**Probability Arc Petri net and Its Application in Path Optimization**

**基于蚁群算法的路径寻优算法**

**Abstract：**In the present Petri net, there is no proper model to describe the business process with choice structures and it is impossible to judge which branch is optimal with respect to the entire process. In order to solve this problem well, we propose a Probability Arc Petri Net for this problem. The transition Probability and some function are added, in Probability Arc Petri Net. And Aiming at the problem of accuracy for dynamic optimal path finding in intelligent road systems, a Dynamic Path Optimization Algorithm based on Ant Colony Algorithm is proposed. The transition probability of ant colony is changed, and the velocity prediction function is added based in the original transition probability. At the same time, the intelligent road traffic system is modeled by the Probability Arc Petri net, and the optimal path in the road is found according to the transition firing rule. According to the Reachable Graph of the Probability Arc Petri net, the intersection passing through the optimal path is found. In the last, the Dynamic Path Optimization Algorithm is comparing with others algorithm. And analyze the shortcomings of other Petri net in modeling business processes with choice structures.

**Key words:** Probability Arc Petri net; Optimal Path; Speed Prediction; Intelligent Traffic System; Ant Colony Algorithm;

**Introduction**

With the development of information technology, Intelligent Transportation System (ITS) has also emerged. ITS uses computer, wireless communication and other technical means to closely coordinate and communicate with traffic-related factors such as vehicles-object, vehicles-vehicles, so that the traffic management system has real-time, accuracy and high efficiency [1, 2]. The establishment of ITS can obtain the status of roads and infrastructure in real time. ITS has enhanced the service capacity and level of the transportation system to the vehicle, providing safe, efficient and convenient transportation services for the vehicle.

In recent years, with the development of the economy and the pace of life, people have also put forward new requirements for the travel environment, eager for the traffic environment to be comfortable and fast. Computers and road infrastructure provide support for the rapid development of intelligent transportation systems, while intelligent road transportation systems also place new demands on computer technology and communication technology [3]. While meeting the needs of people, ITS realizes large-scale information exchange through advanced technology and establishes a remote communication framework to enable travellers to know road conditions and travel time in advance. The optimal path has become a new requirement for ITS. ITS requires computer technology to give an accurate optimal path finding algorithm. Optimal path finding and modeling has become an important topic for people to study. Since the various factors in the road are constantly changing, the changing factors will affect the running speed of the vehicle and the time to reach the destination. Therefore, the static route finding can no longer meet the needs of ITS, and the dynamic factors affecting the running time of the vehicle need to be considered in the ITS. The optimal path is sought, so the search for the optimal path is to be accurate. Dynamic factors in intelligent road traffic systems are a key factor in finding the optimal path. In the ITS optimal path search, not only the dynamic factors but also the dynamic search are needed. Therefore, an accurate algorithm is needed to optimize the path of the intelligent road traffic system.

However, the analysis of intelligent road traffic systems is inseparable from effective modeling tools. Modeling and simulation play an important role in algorithm optimization for overly complex road traffic systems. The simulation tool can provide some parameters of the current scenario to the model established by the road system. These parameters help traffic planners optimize the road system. Some tools are based on rules that describe behavior in the transportation network, while others are based on mathematical models such as the Lighthill Whitham Richards (LWR) model, the Payne model, etc., which can solve traffic congestion and intelligence. Traffic control and optimization of the carrying capacity of urban transport systems [4]. These performance metrics include the total time spent on all vehicles in the intelligent road transport system, as well as the total delay at the intersection. In models that accurately describe and predict traffic conditions, in addition to the LWR and Payne models, other traffic models can predict traffic state information for future times and different roads, as well as the performance of the analysis system. These models include queuing theory [5], agent-based modeling [6], neural networks [7, 8] and so on. Traffic flow, uptime, delay, and steering in road traffic systems can be predicted from these models. These predicted parameters play an important role in the performance evaluation and optimization strategies of intelligent road transportation systems.

In addition to the above models, Petri net have been used for more than a decade in modeling, performance analysis, and traffic system control. As a visual modeling tool, Petri net has a good effect in simulating the dynamics and concurrent activities of business processes [9, 10]. As a mathematical tool, it enables the system to be controlled by a set of mathematical equations, such as state equations. Petri net modeling adds another modeling paradigm to the previous modeling approach because it can properly describe urban traffic and transportation systems with distributed, parallel, deterministic, random, discrete, and continuous [11]. Therefore, Petri net becomes an effective tool for traffic system modeling and performance analysis. From the model established by Petri net, it can analyze system performance and optimize traffic control.

The dynamic path planning algorithm is the core algorithm of path optimization in ITS, and the path optimization algorithm is fundamentally searching for the path that takes the shortest time. The accuracy of the dynamic path optimization algorithm plays a decisive role in path finding. Different path optimization algorithms, taking into account different influencing factors, will also find differences in the optimal path. However, in the path optimization research, many experts at home and abroad have done a lot of research. In the 1960s, Ramser et al. proposed the optimal path optimization for the first time, mainly combining mathematical theory, cutting plane method, branch and bound method, Dijkstra algorithm [13]. Kensuke Takami et al. [14] proposed an optimization path model based on the form of time function, setting the ratio of parameters according to the increase or decrease of traffic volume to obtain a robust optimal path, which is proposed when it is difficult to predict traffic volume. The model can be considered as a whole to increase and decrease the amount of traffic is very useful, but the model is more complicated to use. Williams Billy M. improved the traditional Kalman filter. The improved adaptive Kalman filter algorithm showed strong adaptability when the traffic volume was unstable. Ghosh Bidisha [15] used Bayesian instead of the traditional The least squares method is used to estimate the parameters of the SARIMA prediction model to solve the integration problem of the model with high-dimensional data, and it can better match the fast-fluctuating traffic changes.

The application of Petri net in intelligent transportation system has also been recognized by some domestic and foreign authoritative experts. Italian scholar A.D. Febbraro based on the hybrid Petri net (HPN) theory, uses a hybrid Petri net to specify traffic control at the intersection, and uses a second-order macro model to simulate the motion of the vehicle extending between two consecutive intersections [30]. In addition, J. Jorge et al. used a continuous Petri net model to establish a macroscopic model of the road traffic system and simulated it to avoid state explosion in large discrete business process modeling [31]. Angela Di Febbraro, Davide Giglio et al. used stochastic Petri net to simulate traffic congestion, intersections, road capacity, etc. to reduce congestion in urban areas [32-33]. In the modeling of road traffic system, hybrid Petri net have advantages in modeling traffic road traffic systems, and at the same time establish refined Petri net models for road factors, road intersection steering, signal control, etc. in traffic road systems. That is, based on the road intersection model, it is expanded into a multi-junction model, and a multi-layer model is established to realize the road network modeling analysis of the whole city, and then combine the actual traffic data to realize short-term traffic flow prediction, dynamic traffic flow evaluation, and guarantee Accuracy of traffic prediction [34-40]. However, the above-mentioned extended Petri net does not uniformly predict the predicted vehicle density, speed, traffic flow and optimal path, and there are still some shortcomings in dynamic path modeling.

This paper is organized as follows: In Sec.2 we give the definition about basic Petri net LPN and the knowledge of ant colony algorithm; in Sec.3, we define PAPN and firing rule, and in Sec. 4, Dynamic Path Optimization Algorithm based on Ant Colony Algorithm(DPOA) is proposed, in Sec. 5, the ITS model was given, and find the optimal Path by the PAPN firing. And, the DPOA compared with other algorithm. Finally, we draw some conclusions and outline future research works in Sec. 6.

**Paper Preparation**

## Petri Net

The Petri net is a kind of special directed graph. It has two disjoint nodes, these are transition and place. The arc from place to transition or from transition to place is relationship arcs.

**Definition1.** *N* = (*P*, *T*, *F*) is a net, where:

1. *P* is a finite set of place;
2. *T* is a finite set of transition, *P* ∪ *T* ≠∅ and *P* ∩ *T* =∅；
3. *F* ⊆ (*P*×*T*) ∪ (*T* × *P*) is a set of directional arcs.

**Definition2.** Input and output set: set *x* ∈*P* ∪ *T* is element of the net N

•x = {y|(y, x)∈ F} is called the input set or the pre-set of x;

x•= {y|(x, y) ∈F} is called the output set or the post-set of x;

*N* = (*P*, *T*, *F*) is a pure net, satisfied ∀*t*∈*T: •t*∩ *t• =*∅.

**Definition3.** A four tuple, ∑ = (*P*, *T*, *F*, *M*) is called Petri net, iff:

1. *N* = (*P*, *T*, *F*) is a net;
2. M: *P→N* is a marking function, where *M0* is the initial marking;
3. ∑ has the following transition firing rules:

(a) For the transition *t*∈*T*, if ∀p∈*•t*: *M(p) ≥1*，then the transition *t* is enabled under the marking M, and it is denoted by M[ t >;

(b) If *M*[*t* >, it represents the transition *t* is enabled under the marking M. And after the fired, producing a new marking *M’*, and it is denoted by *M [t >M’*,

Among them:

**Definition4:** set *LPN* = *(P, T, F, I, O)*, *LPN* = *(LN, M)* is called a Logical Petri Net, if and only when:

1. P is a finite set of place ;
2. *T* = *TD* ∪ *TI* ∪ *TO* is a finite set of transition; *T* ∪ *P*≠*∅,* *t* *TI* ∪ *T­o*: *•t*∩ *t•* =∅

Among them:

(a)*TD* represents the transition of classic Petri net;

(b)*TI*represents the logical input transition set of *T*, and ∀*t*∈*TI*, all the input places of *t* are constrained by a logical input expression *fI*;

(c)*TO* represents the logical output transition set of *T* , and ∀*t*∈*TO*, all the output places of *t* are constrained by a logical input expression *fO*;

1. *F*⊆*(P*×*T)*∪*(T*×*P)* is a finite set of arcs;
2. *I* is a logical restriction input function, such that ∀*t*∈*TI*, *I(t)*=*fI* is a logical input expression;
3. *O* is a logical restriction output function, such that ∀*t*∈*TO*, *O(t)*=*fO* is a logical output expression;
4. *M:P→*{0,1}is a marking function,∀*p*∈*P*, *M(p)* represents the number of token in place *p*;
5. Transition firing rule is:

(a) ∀t∈TD, t is enabled if ∀p∈•t; M(p)=1. After transition t fired, LPN produce a new marking M′: ∀p∈•t: M′(p)=0; ∀p∈t•: M′(p)=M(p)+1;

(b)For∀*t*∈*TI* , *I*(*t*)=f*I* , if *fI* | *M*=•T•, the •t satisfies with the logical expression fI, it is called *t* is enabled under the M; if *t* enabled, it can be firing, and after *t* fired under the marking M, producing a new marking M':∀p∈•t M'(p)=0, ∀p∈t• : M’(p)=M(p)+1, ∀p∉•t ∩ t•：M’(p)=M(p);

(c)For ∀t∈TO, O(t)=fO, if ∀p∈•t, M(p)=1, then *t* is enabled under the M; if *t* enabled, it can fire. And after *t* fired under the marking M, producing a new marking M’:∀p∈•t: M’(p)=M(p)-1, ∀p∉•t ∩ t•: M’(p)=M(p). For t• satisfies fO | M’=•T•, and t• must satisfied the logical expression *fO* under the *M’*.

## Ant Colony Optimization

The ant colony algorithm is a heuristic algorithm that finds the optimal path based on heuristics in the path [57]. The ant colony algorithm is essentially a global optimization algorithm with distributed computing, information feedback and heuristic search features. The characteristics of information feedback and heuristic search in ant colony algorithm are widely used by scholars in path optimization [58].

The idea of the ant colony algorithm in solving the problem is: Firstly, according to the size of the solution problem, a certain number of ants are released to form an ant colony, and then the ants explore the optimal path to the destination on the path, and use these paths as initial solutions. Based on the pheromone concentration or heuristic information on the path, these ants select the path to the next node, repeatedly select the nodes that have not passed, until they reach the destination, repeat the appeal process, and finally find the global optimal solution [46].

When the ant colony algorithm solves the TSP problem, the probability formula of the ant k moving from the city i to the city j at time t is as shown in (2.2):



(2.2)

Where *i* is the initial position in the path finding and *j* is the target city. The distance from city *i* to the target city *j* is represented by *dij*; *τij*(*t*) represents the pheromone concentration left by ants from city *i* to city *j* at time *t*; The value of *ηij*(*t*) is 1/1/*dij.*

The pheromone will be updated during the ant's operation, where (1-ρ) is the volatility coefficient of the pheromone. After the time *n*, the ant completes a cycle, the pheromone expression is expressed by the formula (2.3):

 （2.3）

 （2.4）

in equation 2.4 represents the amount of pheromone left by the kth ant on the path from city *i* to city *j*. represents the total amount of pheromone left by all ants in the process as they pass through the path.

 （2.5）

Where *Q* is a constant indicating the total amount of pheromone released by the ant on the path, and L\_k is the length of the ant walking the path on the path. At the initial time , .

**3 Probability Arc Petri Net**

PAPN combines Petri net, stochastic Petri net, and logical Petri net to define probabilistic arc Petri net. In the probability arc Petri net, the probability that Token enters different libraries is added, that is, the probability that Token is transferred to different libraries after the transition occurs, so that the probability arc Petri net can describe the probability that different resources will be transferred to different states, except for the transfer. The probabilistic outer probability arc Petri net adds other factors to the logical Petri net.

3.1the definition of Probability Arc Petri Net

3.1.1 Probability Arc Petri Nets Definition

**Definition 3.1:** Let PAPN=( *P*，*T*，*F*；*λ*，*TOK*，*t*，Find ( *TokM*)，IF(*t*)，*M*，*Ttim* ) is a Probability Arc Petri Net, where:

(1)*P* = *Pb*∪*Po*∪*Pe* is a finite set of place, each place contains a property collection Pc for storing the properties required by the prediction function.

Among them:

1. *Pb* is a beginning set of place;
2. *Po* is a ordinary set of place;
3. *Pe* is aending set of place;

(2)T = Tb ∪ Tf ∪Te is a finite set of transition, which contains three types of transitions:

Among them:

1. Tb is beginning set of transition, Which contains beginning transitions, ∃*pb*∈;
2. *Tf* is ordinary set of transition, which is running transitions in the modeling, *pi*∈，∀*pi*≠*pb*，*pj*∈，∀*Pj*≠*Pe*
3. *Te* is ending set of transition, which contains ending transitions, ∃*pe*∈;

(3)F⊆(P×T)∪(T×P) is a finite set of directional arcs, and P∩T = Φ，P∪T≠Φ;

(4) λ represents the divert probability of the token on the input arc of the place; λ = { λ1, λ2, λ3, …, λn } is divert probability collection. λi exists on the every input arc of the place;

(5) *TOK* = *TOKM*∪ *TOKO*, TOK represents a finite set of tokens in a Probability Arc Petri net. TOKM is mainly token, TOKO is simulation token. Simulation tokens to simulate the operation of the business process and feed back the optimal path to the primary token. TOKM runs along with the optimal path to the ending place;

(6) t is the time label on TOKO, and t is the sum of time that the TOKO spent on transition firing;

(7) Find(TOKM) is a function, which function is find the post-set transition of the place where TOKM is located, if Find(TOKM)=Te, then TOKO stay in the ending place waiting the TOKM to arrive the ending place;

(8) IF(t) is transition judgment function, which function is judgment whether this transition is a ending transition . TOKM and TOKO contain this function. If IF(t) on the TOKM and IF(t)=Te, then this process ending; if IF(t) on the TOKO and IF(*t*) = *Te*∧ Find( *TokM*) ≠*Te*, then the TOKO makes the Find(TOKM) enable;

(9) *M* : *P*→*N* is a marking function , where *MO­* is the initial marking;

(10) Ttim is the firing time which the TOKM firing on the transition;

**Definition 3.2:** Initial function Initial(λ): The role of the initial function is initialize the λ value on the input arc of each place. Initial moment means that all TOKO have not yet reached Pe, and the value of the λ is undefined.

**Definition 3.3:** Prediction function Pre(v): Predict the factors that affect system performance at some moment in the future based on the collection of attributes in the place. If the influencing factor is constant, the function can be omitted.

**Definition 3.4:** Transfer Probability Update function TPU(x): TPU(x) used to update the λ value on the input arc of place. The TPU(x) performed, when TOKO reaches the ending place.

**Definition 3.5:** Branch Update function Update(x): this function located on TOKM. Performed Update(x), when all the TOKO arrived Pe. The Update(x) is executed once, before the TOKM makes the transition firing. When the latest optimal branch conflicts with the previous optimal branch, the latest optimal branch is taken as the optimal branch. If the optimal branch has not changed, it will continue with the original branch running until the TOKM enters Pe.

**Definition 3.6:** The Tabu List Tabu*P* suitable for storing the place where the TOKO has passed. The Tabu List can control the place that the TOKO passes before it enters again.

**Definition 3.7:** Place Judgment Function IFP(p): This function is used to determine where the next place in the Tabu*t*. If it does not exist in the Tabu*t*, TOKO enters the place, otherwise it cannot enter the place.

3.1.1 The Probability Arc Petri net Firing Rule of Transition

The Petri net Firing Rule refers to the rule that the token followed when the transition is firing. Due to the Probability Arc Petri net classifies token and the divert rate is added to the input arc, the Firing Rule of Transition changed.

The Probability Arc Petri net Firing Rule of Transition as following:

1. When TOKM is included in the place, the TOKO are released in those places. If TOKM reaches the ending place, all the TOKO remain in the ending place.
2. The λ value in each flow relationship is initialized by the Initial(λ) function.
3. In the part with the selected branch, if λ is initial value, the tokens enter the place with high implementation.
4. At the same time, if the logical output expression is satisfied, if the λ value is different, the token is entered into the place with the largest λ on the input arc of the place after the transition is firing.
5. When the transition is firing, TOKO needs to execute IF(t). If IF(t)=Te, the TPU(x) is executed.
6. After TOKO executes the IFP(p) and IFP(p)=Pe, TPU(x) is executed to update the transfer rate on the branch, and an arc of Pe to Find(TOKM) is added. If IFP(*Pi*∈ on TOKM, the token in the place *pi* will not execute the TPU(x).
7. TOKM detects IF(TOKM)=Te before the next transition firing. If IF(TOKM)=Te, then the Probability Arc Petri net operation ends.
8. Before the TOKM enters the transition, it is judged whether the IF(TOKM) is belong to Tf. If they are belong to the Tf, executing the TPU(x) to find the optimal path.
9. When λ on the flow relationship arc is the initial value, TOKM cannot make the transition enable. Only when all TOKO enters the ending place and feeds the optimal path to TOKM, TOKM can cause the transition to be firing.
10. Each transition firing needs to satisfy the logical expression.
11. When TOKO causes the transition returned by Find(TOKM) to be enabled, t and Tabu List are initially empty.

3.1.3 Probability Arc Petri net reachable graph algorithm

Input: SAPN=( *P*，*T*，*F*；*λ*，*TOK*，*t*，Find ( *TokM* )，IF(*t*)，*M*，*Ttim* )；

Output: reachable graph of Probability Arc Petri net;

Step0: When the Probability Arc Petri net is running, the S0 identified by transition first time firing is identified as the root identifier and marked as “new”.

Step1: When there is a mark labeled "new", select an identifier with a "new" node and mark it as S;

Step2: if *S*∈*S*(*old*)，then back to Step 1；

Step3: if ∀*t*∈*T*： ¬*S*[*t*> then make the S marking “leaf”, and back to Step1;

Step4: If ∀*t*∈*T* and *S*[*t*>, then *t* firing, and it will get new mark S’ according firing rule of transition, then mark the S’ as “new”. Connect a directed arc from S to S', and mark the *t* on the directed arc, then change the mark of node S to "old" and execute Step 1;

3.3.2 Probability Arc Petri Simulation Tool

Since the probabilistic arc Petri net is a new model tool based on the logical Petri net, there is no Logical Petri Net simulation tool in the existing Petri net modeling tool, and we need to verify the extended logic Petri net model. The Tina tool simulates the Petri net through an intuitive drawing method. It is a software that supports the time Petri net. It is suitable for a variety of Petri nets such as a library/transition system and a Petri net with time transition. Tina is divided into modeling interface and running interface. Through the reachability analysis and structural analysis interface in the Tools button of the modeling interface, the accessibility of the built model can be analyzed and structurally analyzed. Among them, the reachability analysis obtains the \*.ktz file, which contains the reachable identity of the Petri net and the transition caused by the arrival of the reachable identity. Structurally analyzed the boundedness of the Petri net, the T invariant, the S invariant, and stored in the \*.text text. Tina establishes the Petri net in the form of \*.ndr, which can be single-stepped and randomly executed at runtime, and can set the delay to the transition during single-step execution. In the correctness verification of the Probabilistic Arc Petri nNet, we convert the probabilistic arc Petri nets equivalently and then simulate it through Tina.

Among them, the firing time each token on the transition is realized by the delay time of the transition. Since the Suppression Arc cannot express the logical output (or) in the Logical Petri Net, in the analog logic output (or) expression, the Logical Petri Net is split into three subnets for simulation analysis. The three structures of the logic input and output expression in the Logical Petri Net are transformed into the Time Petri Net with Inhibitor Arcs, and the simulation of the Time Petri Net with Inhibitor Arcs is carried out by the Tina tool.

As shown in Figure 3.2, the three basic structures of the Probabilistic Arc Petri Net are converted into structures that Time Petri Net with Inhibitor Arcs.

（a）Logical output(and) (b)Logical input(and)

(c)Logical input(or)

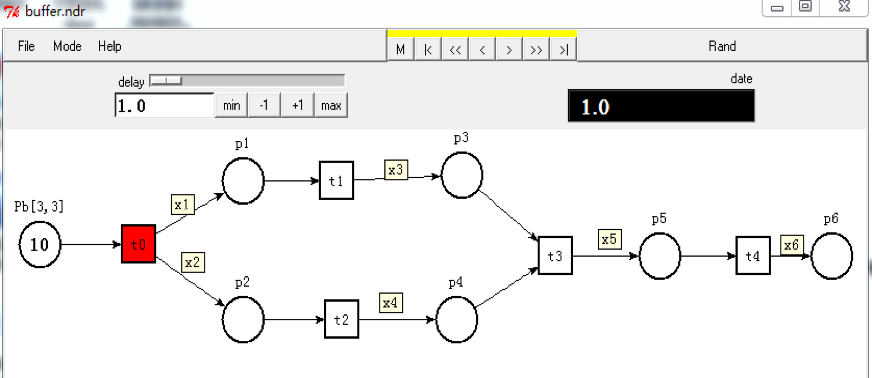
Fig 3.2 Three Basic Structures Represented by the Time Petri Net with Inhibitor Arcs

3.3.1 Modeling and Analysis of Ordinary Road Network Based on SAPN

Fig 4 is an example of Probability Arc Petri net based on the definition of 3.1. The concept of the Probability Arc Petri net and transition rules are illustrated by this example.

Fig 4. Example of Probability Arc Petri net



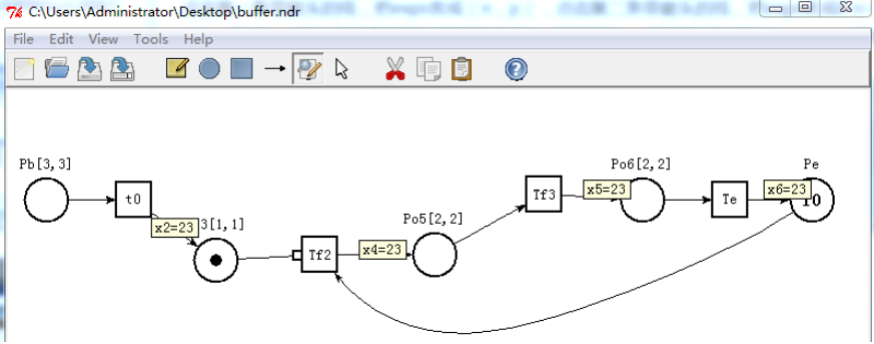
Figure 4 is a modeling of the road network by a Probability Arc Petri net, assuming that the path from *Pb* to *Pe* takes the least time. In order to accurately obtain the path that takes the shortest time, it is necessary to consider the speed change in the road. It is assumed that the speed change on the path follows V = V0 + t, t = L/V, that is, the travel time is inversely proportional to the speed when the path length is fixed. The transfer rate function is λ=1/t. Among them, *POv*, *POL*, *Pev*, *Pel*, *Piv* and *Pil* are the attributes in the attribute set of the place, which represent the running speed of the vehicle and the length of the road. These attributes are the parameters used in predicting the future speed.

In the Fig4, *Pb* is beginning place, which contains two tokens, the red is the main token, and the black is the simulation token. And *PO2*、*PO3*、*PO4*、*PO5*、*PO5* are ordinary places, *Pe* is ending place. There are three types of transitions in the figure, in which *Tb* is initial transition, *Tf1*, *Tf2* and *Tf3* represent the ordinary transition; *Te* represents the ending transition. Where M = 10 represents the number of simulation tokens, and the collection behind the place name is the set of attributes of the place, where each set is represented as [*Poiv*, *Poil*].

3.3.2 The Running Process of Model

Figure 3.5 simulates the Probability Arc by Tina based on the conversion of the logical expression and the Time Petri Net with Inhibitor Arcs and the transition rule of the Probabilistic Arc Petri Net.

Figure 3.5 describes the simulation of the Token in the Probability Arc Petri Net will make the transition t1 firing. And its according to y = V0 + t and t = L / v to get the time of the transition t1 firing is 0.67t. Due to the limitations of the software, the Prediction Function on the place and the translate rate on the input arc of the place need to be manually calculated. When the transition t0 is fired, the Simulation Token will enter the place P1 and P2. When the Simulation Token running, we calculate the time and the translate rate on one branch. TOKM selects the place to be entered after the transition firing, according to the translation rate.

Pb contains 10 Simulation Tokens. At this time, the Simulation Token makes the transition t0 firing. After t0 fired, the time attribute t of all the tokens becomes 1t. Since the values of *λ*1 and *λ*2 are the same as the implementations of P1 and P2. the probability of Simulating Token's entry into P1 and P2 is the same. After the simulation of Token enters P1, the Prediction Function Forec(v) is calculated after 1t, and the time is passed. The speed at which P1 is calculated by the Forec(*v*) is 3, and the time taken for P1 is 0.67t. It is predicted that the time required for t2 firing after 1t is 0.5t, then t2 is firing firstly, and the value of these Token update attributes t is 1.5t, and the time taken to predict t3 is 0.5t. At this time, t1 is fired, and the attribute t is 1.67t in the P3, and the Taboo Table is updated as Pb and P1. The Forec(*v*) predicts that t1 takes 0.64t, while the Forec(*v*) predicts t2 to take 0.4t, so the Simulation Token in p4 first enables t3. After t3 is fired, the value of time attribute t in P4 is updated to t=2.07t, and the taboo table is updated as Pb, P2, and P4. The Forec(*v*) predicts that the time taken by t3 is 0.49t. When t=2.56t, the Simulation Token of P4 enable t4 to be fired. Therefore, the token of P3 is fired by t3, and the attribute of t is updated to 2.31t. After t3 is fired, at time 2.56t, t4 is enabled by the Simulation Token of p4. At this time, t4 takes 0.46t, and when the time attribute t becomes 2.77t, Simulation Token can make t4 to be fired. Therefore, through the P5 of the Token make t4 firing firstly, after t4 is fired, Simulation Token enters p6, and t is updated to 3t. When the taboo table is updated to Pb, P2, P4, P5, and Pe, the transfer rate on the path is updated 1/3. The token though the place Pb, P1, PO4, P5, and Pe, the time attribute t is updated to 3.18t, and the transfer rate on the path is 1/3.18. Then, the transfer rate of the path Pb-P2-P4-P5-Pe is large. The value of *xi* is not 0, and the main Token TOKM starts running. According to the calculation of the transfer rate on the two paths, the transfer rate on the Pb-P2-P4-P5-Pe path is large. so TOKM makes t4 firing, and it enters P2.

TOKM makes t1 enable. After t1 firing, TOKM enters p2, And the Simulation Token in p6 reaches he next transition of TOKM through the Find(ToKM). Simulation Token arriving at p6 can make t2 firing. After t2 is fired, the search path is performed according to the first simulation process. When the Simulation Token reaches p6 again, the probability on the transition output arc is updated by the Prob(*x*), and the branch is updated by Update(*x*). Since the transition t2 firing, the Simulation Token can only enter P4 , and the Optimal path be updated by the Update(x)。We find that the optimal path is still P4-P5-P6. Repeat the above process until the IF(t) function of TOKM returns to t4. The entire process of the Probabilistic Arc Petri Net ends, and the optimal path from Pb to P6 is Pb-P2-P4-P5-P6.

The above is the definition and example description of the Probabilistic Arc Petri Net.

**4、Application of Path Optimization Algorithm Based on Ant Colony Algorithm in Probability Arc Petri Nets**

The ant colony algorithm has the characteristics of dynamic optimization, which can make path adjustment for the vehicle in time, and the probabilistic arc Petri net plays an important role in business process and path optimization. Therefore, we derive the path optimization algorithm based on ant colony algorithm and probabilistic arc Petri net running. Simulating Token constantly travels between the departure point and the destination, and simultaneously feeds the optimal path to the vehicle to optimize the searched road. Simulation Token constantly travels between the departure point and the destination, and simultaneously feeds the optimal path to the vehicle to optimize the road.

**4.1、Parameters of** [**Probability**](http://www.baidu.com/link?url=WRUMQruSM8XdZWUx8YY0dZwHMfivSVv92YLdTkD1xsAjcf2Mp4rVd4XdMwdQY14RuFTs-uPI5J7dPq5rvweacTM5PekxYHm1lp58BqHdb5S) **Arc Petri Nets In Intelligent Road Traffic System**

The Probabilistic Arc Petri Net has flexibility, and can change the parameters, Prob(*x*) and Forec(*v*) of the library attribute set according to different application scenarios to accurately simulate the business process. By analyzing the operation of the Probabilistic Petri net, the path optimization algorithm based on the ant colony algorithm obtains the optimal path from the departure point to the destination. The following is the mapping of parameters in the intelligent road traffic system in the probabilistic arc Petri net, as shown in Table 4.1:

Table4.1 Symbolic Meanings in the Probability Arc Petri Nets in the Optimal Path Finding Process

|  |  |
| --- | --- |
| Symbolic | Meaning |
| Pi | road |
|  | 道路上车辆的自由速度 |
|  | 道路上的阻塞密度 |
|  | 道路上的输入流 |
|  | 道路上的输出流 |
|  | 道路的初始密度 |
| PiL | 道路长度 |
| *Vpi*(*t*) | *t*时刻道路*i*车辆的速度 |
|  | 路段*i*时间*t*的车辆密度 |
| *T* | 路口 |
| *Tb* | 开始路口 |
| *Tfok* | 经过路口 |
| *Te* | 目标路口 |
| *TOKM* | 车辆 |
| *TOMO* | 蚂蚁 |

Table 4.1 illustrates the meanings of various symbols such as place, place attribute set, and transition in the intelligent road traffic system application in the probability arc Petri net. The attributes in the place collection are used to calculate the probability that the ants choose different paths at the intersection. Three kinds of transition also represent different intersections, and two kinds of tokens represent vehicles and ants.

In describing business processes using probability arc Petri nets, developers not only define the property set, transition, and tokens, but also set prediction functions, calculation formulas for transfer rates, and branch update functions according to different business processes. When the intelligent road transportation system is searching for the optimal path, the speed of the vehicle and the length of the path are important factors that affect the speed of the vehicle. The optimal function in the intelligent road traffic system is calculated by predicting the speed of the vehicle through the prediction function simulation speed prediction algorithm in the probability arc Petri net.

**4.2Prediction Function of Path Optimization Algorithm Based on Ant Colony Algorithm**

The real-time nature of vehicle speed is an important factor for intelligent road traffic systems to find the optimal path. In order to ensure the real-time and reliability of status information transmission under the influence of dynamic speed, the vehicle speed needs to be predicted, and the vehicle is controlled by predicting the vehicle speed at a certain time in the future to prevent the vehicle from entering the congested road section. Based on the vehicle density prediction algorithm, the vehicle speed prediction algorithm is obtained. At the same time, the initial density of the vehicle is also an important parameter for calculating traffic flow. The traffic flow has its own time and space change rules. The traffic density model can be used to predict the vehicle density, and the vehicle density prediction algorithm can be used to predict the vehicle speed [60].

The three parameters that reflect the macro operation state of the traffic flow mainly include vehicle flow, vehicle speed, and vehicle density. The three satisfy the following relationships [61]:



Among them, q (x, k) is the traffic volume on the section x at time k, v (x, k) is the speed of the vehicle on the section x at time k, and ρ (x, k) is the vehicle density on the section x at time k.

The traffic flow conforms to the hydrodynamic model and satisfies the conservation law. The LWR model in the traffic flow model is an ordinary differential equation regarding vehicle density and vehicle flow [62]:

Among them, is the rate of vehicle entry on road segment i per unit time, and is the rate of vehicle exit on road segment i per unit time.（驶出率）

In a real traffic environment, the speed of a vehicle is constrained by the density of the vehicles on the road. Before the density of the road reaches saturation, the greater the density of the vehicles on the road, the lower the speed of the vehicles.In order to dynamically obtain the density of vehicles on the road, the forward road density speed model on the road is used to describe the relationship between vehicle density and vehicle speed. The forward-vehicle density-dependent vehicle speed model is [63]:

=(1 -)

Where is the free flow velocity of the vehicle, is the blocking density of the vehicle, and is the density of the vehicle at position i at time t.

The following is the relevant formula in the traffic fluid dynamics model. A vehicle speed prediction algorithm is proposed based on the road vehicle density prediction algorithm. The calculation algorithm of prediction function in probability arc Petri net is obtained according to the vehicle speed prediction algorithm. The following is the prediction function algorithm of the probability arc Petri net。

**Predictive function algorithm for probability arc Petri nets：**

Input：、、 、*CI*(*x*,*t*)、*CO*(*x*,*t*)；

Output: vehicle speed;

Step 0：Obtain the free speed of the vehicle and the blocking density of the vehicle in the collection of attributes in the place;

Step 1：Get the traffic flow on the road at t = 0, and initialize the initial density in the collection of attributes in the place;

Step 2：Initialize the values of and in the collection of attributes in the place according to the traffic flow;

Step 3: Bring , and nto formula (4.3) to obtain *Vpi*(0)

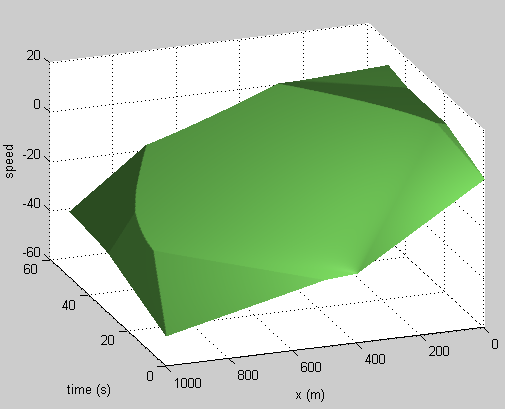
Step4: Bring Vpi(0) into formula (4.2) to calculate the vehicle density at the next time t = 1;

Step 5：Bring the vehicle density into formula (4.3) to obtain the speed of the vehicle at the next time step;

According to the prediction function algorithm of the probability arc Petri net, the vehicle speed on each road section is predicted, and it is used to model the intelligent road traffic system by the probability arc Petri net. Figure 4.3 shows the relationship between speed, time and distance.

**4.4.1 Definition of Path Optimization Algorithm**

Definition 4.1: The formula for the transfer rate of ants from city i to city j in the path optimization algorithm in the following:

Fig 4.3 Speed, Time, Distance Diagram

The prediction function algorithm of the probability arc Petri net, the vehicle speed, time, and distance map obtained by MATLAB, the road is divided into different lengths and he unit is meter, then [0-400], [400-500], [ 500-1000] vehicle density is initialized to 0.08veh / m, 0.01veh / m, 0.03veh / m. Free speed is 30m / s, blocking density is 0.1veh / m, vehicle input flow at [0-20s], [20s-40s], [40s-50s] is 0.4veh / s, 0.01veh / s, 0.2veh / s. The speed, time and distance graphs are obtained through MATLAB.

**4.4 Correlation Definition of Path Optimization Algorithm Based on Ant Colony Algorithm**

Many papers have proved that ant colony algorithm is effective in path optimization, but to dynamically find the least time-consuming path in the path, it is still necessary to improve the ant colony algorithm and add influence to the original ant colony algorithm Dynamic factors of vehicle operation [64].



Among them, *τij*(*t*) represents the pheromone concentration left at the intersection *i* to the intersection *j* at time t, *ηij*(*t*) is the heuristic function of the intersection, and *Vpi*(*t*) is the average speed on the section *i*. *allowekk*( *k* = 1, 2, 3, …, n )indicates the section where ants are allowed to choose next, *tabukk*(*k* = 1, 2, 3, …, *n*) indicates the section where ants have traveled, and *tabuk* will follow the ants The search process *tabuk* collection will also increase.

**Definition 4.2:** Ants are divided into two categories, the first is the common ant *TOKO*; the other is the main ant *TOKM*, and the number of main ants is Num (*TOKM*) = 1; The destination of *TOKM* is *Pe*, *Tf* is an ordinary intersection; IF(t) is an intersection determination function used to determine the type of intersection. Each *TOKO* contains an IF (t), and IF(*t*) determines whether the return intersection is *Te*.

**Definition 4.2:** When IF(*TOKO*)∈*Te*, *TOKO* releases the pheromone，and update the transition probability on this arc by Update(x). And uses the Find(*TOKM*) function to find the next transition of *TOKM*. After executing Find(*TOKM*), it returns *tf* and *TOKM* back to transition *tf* .

**Definition 4.3:** Update(*t*) continuously updates the optimal path according to the pheromone on each road section. When IF(*TOKM*) = *te*, the path update function Update(*t*) must be executed to return the path with the highest pheromone to *TOKM*.

**Definition 4.5:** The following is the pheromone update function:

(4.6)

Where *ρ* is the volatility coefficient, the pheromone on each path is updated after the ant has passed through a cycle After *n* times,.



(4.7)

indicates the pheromone left by ant *K* on path *i* to path *j*, and indicates the total amount of pheromone left by all ants when they pass the path in this process.



(4.8)

Where *Q* is a constant representing the total amount of pheromone released by the ant on this path, and represents the time it takes for the ant to walk on this path. At the initial time ,

**4.4.1 Path optimization algorithm based on ant colony algorithm**

According to the process of the path optimization algorithm based on ant colony algorithm, a path optimization algorithm based on ant colony algorithm is obtained.

The following is a path optimization algorithm based on ant colony algorithm：

Input：*Nmax*、Num(*TOMO*)、Pe、*CI*(*x*,*t*)、*CO*(*x*,*t*)、*ρ*(*x*,0)、*Vf*、*ρj、L*(*x*)、α、β、ρ;

Output: Path optimization algorithm based on ant colony algorithm;

Step 0：After the vehicle sends a destination request, the initial test parameters TOMO.t = 0 and the number of cycles Nc = 0, and the maximum number of cycles Nmax = 100 and the number of ants Num (*TOMO*) = 10 are set. At the same time, put the ant Token in the Pe, set the initial concentration of pheromone on each path = 0, and initialize the variables in the property set of the place

Step 1: IF(*TOKO*) = *tf*, and is the same on each arc, then *TOKO* will randomly enter each unvisited place , predict the speed of the vehicle entering the road segment according to the prediction function, and calculate the time spent on the road segment, and update the value of t on TOKO;

Step2: Modify the value in the taboo table, and move the node just walked into the ant taboo table；

Step 3: When IF(TOKO) ∈Te, update the pheromone on the path by the pheromone update function ;

Step 4: Number of cycles ;

Step 5：IF(*TOKM*) ∈ *Tf* ;

Step 6: Update (t) updates the optimal path;

Step 7: Update the values of the variables in each place's property collection;

Step 8: Find(*TOKM*), the ant backs to Find(*TOKM*) and repeats steps (2), (3), (4);

Step 9: IF(*TOKM*) = *pe*, end the algorithm

This algorithm is based on the ant colony algorithm to predict the speed of the vehicle at a certain time in the future through a prediction function. The pheromone concentration of the road is updated according to the time when the ant passes different paths in the future. Before the vehicle reaches each intersection, the optimal path finding algorithm must be performed again. According to the latest pheromone concentration on each path, the optimal path is fed back again. Optimize the optimal path while continuously correcting the optimal path.

**5．Modeling and Analysis of Road Network Based on Path Optimization Algorithm by Probability Arc Petri Net**

**5.1 Traffic Road Description and Speed Prediction Simulation**

Figure 4.5 shows the network topology of a road section in a certain area. There are nine intersections in the road topology. There are branch intersections at intersection 1, intersection 2, and intersection 3. There are two overlapping roads at intersection 2 and intersection 3 to intersection 5. It is not well represented in the network topology. There are 11 roads in the topological structure of the road, and an optimal path search algorithm based on the ant colony algorithm is used to find the optimal path from road 1 to road 11.

Fig 4.5 Topological Structure of a Section of a Certain Area

The initial information in the road network is shown in Table 4.1. *CI*(*x*,*t*) is the inbound traffic volume of each link, *CO*(*x*,*t*) is the outbound traffic volume of each link, *ρ*(*x*,0) is the initial density on the link, and *Vf* is the Blocking speed, *ρj* is the blocking density on the link, and *L*(*x*) is the length of the path on the link. The unit of input and output flow is veh / h, the unit of initial density and block density is veh / km, the free speed is km / h, and the unit of each road length is km. Table 4.2 gives the initial parameters of some road sections:

Table 4.2 Initial Parameters of Each Section

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| *NO* | *CI*(*x*,*t*) | *CO*(*x*,*t*) | *ρ*(*x*,0) | *Vf* | *ρj* | *L*(*x*) |
| 1 | 110 | 110 | 50 | 100 | 100 | 2.0 |
| 2 | 120 | 105 | 45 | 100 | 100 | 1.6 |
| 3 | 125 | 113 | 53 | 100 | 100 | 1.8 |
| 4 | 114 | 116 | 46 | 100 | 100 | 1.9 |
| 5 | 100 | 120 | 55 | 100 | 100 | 2.2 |
| 6 | 124 | 132 | 48 | 100 | 100 | 1.7 |
| 7 | 110 | 110 | 50 | 100 | 100 | 2.9 |
| 8 | 132 | 130 | 47 | 100 | 100 | 1.6 |
| 9 | 127 | 129 | 65 | 100 | 100 | 2.1 |
| 10 | 142 | 140 | 56 | 100 | 100 | 2.0 |
| 11 | 127 | 120 | 64 | 100 | 100 | 1.9 |

**5.2 Modeling and Analysis of Probability Arc Petri Nets**

Probabilistic arc Petri nets are established according to the topology given in Figure 4.5 and the initial parameters of each section in Table 4.2. The probabilistic arc Petri net is used to describe the optimal path finding process dynamically, and the optimal path is obtained.

Figure 5.2 is a probabilistic arc Petri net modelling an intelligent road system based on ant colony algorithm for path optimization algorithm. The attributes used in the prediction function algorithm include [*Pici*，*Pico*，*Piρ0*，*PiL*] in the place property set, which respectively represent the input flow, output flow, initial density, and length of the road. Assume that the blocking density on each road section is 100veh / km, and the free speed on each road section is 100km / h. The unit of *Pici* and Pico is veh / h, the unit of *Piρ0* is veh / h, and the unit of *PiL* is km. And use Tina to simulate a road network with a path optimization algorithm based on ant colony algorithm, and find the optimal path through simulation.

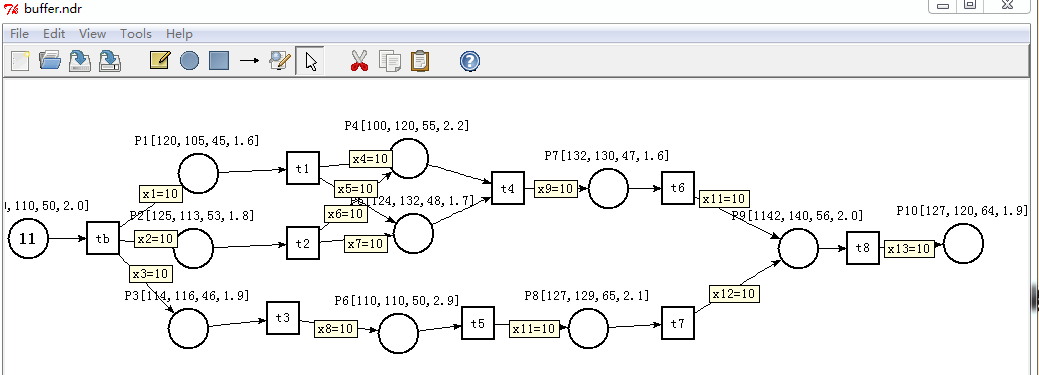
Fig 5.2 Modeling of Intelligent Road System Based on Path optimization algorithm based on ant colony algorithm with Probability Petri Nets

Fig5.2 is a probabilistic arc Petri net for modeling intelligent road traffic systems based on ant colony algorithm for path optimization algorithm. The speed of the vehicle is predicted according to the prediction function of the probability arc Petri net. *λi* in Figure 5.2 is the probability of which road to choose after the vehicle enters the intersection. The ant colony parameters are set to α =0.5、*β* = 0.5、*ρ*=0.6. According to the recommendation of reference [24], the initial token number in the place *Pb* is set to 10, the initial pheromone of *λ*1~*λ*13 is 10, and the initial implementation degrees of the places *P1* to *P10* are respectively It is 34, 26, 28, 20, 30, 17, 33, 16, 22, 24.

**5.1.2 Process analysis**

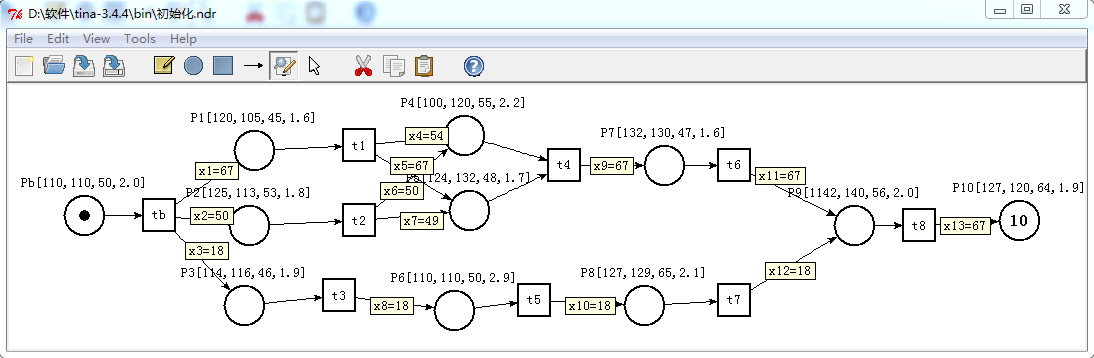
Figure 5.3 is a probability arc Petri net that updates the value of AA on each road after a path search according to each initial value. And the found optimal path is fed back to the *TOKM* through the information feedback mechanism.

Fig 5.3 Probability Arc Petri Nets Simulation for The First Time Path Finding

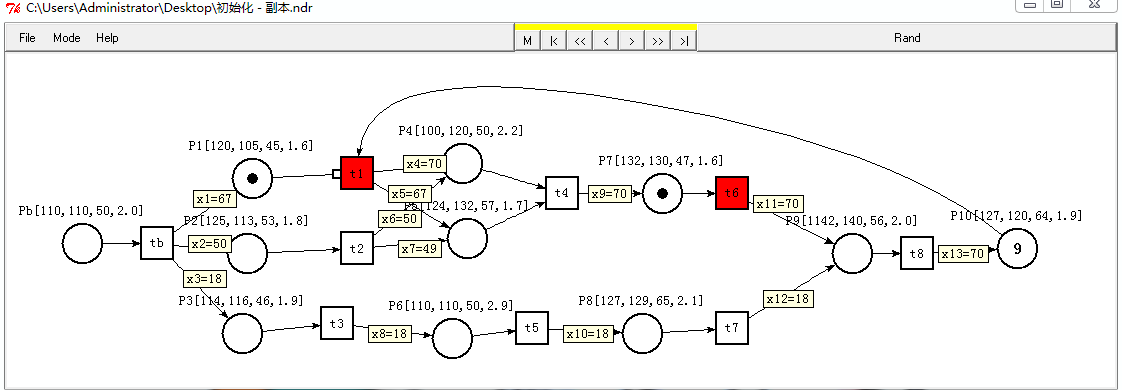
After updating the value of *λ* on each path, the optimal path obtained is *Pe*→*P*1→*P*5→*P*7→*P*9→*P*10, and the optimal path takes 0.185h. The optimal path is fed back to *TOKM* through the information feedback mechanism in the probability arc Petri net. *TOKM* enables the transition *tb*. When the transition tb is fired, *TOKM* enters the place P1, and the Find (t) function on *TOKO* is used to find the place where *TOKM* is located. The place P10 forms an arc pointing to the transition t1, and again finds the optimal path from P1 to P10.

Fig 5.4 After TOKM Makes Transition tb Firing

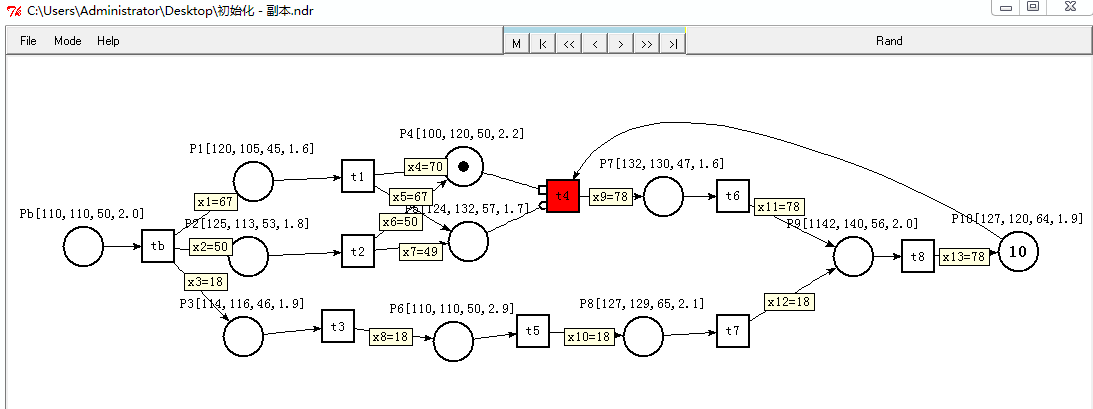
In the operation of the probability arc Petri net, the *TOKM* fired transition Tb took 0.029h. The probability arc Petri net obtains the information of each road from road 2 to road 11, and *TOKO* searches for the optimal path from *P1* to *P10* again, and updates the value of *λ* on each arc. According to Figure 5.4, it can be seen that due to changes in vehicle density, the vehicle density and vehicle speed on each road section are updated, and the variables in the attribute set of the place are dynamically updated. After *TOKO* searches for the optimal route once, the value of *λ* on the road *P*1→*P*4→ *P*7→*P*9→ *P*10 is the largest, so the optimal path is updated to *P*1→*P*4→ *P*7→*P*9→ *P*10 by the road update function.

Fig 5.5 After TOKM Makes Transition t1 Firing

The path updated by the update function, the value of *λ* on *P*1→*P*4→ *P*7→*P*9→ *P*10 is larger than other roads, so *TOKM* fires transition *t1* and enters the *P4*. After *TOKM* enters the place *P4*, even if *P4* to *P10* does not choose a branch, *TOKO* will still find the optimal path again until the IF (t) function in *TOKM* returns *t8*. After the transition *t1* is fired, the arc at the place *P10* pointing to the transition *t1* disappears. When *TOKO* all enters the target place again, an arc is formed from the place *P10* to the transition *t4*.

Fig 5.6 End of Optimal Path Finding

The firing of other transitions is similar and will not be repeated here. Figure 5.6 shows the results of the optimal path from *Pb* to *P10*. The optimal path is P *P*1→*P*4→ *P*7→*P*9→ *P*10. *TOKM* takes a total of 0.183h. After *TOKM* enters *P10*, the probability arc Petri net operation ends.

**5.2 Property Analysis**

Reachability is the basic property of Petri nets. Reachability of Petri nets is the basis of analyzing other dynamic properties. The shortest path from Pb to P10 can be obtained through the reachability graph. According to Figure 3.2, M is the reachable state: M=([num(*TOK*), Type(*TOK*)], *Poiv*, *Poil*). Where num(*TOK*) represents the number of tokens in the place, Type(*TOK*) is the type of token, *POiv* is the speed of the place at this time, and *POil* is the path length of the place. Due to space limitations, we only list the reachability status of *P*1→ *P10* and *P*5→ *P*10.

Table 5.2 Optimal Path Finding of *P*1→ *P10*

|  |  |
| --- | --- |
| S19(tb) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45, 2.2), ([0],52, 1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([10,TOKo],36.5,1.9)) |
| S20(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([1,TOKO],45.7,2.2), ([0],52, 1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([9,TOKo],36.5,1.9) ) |
| S21(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7), ([0],50,2.9), ([1,TOKO],53.8,1.6), ([0],45,2.1), ([0],44,2.0), ([9,TOKo],36.5,1.9) ) |
| S22(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7), ([0],50,2.9),([0],53.8,1.6),([0],45,2.1),([1,TOKO],44.4,2.0), ([9,TOKo],36.5, 1.9) ) |
| S23(t8) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7), ([0],50,2.9), ([0],53.8,1.6), ([0],45,2.1), ([0],44,2.0), ([10,TOKo],36.7, 1.9) ) |
| S24(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([3,TOKO],45.7,2.2), ([0],52, 1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([7,TOKo],36.5,1.9) ) |
| S25(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7), ([0],50,2.9), ([3,TOKO],53.8,1.6), ([0],45,2.1), ([0],44,2.0), ([7,TOKo],36.5,1.9) ) |
| S26(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7),([0],50,2.9),([0],53.8,1.6),([0],45,2.1), ([3,TOKO],44.4,2.0), ([7,TOKo],36.5, 1.9) ) |
| S27(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([5,TOKO],45.7,2.2), ([0],52, 1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([5,TOKo],36.5,1.9) ) |
| S28(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7), ([0],50,2.9), ([5,TOKO],53.8,1.6), ([0],45,2.1), ([0],44,2.0), ([5,TOKo],36.5,1.9) ) |
| S29(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7),([0],50,2.9),([0],53.8,1.6),([0],45,2.1), ([5,TOKO],44.4,2.0), ([5,TOKo],36.5, 1.9) ) |
| S30(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([7,TOKO],45.7,2.2), ([0],52, 1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([3,TOKo],36.5,1.9) ) |
| S31(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7), ([0],50,2.9), ([7,TOKO],53.8,1.6), ([0],45,2.1), ([0],44,2.0), ([3,TOKo],36.5,1.9) ) |
| S32(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7),([0],50,2.9),([0],53.8,1.6),([0],45,2.1), ([7,TOKO],44.4,2.0), ([3,TOKo],36.5, 1.9) ) |
| S33(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([1,TOKO], 52,1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([9,TOKo],36.5,1.9) ) |
| S34(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([1,TOKO],53.9.,1.6), ([0],45,2.1), ([0],44,2.0), ([9,TOKo],36.5,1.9) ) |

|  |  |
| --- | --- |
| S35(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([0],53.,1.6), ([0],45,2.1), ([1,TOKO],44.4,2.0), ([9,TOKo],36.5,1.9) ) |
| S36(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([5,TOKO], 52,1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([5,TOKo],36.5,1.9) ) |
| S37(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([5,TOKO],53.9.,1.6), ([0],45,2.1), ([0],44,2.0), ([5,TOKo],36.5,1.9) ) |
| S38(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([0],53.,1.6), ([0],45,2.1), ([5,TOKO],44.4,2.0), ([5,TOKo],36.5,1.9) ) |
| S39(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([8,TOKO], 52,1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([2,TOKo],36.5,1.9) ) |
| S40(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([8,TOKO],53.9.,1.6), ([0],45,2.1), ([0],44,2.0), ([2,TOKo],36.5,1.9) ) |
| S41(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([0],53.,1.6), ([0],45,2.1), ([8,TOKO],44.4,2.0), ([2,TOKo],36.5,1.9) ) |
| S42(t1) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6),([0],47,1.8),([0],54,1.9), ([0],45,2.2), ([10, TOKO], 52,1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([0],44,2.0), ([0],36.5,1.9) ) |
| S43(t4) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([10,TOKO],53.9.,1.6), ([0],45,2.1), ([0],44,2.0), ([0],36.5,1.9) ) |
| S44(t6) | (([0],50,2.0), ([(1,TOKM)],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52,1.7), ([0],50,2.9), ([0],53,1.6), ([0],45,2.1), ([10,TOKO],44.4,2.0), ([0],36.5,1.9) ) |

Table 5.3 Running of TOKM on *P*5→ *P*10

|  |  |
| --- | --- |
| S45(t1) | (([0],50,2.0), ([0],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([(1,TOKM)],52.5, 1.7), ([0],50,2.9), ([0],53.,1.6), ([0],45,2.1), ([0],44.4,2.0), ([10, TOKO], 36,1.9)) |
| S46(t4) | (([0],50,2.0), ([0],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52, 1.7), ([0],50,2.9), ([(1,TOKM)],53.8,1.6), ([0],45,2.1), ([0],44.4,2.0), ([10, TOKO],36,1.9)) |
| S47(t6) | (([0],50,2.0), ([0],65.3,1.6), ([0],47,1.8), ([0],54,1.9), ([0],45,2.2), ([0],52.5, 1.7), ([0],50,2.9), ([(0)],53,1.6), ([0],46,2.1), ([1,TOKM],44.4,2.0), ([10 , TOKO],36,1.9)) |
| S48(t8) | (([0],50,2.0),([0],65.3,1.6),([0],47,1.8),([0],54,1.9),([0],45,2.2),([0],52.5,1.7),([0],50, 2.9), ([(0)],53,1.6), ([0],46,2.1), ([0],44.4,2.0), ([(1,TOKM),(10, TOKO)], 36, 1.9)) |

Figure 5.7 is based on the reachability graph algorithm of the probability arc Petri net. According to Figure 5.7, the reachable states S0, S18, S19, S23, S45, S46, S47, and S48 of TOKM when the transition is triggered are obtained. According to the probabilistic arc Petri net based on the ant colony algorithm of the path optimization algorithm to model and analyze the intelligent road traffic system, the path that takes the shortest time from path 1 to path 1→2→5→8→10→ 11. Through the dynamic operation of the probability arc Petri net, the prediction time under this path is 0.183h. This time is obtained by the operation of the probability arc Petri net based on the prediction of the vehicle speed by the prediction function in the probability arc Petri net and the vehicle path optimization algorithm.

Fig 5.7 Reachable Graph of Intelligent Road Modeling Based on Ant Colony Algorithm for Optimal Path Finding

**5.3 Algorithm Comparison**

Figure 5.4 is the prediction time obtained by different algorithms for finding the optimal path for the road described in Figure 4.5 when the initial parameters of the variables are the same. Figure 5.8 shows the optimal path pair running time obtained from the improved Dijkstra algorithm proposed in [64] and the ant colony algorithm and the ant colony-based path optimization algorithm proposed in [48]. Using Matlab to simulate the given traffic network, the time spent on each road is obtained, and the time taken by the shortest path is 0.18h. Figure 5.8 is a comparison chart of prediction time and running time under different algorithms:

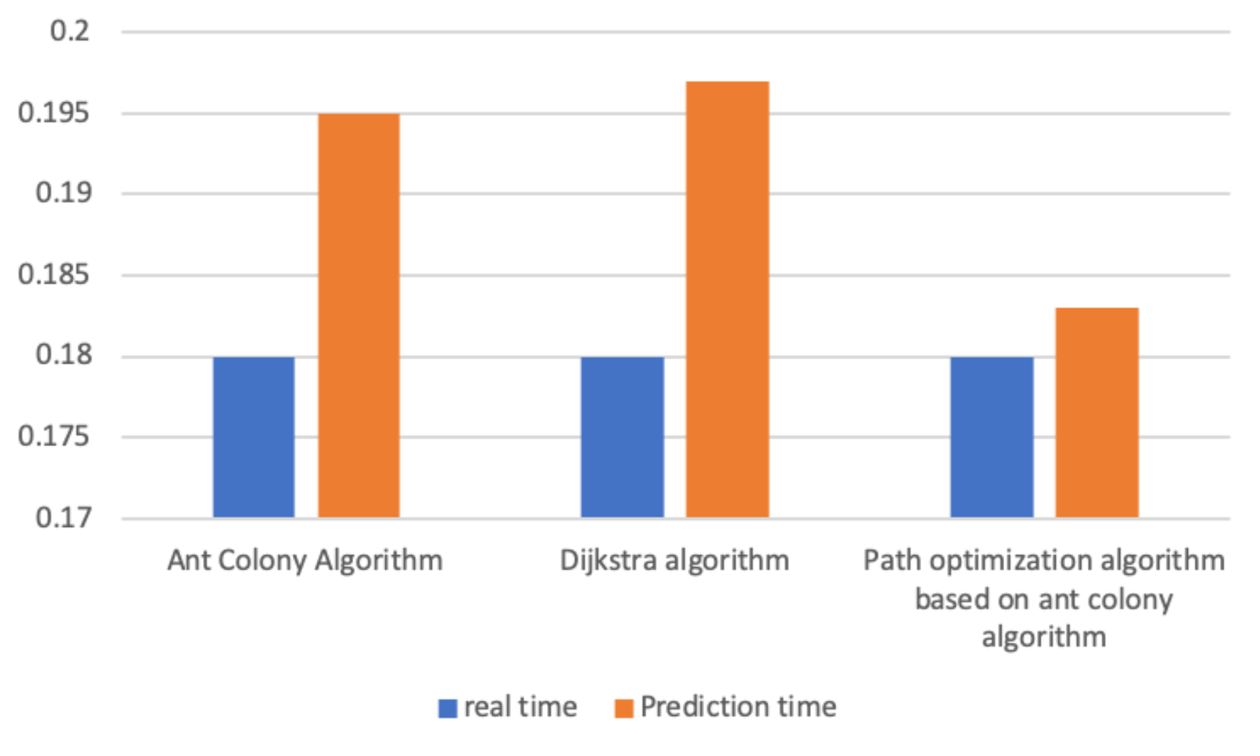


Fig 5.8 Prediction Time and Vehicle Running Time under Different Algorithms

In the histogram, blue indicates the time of the optimal path obtained from the simulation of the traffic network through MATLAB, and red indicates the time of the optimal path prediction under different algorithms. The optimal path obtained by the improved Dijkstra algorithm and the path optimization algorithm based on ant colony algorithm is the same road, but the predicted time of the improved Dijkstra algorithm for this path is 0.197, and the predicted time by the path optimization algorithm based on ant colony algorithm is 0.183. The prediction time of the optimal path sought by the ant colony algorithm is 0.189h. Compared with the other two algorithms, the difference between the predicted time of the path optimization algorithm based on ant colony algorithm and the time of vehicle simulation is the smallest. Although the optimal path obtained by the path optimization algorithm based on the ant colony algorithm is not the shortest, it can take into account the factors of vehicle flow, vehicle speed, and path length on the road to find a road that takes the shortest time for the vehicle. And can successfully avoid the sudden congestion of the road caused by various reasons.

**6 Conclusions and Prospects**