M_2), \\ \lambda_2 P(M_1) = \lambda_1 P(M_3), \\ \lambda_3 P(M_4) = \lambda_2 P(M_2) + \lambda_1 P(M_3), \\ \sum_{0}^{4} P(M_i) = 1. \end{cases} \tag{11}$$

Assume that the value of the transition implementation rate λ_i is shown in Table 5; then, we can get the steady-state probability in each state, as shown in Table 6.

From Table 6, the steady-state probability $P(M_i)$ of each state M_i occurring at the corresponding implementation

Table 4: Meaning of each place and transition.

Place	Meaning
p_0	The vehicle is running normally
p_1	Acceleration controller
p_2	Front sensor subsystem
p_3	Rear sensor subsystem
p_4	Front sensor subsystem is normal
p_5	Rear sensor subsystem is normal
Transition	Meaning
t_0	Continuous monitoring of vehicle status during normal driving
t_1	Processing normal information for front sensor subsystem
t_2	Processing normal information for rear sensor subsystem
t_3	Accelerating

rate of transition can be obtained. The steady-state probability should be as large as possible for the state favorable to the system stability, and the steady-state probability should be as small as possible for the state unfavorable to the system stability. For a state with a small probability, if the state is a key state in the system, relevant physical means can be used to increase the probability of the state, thereby improving the stability of the system. For example, the vulnerable parts of the vehicle should be inspected and repaired regularly. In addition, the probability of the state can also be changed by changing the rate of implementation of the transitions. For the state M_0, its steady-state probability is 0.4651, which ranks first among all states. The meaning of the state M_0 is that the vehicle is running normally, so there is no need to adjust the transition implementation rate λ_i of the SPN. However, it should be noted that the meaning of state M_2 is to process the normal information of the sensor in front of the vehicle first, and the meaning of state M_3 is to process the normal information of the sensor behind the vehicle first. Although the two have the same effect, the steady-state probability is different. The reason is that the transition implementation rates λ_1 and λ_2 corresponding to the two states are different. It is not difficult to see that different transition implementation rates have a significant impact on the steady-state probability.

5.2. The Random Event of the Vehicle. In this subsection, because the random events in front of the vehicle and the random events behind the vehicle are essentially the same, only the situation where the random event occurs in front of the vehicle is considered, and corresponding measures are taken according to the different random events that occur. The constructed SPN model is shown in Figure 16, and the meanings of the places and transitions in this figure are shown in Table 7, and the corresponding Markov chain is constructed as shown in Figure 17.

It can be seen from Figure 17 that the Markov chain contains 8 states, so we can obtain a 8×8-order transition matrix Q from Definition 1, as shown as follows. Assume that there is a steady-state probability vector $P = (P(M_0), P(M_1), \ldots, P(M_7))$, and the following equation can be obtained from equation (5):

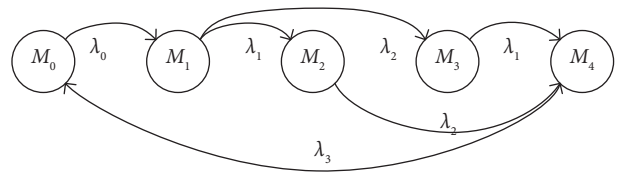


Figure 15: Markov chain corresponding to the SPN model under the normal driving of the car.

Table 5: Implementation rate of each transition.

Rate	Value
λ_0	2
λ_1	3
λ_2	5
λ_3	6

Table 6: Steady-state probability in each state.

State	Probability
$M_0(1,0,0,0,0,0)$	0.4651
$M_1(0,1,1,1,0,0)$	0.1163
$M_2(0,1,0,1,1,0)$	0.0698
$M_3(0,1,1,0,0,1)$	0.1938
$M_4(0,1,0,0,1,1)$	0.1550

$$Q = \begin{pmatrix} -\lambda_0 & \lambda_0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda_1 & \lambda_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -\lambda_2 & \lambda_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -\lambda_3 - \lambda_4 & \lambda_3 & \lambda_4 & 0 \\ 0 & 0 & \lambda_5 & 0 & -\lambda_5 - \lambda_7 & 0 & \lambda_7 \\ 0 & 0 & \lambda_6 & 0 & 0 & -\lambda_6 - \lambda_8 & \lambda_8 \\ \lambda_9 & 0 & 0 & 0 & 0 & 0 & -\lambda_9 \end{pmatrix}. \tag{12}$$

Assume that the transition implementation rate λ_i is shown in Table 8; according to equation (12), the steady-state probability shown in Table 8 can be obtained.

$$\begin{cases} \lambda_0 P(M_0) = \lambda_9 P(M_6), \\ \lambda_0 P(M_0) = \lambda_1 P(M_1), \\ \lambda_1 P(M_1) = \lambda_2 P(M_2), \\ \lambda_2 P(M_2) = (\lambda_3 + \lambda_4) P(M_3), \\ \lambda_3 P(M_3) = (\lambda_5 + \lambda_7) P(M_4), \\ \lambda_4 P(M_3) = (\lambda_6 + \lambda_8) P(M_5), \\ \lambda_7 P(M_4) + \lambda_8 P(M_5) = \lambda_9 P(M_6), \\ \sum_0^6 P(M_i) = 1. \end{cases} \tag{13}$$

It can be seen from Table 9 that the steady-state probability $P(M_0)$ of state M_0 is 0.2907, which is the largest

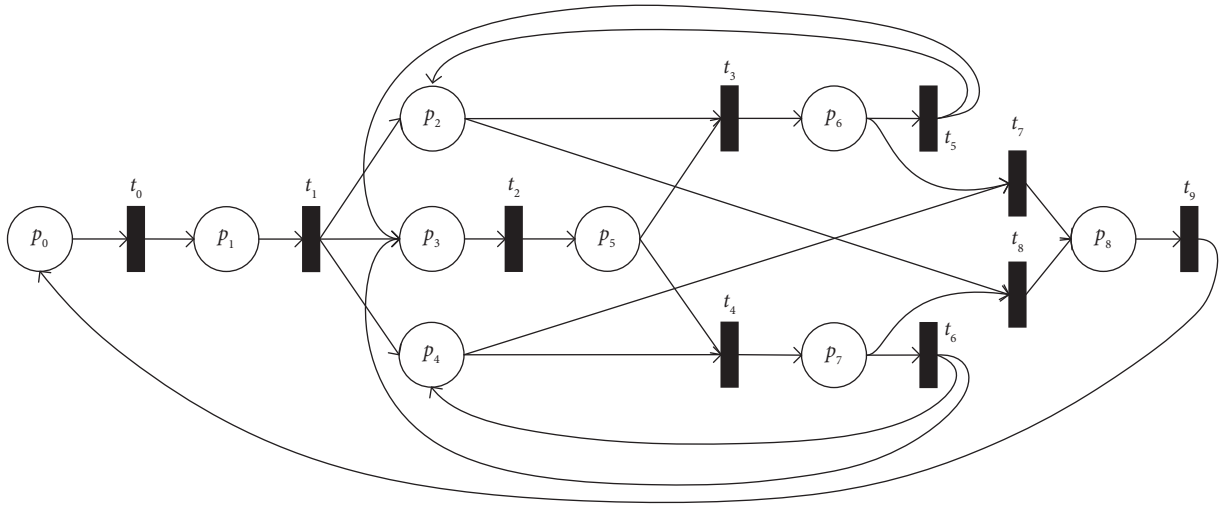


Figure 16: SPN model for random events in front of the vehicle.

among all steady-state probabilities. It can be seen from Table 10 that the meaning of the M_0 state is the vehicle initialization state. For a smart car, the initialization state is obviously not the state we want to maintain for a long time, but this state is not an unacceptable state. So, the state is neither a good state nor a bad state. For the good state in the system, the steady-state probability needs to be increased, and for the bad state in the system, the steady-state probability needs to be reduced, thereby improving the system stability.

We keep the other transition implementation rates unchanged and only change the value of the transition implementation rate λ_0, so the change trend of the steady-state probability of each state as shown in Figure 18 can be obtained. It can be seen that with the increase in λ_0, that is, the increase in the starting trigger rate of the vehicle, the steady-state probability of state M_0 will decrease rapidly, and the probability of being in other states will also increase with the increase in λ_0. Different states have different magnitudes of increase. The steady-state probability increases the most for state M_1 in this figure, since this state is directly affected by the transition implementation rate λ_0. It is not difficult to see that when $\lambda_0 > 13$, the steady-state probability of each state tends to be stable, and the steady-state probability of state M_1 is the highest. The corresponding meaning of state M_1 is that the vehicle is running normally, which is the state we want to achieve and maintain for a long time, so increasing the value of λ_0 helps to maintain the stability of the system.

We know that in normal driving, smart cars need to prevent and avoid random events as much as possible.-Whether it is a random event inside the car such as brake failure or high engine temperature, or a random event outside the car such as emergency braking of the car in front or pedestrians passing by, it can cause a huge impact on the normal driving of smart cars. For random events outside the car, we cannot predict in advance, but we can formulate different measures to prevent different random events. For random events in the car, protective facilities can be installed in the relevant parts of the car, and relevant measures can be taken to prevent them.

Table 7: Meaning of each place and transition.

Place	Meaning
p_0	Initial state of the vehicle
p_1	The vehicle is running normally
p_2	Deceleration controller
p_3	Front sensor subsystem
p_4	Automobile brake
p_5	Warning message
p_6	Deceleration
p_7	Braking
p_8	Vehicle damage
Transition	**Meaning**
t_0	Start
t_1	Continuous monitoring of vehicle status during normal driving
t_2	Handling exception information for front sensor subsystem
t_3	Slowdown caused by random events in front of the vehicle
t_4	Braking due to random events in front of the vehicle
t_5	Slow down and continuously monitor vehicle status
t_6	Braking and continuous monitoring of vehicle status
t_7	Vehicle collides while decelerating
t_8	Vehicle collides while braking
t_9	Sending the vehicle to a repair shop for repair

Through the above experiments, it is found that the transition implementation rate has a significant impact on the steady-state probability. Therefore, we use the reinforcement learning method to optimize the transition implementation rate λ and take corresponding preventive measures and solutions to the system according to the optimized transition implementation rate, which can effectively improve the security of the system in a stochastic dynamic environment.

In this case, there are a total of 7 states, in which the set of good states is only M_1, and the set of bad states is only M_6.

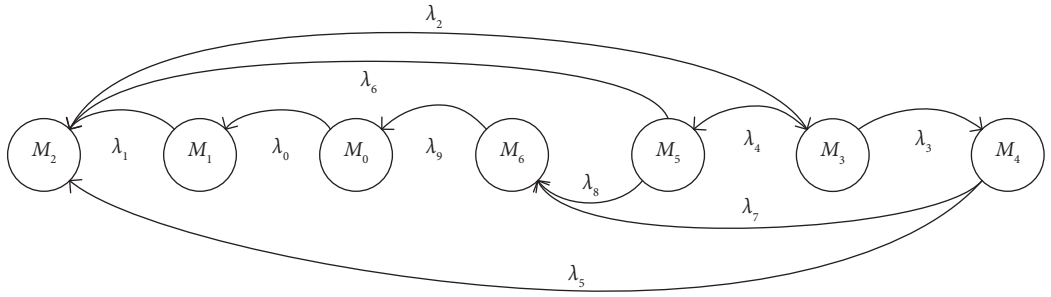


Figure 17: Markov chain corresponding to the SPN model when a random event occurs in front of the car.

Table 8: Implementation rate of each transition.

Rate	Value
λ_0	3
λ_1	4
λ_2	5
λ_3	5
λ_4	3
λ_5	7
λ_6	6
λ_7	2
λ_8	2
λ_9	7

Table 9: Steady-state probability in each state.

State	Probability
$M_0(1,0,0,0,0,0,0,0,0)$	0.2907
$M_1(0,1,0,0,0,0,0,0,0)$	0.2180
$M_2(0,0,1,1,1,0,0,0,0)$	0.1744
$M_3(0,0,1,0,1,1,0,0,0)$	0.1090
$M_4(0,0,0,0,1,0,1,0,0)$	0.0606
$M_5(0,0,1,0,0,0,0,1,0)$	0.0227
$M_6(0,0,0,0,0,0,0,0,1)$	0.1246

Table 10: Corresponding meaning of each state.

State	Meaning
M_0	Vehicle initialization
M_1	The vehicle is running normally
M_2	Sensors continuously monitor vehicle status
M_3	The sensor receives abnormal information
M_4	Vehicle deceleration
M_5	Vehicle braking
M_6	Vehicle collision damage

According to the previous definitions of action, environment, good state set, and bad state set, Table 11 can be obtained.

After the basic parameters are set, we still use the data in Table 8 to initialize the transition implementation rate λ. The state corresponding to the transition implementation rate, that is, the steady-state probability vector of each state, is $s = (0.2707, 0.2180, 0.1744, 0.1090, 0.0606, 0.0227, 0.1246)$. After simulation with Algorithm 1, the optimized state is obtained as

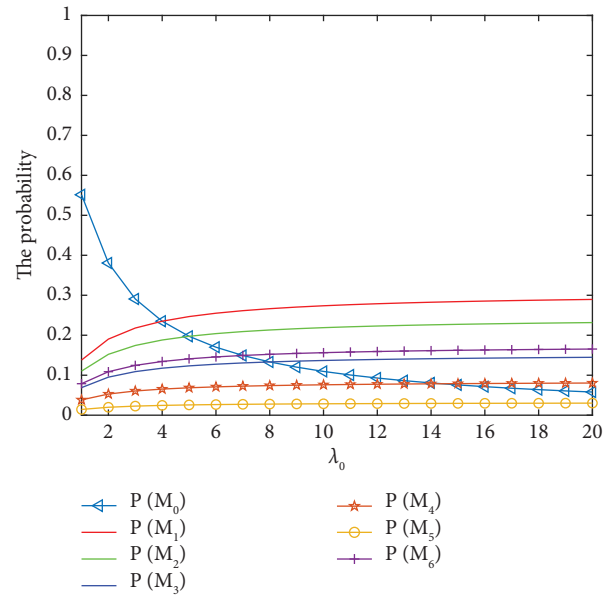


Figure 18: Steady-state probability trend of each state when only changing λ_0.

Table 11: Parameters involved in the experiment.

Parameter	Contents
Action	$\{\lambda_0++, \lambda_0--, \ldots, \lambda_9++, \lambda_9--\}$
GS	$\{M_1\}$
BS	$\{M_6\}$
s_t	$P_t = (P_t(M_0), P_t(M_1), \ldots, P_t(M_6))$
Environment	$\lambda_0 P(M_0) = \lambda_9 P(M_6)$ $\lambda_0 P(M_0) = \lambda_1 P(M_1)$ $\lambda_1 P(M_1) = \lambda_2 P(M_2)$ $\lambda_2 P(M_2) = (\lambda_3 + \lambda_4) P(M_3)$ $\lambda_3 P(M_3) = (\lambda_5 + \lambda_7) P(M_4)$ $\lambda_4 P(M_3) = (\lambda_6 + \lambda_8) P(M_5)$ $\lambda_7 P(M_4) + \lambda_8 P(M_5) = \lambda_9 P(M_6)$ $\sum_0^6 P(M_i) = 1$

$s^* = (0.1099, 0.3297, 0.3297, 0.1099, 0.0366, 0.0110, 0.0733)$. It is not difficult to see that the steady-state probability of state M_1 has increased from 0.2180 to 0.3297, and the steady-state probability of state M_6 has decreased from 0.1246 to 0.0733, successfully increasing the steady-state probability of good states

Table 12: Implementation rate of each transition after optimized.

Rate	Value
λ_0	6
λ_1	2
λ_2	2
λ_3	3
λ_4	3
λ_5	7
λ_6	8
λ_7	2
λ_8	2
λ_9	9

and reducing the steady-state probability of bad states. The optimized transition implementation rates are shown in Table 12.

By observing the optimized transition implementation rates, we can find that compared with the previous transition implementation rates, λ_0, λ_6, and λ_9 increase, λ_1, λ_2, and λ_3 decrease, and λ_4, λ_5, λ_7, and λ_8 remain unchanged. According to the corresponding meaning of each transition implementation rate, on the basis of the original vehicle setting, increasing the vehicle start trigger rate and the vehicle brake trigger rate will help to improve the system stability. Prompt maintenance after a car crash also contributes to the stability and safety of the vehicle.

6. Conclusions

In this study, a formal modeling and verification method for smart cars based on stochastic Petri nets and Z language framework is proposed. With this method, the simulation of stochastic events during the driving process of smart cars can be better implemented. In addition, the study uses reinforcement learning to optimize the proposed method to improve the safety of smart connected cars. Experimental cases are given to explain and study the method in detail. Although the method addresses the stochastic nature of events and the abstraction of the system well, it only considers the speed control system of a single smart car driving on a straight road and does not consider the coordinated control of multiple smart vehicles. The complexity and size of the current model will not be conducive to modeling and validation using Petri nets. How to abstract and simplify the model will be our next work.

Data Availability

The experimental data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

Acknowledgments

This work was supported by the National Natural Science Foundation of China under Grants 62273065 and 61903053 and the Science and Technology Research Program of Chongqing Municipal Education Commission in China under Grants KJZD-K201800701 and KJQN201900702.

References

[1] C. Chen, L. Liu, and S. H. Wan, "Data dissemination for industry 4.0 applications in internet of vehicles based on short-term traffic prediction," *ACM Transactions on Internet Technology*, vol. 22, no. 1, pp. 1–18, 2021.

[2] W. Wei, R. Yang, and H. Gu, "Multi-objective optimization for resource allocation in vehicular cloud computing networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 1, pp. 1–10, 2021.

[3] C. Chen, Y. Zhang, and Z. Wang, "Distributed computation offloading method based on deep reinforcement learning in ICV," *Applied Soft Computing*, vol. 103, no. 7, pp. 1–11, 2021.

[4] C. Wang, C. Chen, and Q. Pei, "An information centric in-network caching scheme for 5g-enabled internet of connected vehicles," *IEEE Transactions on Mobile Computing*, vol. 69, no. 12, pp. 1–14, 2021.

[5] C. Wang, C. Chen, and Q. Pei, "Popularity incentive caching for vehicular named data networking," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 4, pp. 3640–3653, 2020.

[6] S. Liu, J. Yu, and X. Deng, "FedCPF: an efficient-communication federated learning approach for vehicular edge computing in 6G communication networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 2, pp. 1616–1629, 2021.

[7] L. Zhao, C. Wang, and K. Zhao, "INTERLINK: a digital twin-assisted storage strategy for satellite-terrestrial networks," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 1, 2022.

[8] N. Soni, R. Malekian, and D. Andriukaitis, "Internet of vehicles based approach for road safety applications using sensor technologies," *Wireless Personal Communications*, vol. 105, no. 4, pp. 1257–1284, 2019.

[9] C. M. Martinez, M. Heucke, and F. Y. Wang, "Driving style recognition for intelligent vehicle control and advanced driver assistance: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 666–676, 2017.

[10] A. Colombo, G. R. D. Campos, and F. D. Rossa, "Control of a city road network: distributed exact verification of traffic safety," *IEEE Transactions on Automatic Control*, vol. 62, no. 10, pp. 4933–4948, 2017.

[11] F. Khalid, S. R. Hasan, and O. Hasan, "Runtime hardware trojan monitors through modeling burst mode communication using formal verification," *Integration*, vol. 61, no. 3, pp. 62–76, 2018.

[12] Y. Liu, J. Z. Wu, and R. Qiao, "Consistency verification between goal model and process model in requirements analysis of networked software," *Journal of Computational and Theoretical Nanoscience*, vol. 11, no. 5, pp. 1385–1393, 2014.

[13] Y. Liu, J. Z. Wu, and R. Qiao, "Dynamic evolution of requirements process model deployed on networked environment with PZN," *Journal of Computational Information Systems*, vol. 9, no. 8, pp. 3329–3336, 2013.

[14] Y. Liu, J. Z. Wu, and R. Zhao, "Formal verification of process layer with Petri nets and Z," *Advances in Information Sciences and Service Sciences*, vol. 5, no. 1, pp. 68–77, 2013.

[15] P. Herrmann and J. O. Blech, "Formal model-based development in industrial automation with reactive blocks," *Federation of International Conferences on Software Technologies: Applications & Foundations*, vol. 9946, no. 12, pp. 253–261, 2016.

[16] K. Rahul and S. Dutta, "Formal verification of a medical insurance system prototype: the event-B modeling approach," *Journal of Information Assurance & Security*, vol. 17, no. 1, pp. 25–34, 2022.

[17] Y. H. Wang, Q. Zhou, and Y. Zhang, "A formal modeling and verification scheme with an RNN-based attacker for CAN communication system Authenticity," *Electronics*, vol. 11, no. 11, pp. 1–21, 2022.

[18] L. Qi, M. C. Zhou, and W. J. Luan, "Emergency traffic-light control system design for intersections subject to accidents," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 170–183, 2015.

[19] L. Qi, M. C. Zhou, and W. J. Luan, "Impact of driving behavior on traffic delay at a congested signalized intersection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 7, pp. 1882–1893, 2016.

[20] L. Qi, M. C. Zhou, and W. J. Luan, "A two-level traffic light control strategy for preventing incident-based urban traffic congestion," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 13–24, 2016.

[21] K. Labadi, T. Benarbia, and J. P. Barbot, "Stochastic Petri net modeling, simulation and analysis of public bicycle sharing systems," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 4, pp. 1380–1395, 2017.

[22] Y. Liu, L. Y. Huang, and J. W. Chen, "Formal verification on the safety of internet of vehicles based on TPN and Z," *Mathematical Problems in Engineering*, vol. 2020, no. 2020, Article ID 6618168, 11 pages, 2020.

[23] Q. Y. Sun and X. Y. Li, "Establishment of emergency coordination model across organizations based on stochastic Petri net," *Journal of Safety Science and Technology*, vol. 11, no. 9, pp. 63–69, 2015.

[24] Y. B. Song, H. B. Mou, and Z. Y. Jiang, "Safety performance analysis of urban rail transit system based on stochastic Petri net," *China Safty Science Journal*, vol. 21, no. 9, pp. 82–87, 2011.

[25] D. Nallaperuma, R. Nawaratne, and T. Bandaragoda, "Online incremental machine learning platform for big data-driven smart traffic management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4679–4690, 2019.

[26] B. Ji, X. Zhang, and S. Mumtaz, "Survey on the internet of vehicles: network architectures and applications," *IEEE Communications Standards Magazine*, vol. 4, no. 1, pp. 34–41, 2020.

[27] M. K. Molloy, "Discrete time stochastic Petri nets," *IEEE Transactions on Software Engineering*, vol. 11, no. 4, pp. 417–423, 1985.

[28] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Transactions on Computers*, vol. 31, no. 9, pp. 913–917, 2006.

[29] A. D. 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, 2015.

[30] M. A. Marsan, G. Balbo, and G. Chiola, "An introduction to generalized stochastic Petri nets," *Microelectronics Reliability*, vol. 31, no. 4, pp. 699–725, 1991.

[31] Z. H. Muhamad, D. A. Abdulmonim, and B. Alathari, "An integration of uml use case diagram and activity diagram with Z language for formalization of library management system," *International Journal of Electrical and Computer Engineering*, vol. 9, no. 4, pp. 3069–3076, 2019.

[32] P. F. Niu, X. F. Wang, and L. Lu, "Survey on vehicle reinforcement learning in routing problem," *Computer Engineering and Applications*, vol. 58, no. 1, pp. 41–55, 2022.

[33] S. M. Yang, Z. Shan, and Y. Ding, "Survey of research on deep reinforcement learning," *Computer Engineering*, vol. 47, no. 12, pp. 19–29, 2021.

[34] K. H. Du, R. Z. Song, and Q. L. Wei, "Review of reinforcement learning applications in machine games," *Control Engineering China*, vol. 28, no. 10, pp. 1998–2004, 2021.

[35] T. M. Abu, S. Abbas, and Q. Nasir, "Systematic literature review on Internet-of-Vehicles communication security," *International Journal of Distributed Sensor Networks*, vol. 14, no. 12, pp. 1–21, 2018.

[36] S. Vijayarangam, G. C. Babu, and S. Ananda Murugan, "Enhancing the security and performance of nodes in Internet of Vehicles," *Concurrency and Computation: Practice and Experience*, vol. 33, no. 7, pp. 1–10, 2021.