mathematics

MDPI

Article

A Discrete-Event Mathematical Model for Resource Allocation Optimization: A Case Study of Vehicle Scheduling in a Signal-Free Intersection

Yunfeng Hou [1], Yue Mao [2], Yanmei Zhang [3], Qingdu Li [1], Yunfeng Ji [1] and Wei Li [4,5,*]

[1] Institute of Machine Intelligence, University of Shanghai for Science and Technology, Shanghai 200093, China
[2] School of Health Science and Engineering, University of Shanghai for Science and Technology, Shanghai 200093, China
[3] Department of International Trade, College of Commerce, Jeonbuk National University, Jeonju 54896, Korea
[4] Postdoctoral Station of Applied Economics, Fudan University, Shanghai 200433, China
[5] School of Finance, Shanghai Lixin University of Accounting and Finance, Shanghai 201209, China
* Correspondence: li_wei@lixin.edu.cn; Tel.: +86-189-0178-3118

Abstract: In industrial applications, many systems present serious productivity problems due to limited resources. Generally, the dynamics of resource allocation are inherently discrete-event driven, such as the buffer allocation in production line systems. In this paper, we develop a discrete-event mathematical model for resource allocation optimization. In this work, we consider two crucial optimization objectives, e.g., deadlock-free and efficiency, that originate from the customer's actual requirements. The main aim is to develop a resource allocation scheme for fulfilling the production process (without deadlock) while ensuring that the cost of the process is minimized. As a case study, we consider the vehicle scheduling problem in a signal-free intersection. The intersection is divided into several disjoint spatial traffic resources, and vehicles need to occupy different traffic resources for passing through the intersection. Thus, the traffic control problem at the signal-free intersection is transformed into a scheduling problem with limited resource constraints. An online control approach is developed to schedule vehicles to go through the intersection safely and efficiently by optimizing the resource allocation order. Simulation results demonstrate the efficiency and robustness of the proposed model and optimization approach.

Keywords: resource allocation; discrete-event model; optimization algorithms; signal-free intersection; vehicle scheduling

MSC: 93

Citation: Hou, Y.; Mao, Y.; Zhang, Y.; Li, Q.; Ji, Y.; Li, W. A Discrete-Event Mathematical Model for Resource Allocation Optimization: A Case Study of Vehicle Scheduling in a Signal-Free Intersection. *Mathematics* **2022**, *10*, 4183. https://doi.org/10.3390/math10224183

Received: 2 October 2022
Accepted: 3 November 2022
Published: 9 November 2022

1. Introduction

A resource allocation system (RAS) consists of a set of concurrently executing processes that use a group of resources to accomplish a given task [1]. A system resource can be reused, but it is in limited supply in the sense that it can be occupied by up to n processes at the same time, where n is its maximum capacity. RASs manifest in many technological systems that involve resource sharing, such as facility planning [2], job scheduling [3], traffic management [4,5], computer system [6], and buffer allocation [7]. For instance, in a typical computer system, jobs, tasks, or transactions are the customers competing for the attention of servers such as various processors, such as the CPU, or peripheral devices (e.g., printers, disks). It is often convenient to represent such a system through an RAS.

The study of RASs can date back to [8–11], where an efficient operating system was developed. For the RASs, liveness or deadlock-freeness is an essential control requirement for their automatic operation [12–14]. There exists a deadlock in an RAS if there exists a group of concurrently executing processes that block each other such that each of them

requires, for its further continuation, some resource(s) currently occupied by another process in this group. Thus, to ensure deadlock-free operation, the underlying resource allocation must be carefully considered. Furthermore, for operational flexibility of the RASs, one would like to avoid deadlocks in the least restrictive manner. Recently, refs. [12–15] developed a deadlock avoidance policy (DAP) that is least restrictive in the sense that no other optimal DAP can generate a larger deadlock-free system behavior. Nevertheless, a DAP only guarantees a task can be finished without any deadlock, but it does not guarantee the task can be finished in a specific route. However, in many practical RASs, such as traffic systems, it is required to "drive" the system to the goal state in a trajectory with a minimum time cost. In addition, the computation of the optimal DAP is proven to be an NP-hard problem [16]. Recently, a sequential RAS structure that admits polynomial-time optimal DAP has been widely investigated [12,14,15,17,18]. In a sequential RAS, every process, during its execution, need to occupy and release in a predefined order of resources. We consider the sequential RAS in this paper.

In contrast to [12–15,17,18], we develop a control mechanism to drive the system to the goal state (marked state) in a specific trajectory. Specifically, we develop an optimal control mechanism to actively achieve a given task in a deadlock-free manner while minimizing the cost of the resource allocation process. The sequential RAS is modeled as an automaton, which has been proven to be a powerful tool for the design, analysis, and control of discrete-event systems [17,19–22]. In the automaton, the occupation and release of a resource are both viewed as an "event". To accomplish a given task, the resource allocation controller not only disables events but also enforces events. To determine which event should be disabled or enforced, an online resource allocation strategy, whose objective is to minimize the cost of the event sequence to be executed (the process of resource allocation and release), is developed. The objective is accomplished by model predictive control. More accurately, by looking l steps forward from the current state, we select a deadlock-free event sequence with the minimum cost, where l is the prediction depth. After each new state updating, we calculate such an optimal event sequence and execute the first event of it. The above process is repeatedly performed until the given task is achieved.

Note that in traditional supervisory control of discrete-event systems [23–32], a supervisory controller, which is referred to as a supervisor, is desired to enable a maximum allowable set of controllable events at any instant to ensure the controlled system is within the desired specification language. However, an RAS requires us to drive the system to the goal state at a minimum cost. It is about how to implement the system. However, a supervisor fails to drive a system from one state to another state following a specific trajectory (with the minimum cost). Thus, in contrast to traditional supervisory control, we consider a "positive" controller that selects at most one controllable event to be enforced and disables all the remaining controllable events at each instant, as described in the last paragraph.

As an application, we apply the proposed modeling and optimizing method to solve the vehicle scheduling problem in a signal-free traffic intersection [33–38]. It is worth noting that approaches for the intersection management problem can be classified into two categories: the signal-based approach [4,39–41] and the signal-free approach [5,33–38,42–52]. The signal-based approach aims at minimizing queue lengths by optimizing the duration of each signal phase, and the signal-free approach aims at minimizing the intersection travel time by optimizing the passing order of vehicles. The signal-based approach has better adaptiveness and reliability, whereas the signal-free approach is more efficient and economical, especially when the infrastructure cost is a concern. In this paper, we focus on the signal-free approach. Given the source road and the destination road, the trajectory of a vehicle is fixed. We define a resource as the crossing location of two trajectories of vehicles. The intersection should be divided into several resources in terms of granularity. For safety, each resource can be occupied by one vehicle at a time. A process is defined as the process of a vehicle passing through an intersection. Then, the intersection management system is "abstracted" as an RAS modeled as an automaton. In the automaton, vehicle's movement between adjacent resources is viewed as an "event", without considering the vehicle's kinematics. Using

the proposed optimization techniques, we present an online vehicular (high-level) control strategy that minimizes the travel time of vehicles in the intersection. The strategy can be actively achieved by disabling and enforcing events. The control decisions are made on-the-fly, and a vehicle can be immediately considered by the controller when it arrives at the intersection. Simulation results demonstrate the expressiveness of the proposed model and the effectiveness of the proposed algorithm.

The contributions of this paper are twofold. First, in this paper, we introduce a novel control mechanism for the RAS. A controller is used to enforce at most one controllable event and disable all the remaining controllable events at any instant. An optimization-based approach is developed to design a controller: starting from a given state, it drives the system to the goal state over a trajectory with the minimum time cost. This is in contrast to the DAP developed for the RAS in [12–15], where the controller only enables a maximum allowable set of events at any instant, and no specific selection for executing an event is made. Second, we show how a signal-free intersection can be modeled as an RAS and demonstrate how the proposed approach can be applied to schedule vehicles to go through the intersection. The proposed method for intersection management differs from the existing works in the following sense.

1. In this paper, we assume that different vehicles can travel in the intersection at the same time as long as they do not conflict with each other, whereas in [42–45], only one vehicle is allowed to go through the intersection at a time. Thus, the spatial spacing of the intersection is well utilized in this paper;
2. Different from [46–48], our control scheme does not involve a complex nonlinear programming (NLP) problem with high-dimensional collision avoidance constraints, as vehicle travel safety has been guaranteed by the principle that a spatial resource can be occupied by one vehicle at a time;
3. The passing order calculated in this paper is optimal with respect to all the vehicles approaching and traveling in the intersection. This is in contrast to [5,33–35,37,38,51,52], where the passing order of vehicles is calculated for only a batch of vehicles at a time. Before the first batch of vehicles finish their journey in the intersection, the second batch of vehicles must wait at the intersection. This may damage traffic efficiency due to the lack of a global view.

The rest of this paper is organized as follows. In Section 2, some preliminary concepts and the mathematical model are introduced. In Section 3, we formally formulate the problem, and a resource allocation optimization algorithm is provided. In Section 4, we discuss how the proposed model and optimization algorithm can be applied to schedule vehicles in a signal-free intersection. In Section 5, the simulation results are obtained. In Section 6, we further discuss how the proposed approach can be applied to schedule robot in a warehouse. Finally, in Section 7, this paper is concluded.

2. Mathematical Model

2.1. Preliminaries

In this section, the resource allocation process is "abstracted" as a discrete-event system modeled as an automaton. In the discrete-event systems, the occupation or release of a resource is viewed as an "event".

Formally, the automaton is denoted by a six-tuple $G = (Q, \Sigma, f, \Gamma, q_0, Q_m)$, where Q is the finite set of states, Σ is the finite set of events, $f : Q \times \Sigma \to Q$ is the transition function, $\Gamma : Q \to 2^{\Sigma}$ is the active function, q_0 is the initial state, and $Q_m \subseteq Q$ is the set of marked states (highlighted by double circles). For any $q \in Q$, $\Gamma(q) = \{\sigma \in \Sigma : f(q, \sigma)!\}$, where "!" means "is defined". Σ^* is the set of strings composed by events in Σ (including the empty string ε). Then, f can be iteratively extended to $Q \times \Sigma^*$ in the way as, $f(q_0, \varepsilon) = q_0$, and for any $s \in \Sigma^*$ and any $\sigma \in \Sigma$, $f(q_0, s\sigma) = f(f(q_0, s), \sigma)$. The language generated by G is denoted by $\mathcal{L}(G) = \{s \in \Sigma^* : f(q_0, s)!\}$. The marked language generated by G is denoted by $\mathcal{L}_m(G) = \{s \in \mathcal{L}(G) : f(q_0, s) \in Q_m\}$. An automaton G is said to be

accessible if any states in Q can be reached from the initial state q_0 via a string $s \in \mathcal{L}(G)$, i.e., $[(\forall q \in Q)\exists s \in \mathcal{L}(G)]f(q_0, s) = q$. The accessible part of G is denoted by $Ac(G)$.

In this paper, the given control objective is achieved using a controller or supervisor, which dynamically disables or enforces event occurrences of the system based on previous events that have occurred or been observed. Specifically, the event set Σ is partitioned into the set of uncontrollable events Σ_{uc} and the set of controllable events Σ_c, i.e., $\Sigma = \Sigma_{uc} \cup \Sigma_c$. For all uncontrollable events (the elapsing of a unit of time, for example), the controller can never disable its occurrence. We also assume that some events in Σ are enforceable in the sense that it can be enforced by preempting an uncontrollable event. We denote by $\Sigma_f \subseteq \Sigma$ the set of all the enforceable events. Overall, a controller can adopt the following two different control behaviors to achieve the control objective.

- Disablement: During the resource allocation process, some controllable events may be disabled by the controller and cannot occur;
- Enforcement: At the same time, some enforceable events can be enforced to execute by the controller.

2.2. Discrete-Event Model

Next, we model the resource allocation process. Suppose that there are h resources, denoted by Resources $1, \ldots, h$. We also suppose that there are d processes, denoted by Processes $1, \ldots, d$. We denote $\mathcal{R} = \{1, \ldots, h\}$ by the set of resources, where i means Resource i, and $\mathcal{P} = \{1, \ldots, d\}$ by the set of processes, where i means Process i. Note that both h and d can be determined by the physical system, and they have nothing to do with each other. Each process needs to use resources in a predetermined order to complete a specific task. Meanwhile, Resource i can be held by at most m_i processes simultaneously for all $i \in \mathcal{R}$. The resource allocation process can be briefly formulated as follows. When a process wants to use a resource, it first sends a request to the centralized resource manager (CRM). The CRM receives requests from agents, decides whether the resource is available, and sends instructions back to it.

To model the resource allocation process, we must first model the resources and the processes. Specifically, given a process $i \in \mathcal{P}$, we assume that it needs to use resources in the order of $x_1 \to x_2 \to \cdots \to x_{n-1} \to x_n$ to complete a given task. As shown in Figure 1a, we build an automaton P_i to model the dynamics of the working status of Process $i \in \mathcal{P}$. For clarity, we interpret the events appeared in P_i as follows. For any $i \in \mathcal{P}$ and any $j \in \mathcal{R}$, e^i_j means that Process i sends a request to the CRM for using Resource j; σ^i_j means that Process i receives an acknowledgement of the usage of Resource j, and then occupies Resource j; $e^i_{j\to k}$ means that Process i finishes the usage of Resource j and sends a request to the CRM for further using Resource k; $\sigma^i_{j\to k}$ means that Process i has occupied Resource k after the usage of Resource j. π^i_j means that Process i releases Resource j. Since the CRM cannot prevent a process from sending a request, e^i_j and $e^i_{j\to k}$ are all uncontrollable for $i \in \mathcal{P}$ and $j, k \in \mathcal{R}$. All the remaining events are controllable.

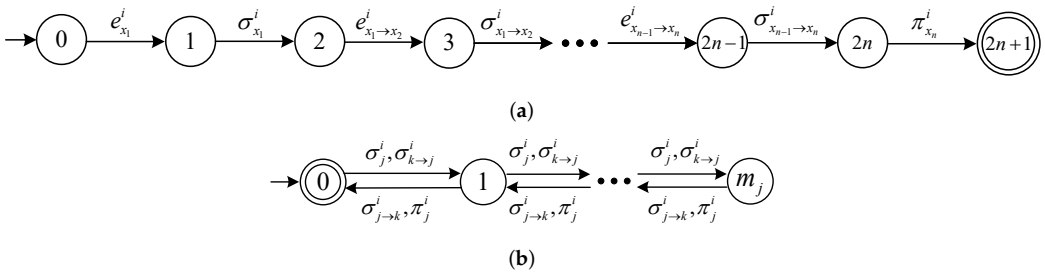

Figure 1. Automata P_i and R_j; (**a**) Automaton P_i for Process i; (**b**) Automaton R_j for Resource j.

In State 0 of P_i, Process i is placed in standby. To complete a given task, Process i should first apply for the usage of Resource x_1. Thus, when Process i starts to work, it

sends a Resource x_1 occupation request to the CRM. Upon the event occurrence of $e^i_{x_1}$, P_i moves from State 0 to State 1. If Process i receives an acknowledgement message from the CRM ($\sigma^i_{x_1}$ occurs), it immediately occupies Resource x_1, and P_i moves from State 1 to State 2. After that, when Process i finishes using Resource x_1, it needs to further apply for the use of Resource x_2. Upon the occurrence of $e^i_{x_1 \to x_2}$, P_i moves from State 2 to State 3. The above process is repeated until $\pi^i_{x_n}$ occurs in State $2n$ (Process i finishes the usage of Resource x_n).

Next, as shown in Figure 1b, we build an automaton R_j to model the dynamics of the working status of Resource $j \in \mathcal{R}$. In State 0 of R_j, Resource j is idle. By definition, when σ^i_j and $\sigma^i_{k \to j}$ for $i \in \mathcal{P}$ and $k \in \mathcal{R} \setminus \{j\}$ occur in State 0, Resource j is occupied and is busy. Correspondingly, Automaton R_j moves from State 0 to State 1 upon the occurrences of σ^i_j and $\sigma^i_{k \to j}$ for $i \in \mathcal{P}$ and $k \in \mathcal{R} \setminus \{j\}$. When Process i finishes the usage of Resource j, and no other processes need to use Resource j, Resource j returns to idle. As we can see, Automaton R_j returns to State 0 from State 1 upon the occurrences of π^i_j and $\sigma^i_{j \to k}$ for $i \in \mathcal{P}$ and $k \in \mathcal{R} \setminus \{j\}$. On the other hand, if Resource j is further occupied by another process, R_j moves from State 1 to State 2 upon the occurrences of σ^i_j and $\sigma^i_{k \to j}$ for $i \in \mathcal{P}$ and $k \in \mathcal{R} \setminus \{j\}$. In this way, we can construct R_j.

To combine the resource automata and the process automata, we next introduce the definition of parallel composition of two different automata [53]. Formally, given $G_1 = (Q_1, \Sigma_1, f_1, \Gamma_1, q_{01}, Q_{m1})$ and $G_2 = (Q_2, \Sigma_2, f_2, \Gamma_2, q_{02}, Q_{m2})$, the parallel composition of G_1 and G_2 is denoted by

$$G_1 || G_2 := Ac(Q_1 \times Q_2, \Sigma_1 \cup \Sigma_2, f, (q_{01}, q_{02}), Q_{m1} \times Q_{m2}).$$

The transition function δ is defined as: for any $(q_1, q_2) \in Q_1 \times Q_2$ and any $\sigma \in \Sigma$,

$$f((q_1, q_2), \sigma) := \begin{cases} (q_1', q_2') & \sigma \in \Gamma_1(q_1) \cap \Gamma_2(q_2); \\ (q_1', q_2) & \sigma \in \Gamma_1(q_1) \setminus \Sigma_2; \\ (q_1, q_2') & \sigma \in \Gamma_2(q_2) \setminus \Sigma_1; \\ \text{undefined} & \text{otherwise,} \end{cases}$$

where $q_1' = f_1(q_1, \sigma)$ and $q_2' = f_2(q_2, \sigma)$.

The overall resource allocation mathematical model G can be obtained by parallelizing all the agents and resources automata. Formally, we have

$$G = P_1 || \cdots || P_d || R_1 || \cdots || R_h. \tag{1}$$

Example 1. *Let us consider a simple example. As shown in Figure 2, suppose that there are two robots, denoted by Robots* 1 *and* 2*, respectively. Both of them need to occupy Resource* 1 *to fulfill their tasks. Resource* 1 *can accommodate one robot at a time. We model Robots 1 and 2 in Figure 3a,b, respectively. Meanwhile, we model Resource 1 in Figure 3c. The resource allocation mathematical model $G = P_1 || P_2 || R_1$ is obtained in Figure 3d.*

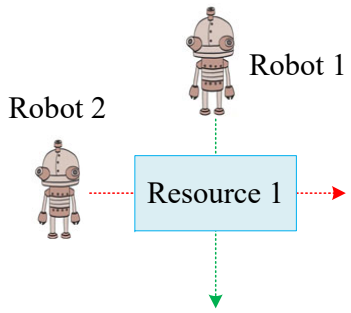

Figure 2. Two robots sharing one resource.

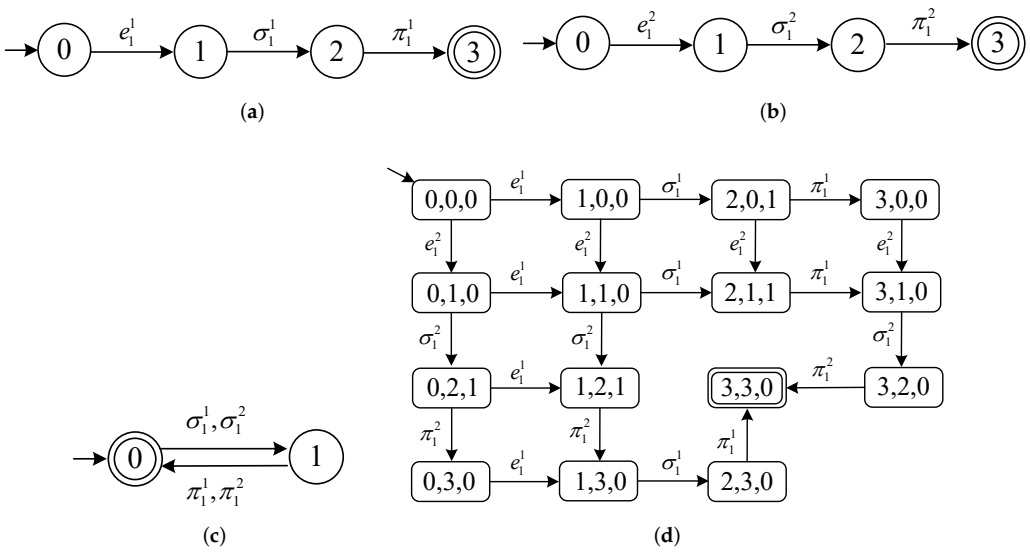

Figure 3. Automata P_1, P_2, R_1, and G; (**a**) Automaton P_1 for Robot 1; (**b**) Automaton P_2 for Robot 2; (**c**) Automaton R_1 for Resource 1; (**d**) System model G.

3. Optimization Algorithm

3.1. Problem Formulation

In this section, we calculate an optimal controller for G to achieve a given task. Particularly, the following two optimization objectives should be satisfied.

- Deadlock-Free : We say a state is a deadlock state if we cannot arrive at a marked state from it. In other words, once we arrive at a deadlock state, we can never reach a marked state;
- Efficiency: Among all the strings of G in the solution domain, efficiency requires us to execute a string with a minimal cost.

Let us first introduce the definition of a deadlock state.

Definition 1. *Given a state $q \in Q$, q is a deadlock state if there does not exist a string $s \in \Sigma^*$ such that $f(q,s) \in Q_m$.*

From a deadlock state, we can never reach a marked state. In this paper, we denote $Q_{dead} = \{q \in Q : (\forall s \in \Sigma^*) f(q,s) \notin Q_m\}$ by the set of all the deadlock states of G [53]. We also denote $Q_{illeg} = \{q \in Q : (\exists s \in \Sigma_{uc}^*) f(q,s) \in Q_{dead}\}$ by the set of illegal states from which we can reach a deadlock state via an uncontrollable event sequence. If we are in an illegal state, we may reach a deadlock state even if we disable all the controllable events in the future. Thus, to ensure the system is deadlock-free, all the illegal states are prohibited from reaching.

We define a cost function $C : Q \times \Sigma^* \to \mathbb{R}$, where $\mathbb{R}$ is the set of real numbers. Formally, for any $q \in Q$ and any $s \in \Sigma^*$, $C(q,s)$ is the cost resulting from executing event s when the system is in state q.

Given a state $q \in Q$ and the search depth $l \in \mathbb{N}$, we let $\Xi(q,l) = \{s \in \Sigma^* : |s| = l \wedge f(q,s)!\}$ be the set of strings $s \in \Sigma^*$ that are defined at z and are of the length l. To save the computational resource, we require that the solution calculated at the current state is optimal for the next $l \in \mathbb{N}$ steps. We assume that for each state of the automaton, there is at least one event defined at this state. This is not a restriction since for any state whose active event set is empty, we can always add a dumb self-loop with a cost of 0 at it. Thus, given the current state $q \in Q$, an optimal solution is always included in $\Xi(q,l)$. Now, the objective function for resource allocation can be expressed as follows.

Problem 1. *Given the system automaton G, the optimization for resource allocation in the current state $q \in Q$ is*

$$[t \in \Xi(q,l)] = \arg\min_{s \in \Xi(q,l)} C(q,s) \tag{2}$$

where $l \geq 1$ is the fixed number of prediction steps, and the model constraints are given as follows:

$$s^1 \in \Sigma_f; \tag{3}$$

$$f(q,s^i) \in Q \setminus Q_{illeg}, \text{ for all } i = 0,1,\ldots,l. \tag{4}$$

Condition (3) states the first event of the solution is enforceable so that it can be enforced by the controller. Condition (4) states that all the illegal states are prevented from being reached. By definition, the solution $t \in \Sigma^*$ of (2) is an event sequence of length l defined at the current state q and has the minimal cost. Note that there may be several minimal solutions of (2), i.e., the solutions of (2) need not to be unique. We can select any one of them and then execute the first event of t, i.e., t^1. After execution, the state of the system is updated, and a new optimal solution of (2) should be calculated. Since the first event of t is enforceable, it can always be enforced. Meanwhile, when we enforce an event occurrence, we must disable all the remaining controllable events. Given the current state $q \in Q$, we next show how to compute the optimal solution $t \in \Xi(q,l)$.

3.2. Solution

In this section, a set of algorithms are developed to solve Problem 1. Given a state $q \in Q$, we define the set of predecessor states of q, denoted by $\mathrm{Pre}(q)$, as:

$$\mathrm{Pre}(q) = \{q' \in Q : (\exists \sigma \in \Sigma) f(q',\sigma) = q\}.$$

Algorithm 1 calculates all the nondeadlock states $Q \setminus Q_{dead}$ of G.

Algorithm 1: Nondeadlock States

Input: Automaton G;
Output: $\mathcal{T}^*$;
1 Initially, set $\mathcal{T} \leftarrow Q_m$ and unflag all the elements in $\mathcal{T}$;
2 **while** $\exists$ *an unflagged state in* $\mathcal{T}$ **do**
3 Pick an unflgged $q \in \mathcal{T}$;
4 **foreach** $q' \in \mathrm{Pre}(q)$, *one by one* **do**
5 **if** $q' \notin \mathcal{T}$ **then**
6 Set $\mathcal{T} \leftarrow \mathcal{T} \cup \{q'\}$;
7 Unflag q' in $\mathcal{T}$;
8 Flag q in $\mathcal{T}$;
9 **return** $\mathcal{T}^* \leftarrow \mathcal{T}$.

To search for all the nondeadlock states, Algorithm 1 considers all the marked states first, then all the nondeadlock states that can reach Q_m within 1 step, then all the nondeadlock states that can reach Q_m within 2 steps, and so on. Algorithm 1 does not terminate until all the nondeadlock states are considered at least once. Since Q is finite, Algorithm 1 will terminate in finite steps.

Proposition 1. *The returned $\mathcal{T}^*$ of Algorithm 1 collects all the nondeadlock states of G, i.e., $\mathcal{T}^* = Q \setminus Q_{dead}$.*

Proof. We denote by $\mathcal{T}^0$ the set of states returned by Line 1 and $\mathcal{T}^i$ the set of states returned at the end of the ith iteration of the while-loop in Line 2.

We first prove that $\mathcal{T}^* \subseteq Q \setminus Q_{dead}$. The proof is by induction. Since $\mathcal{T}^0 = Q_m \subseteq Q \setminus Q_{dead}$, the base case is trivially true.

The induction hypothesis is that $\mathcal{T}^k \subseteq Q \setminus Q_{dead}$. In the $k+1$st iteration of the while-loop, all the predecessor states of $\mathcal{T}^k$ (if they are not included in $\mathcal{T}^k$) are added into $\mathcal{T}$. Hence, for all $q \in \mathcal{T}^{k+1} \setminus \mathcal{T}^k$, there exist a $q' \in \mathcal{T}^k$ and a $\sigma \in \Sigma$ such that $f(q,\sigma) = q'$. By the induction hypothesis, $q' \in Q \setminus Q_{dead}$. By Definition 1, there exists a sequence $s \in \Sigma^*$ such that $f(q',s) \in Q_m$. Moreover, since $f(q,\sigma) = q'$, we have $f(q,\sigma s) \in Q_m$. Therefore, $\mathcal{T}^{k+1} \subseteq Q \setminus Q_{dead}$. By Line 9, $\mathcal{T}^* \subseteq Q \setminus Q_{dead}$.

We next prove $\mathcal{T}^* \supseteq Q \setminus Q_{dead}$. For an arbitrary $q \in Q \setminus Q_{dead}$, by Definition 1, there exists a sequence $s = \sigma_1 \cdots \sigma_n \in \Sigma^*$ such that $f(q,s) \in Q_m$. Without loss of generality, we write $f(q,s^i) = q^i$ for $i = 0,1,\ldots,n$. By the while-loop on Line 2, $q^i \in \mathcal{T}^{n-i}$ for $i = 0,1,\ldots,n$. Therefore, $q \in Q \setminus Q_{dead}$. □

Given the set of nondeadlock states, Algorithm 2 returns the set of all the legal states.

Algorithm 2: Legal States

Input: System automaton G;
Output: $\mathcal{L}^*$;
1 Initially, call Algorithm 1 to compute the set of nondeadlock states $\mathcal{T}^*$;
2 Set $i \leftarrow 0$ and $\mathcal{L}_0 \leftarrow \mathcal{T}^*$;
3 **repeat**
4 Set $i \leftarrow i+1$;
5 $\mathcal{L}_i \leftarrow \mathcal{L}_{i-1} \setminus \{q \in \mathcal{L}_{i-1} : (\exists \sigma \in \Sigma_{uc}) f(q,\sigma) \notin \mathcal{L}_{i-1}\}$;
6 **until** $\mathcal{L}_i = \mathcal{L}_{i-1}$;
7 **return** $\mathcal{L}^* \leftarrow \mathcal{L}_i$.

Algorithm 2 computes all the legal states by iteratively removing the states q from the set of nondeadlock states $\mathcal{T}^*$ if we can arrive at a deadlock state from q via an uncontrollable event sequence. Since Q is finite, the algorithm will terminate in finite steps. The following proposition states that Algorithm 2 indeed returns all the legal states.

Proposition 2. *The returned $\mathcal{L}^*$ of Algorithm 2 collects all the legal states of G, i.e., $\mathcal{L}^* = Q \setminus Q_{illeg}$.*

Proof. We first prove that $\mathcal{L}^* \supseteq Q \setminus Q_{illeg}$. The proof is by induction. Initially, we have $\mathcal{L}_0 = \mathcal{T}^* = Q \setminus Q_{dead}$. By the definitions of Q_{illeg} and Q_{dead}, we have $Q_{dead} \subseteq Q_{illeg}$, which implies $Q \setminus Q_{dead} \supseteq Q \setminus Q_{illeg}$. Thus, $\mathcal{L}_0 \supseteq Q \setminus Q_{illeg}$ is true.

The induction hypothesis is that $\mathcal{L}_k \supseteq Q \setminus Q_{illeg}$. In the $k+1$st iteration of the repeat-until loop, Line 5 of Algorithm 2 removes all the $q \in \mathcal{L}_k$ such that $\exists \sigma \in \Sigma_{uc}$ with $f(q,\sigma) \notin \mathcal{L}_k$. Since $f(q,\sigma) \notin \mathcal{L}_k$ and $\mathcal{L}_k \supseteq Q \setminus Q_{illeg}$, $f(q,\sigma) \in Q_{illeg}$. Then, by the definition of Q_{illeg}, there exists a sequence $s \in \Sigma_{uc}^*$ such that $f(q,\sigma s) \in Q_{dead}$. Since $\sigma s \in \Sigma_{uc}^*$, all the $q \in \mathcal{L}_k$ that are removed from $\mathcal{L}_k$ are illegal states, i.e., $q \in Q_{illeg}$. Therefore, $\mathcal{L}_{k+1} \supseteq Q \setminus Q_{illeg}$.

We next prove that $\mathcal{L}^* \subseteq Q \setminus Q_{illeg}$. Given any $q \in \mathcal{L}^*$, by the repeat-until loop, if there exists $s \in \Sigma_{uc}^*$ such that $f(q,s)!$, then $f(q,s) \in \mathcal{L}^*$. Otherwise, it contradicts the assumption that the repeat-until loop does not terminate until $\mathcal{L}_i = \mathcal{L}_{i-1}$. Moreover, since $\mathcal{L}^* \subseteq \mathcal{L}_0 = Q \setminus Q_{dead}$, for any $q \in \mathcal{L}^*$ and any $s \in \Sigma_{uc}^*$, we have $f(q,s) \in Q \setminus Q_{dead}$. By the definition of Q_{illeg}, $q \notin Q_{illeg}$ or equivalently $q \in Q \setminus Q_{illeg}$. Since q is arbitrarily given, $\mathcal{L}^* \subseteq Q \setminus Q_{illeg}$. □

We let $q \in Q$ be the current state of the system G and $l \in \mathbb{N}$ be the prediction depth. Algorithm 3 is used to calculate $\Xi(q,l)$.

Algorithm 3 involves a breadth-first search of a tree with a depth of h. Each node of the tree is a pair $(q,s) \in Q \times \Sigma^{\leq l}$, where q is a state of G and s is a sequence with the length no larger than l (its length is the search depth). We say a node (q,s) is promising if q is legal,

i.e., $q \in Q \setminus Q_{illeg}$. Algorithm 3 visits the root node (q, ε) first, followed by all promising nodes at level 1, all promising nodes at level 2, and so on. The following theorem states that Algorithm 3 returns all possible promising nodes that can be reached from the initial node (q, ε) over a sequence in $\Xi(q, l)$.

Algorithm 3: Computation of Feasible Sequences

Input: The current state $q \in Q$ and the prediction depth l;
Output: Ξ^*;
1 Call Algorithms 1 and 2 to compute the set of legal states $Q \setminus Q_{illeg}$;
2 Set $\Xi \leftarrow \{(q, \varepsilon)\}$ and unflag (q, ε) in Ξ;
3 **while** $\exists$ *an unflagged element in* Ξ **do**
4 Randomly pick an unflagged $(q', s) \in \Xi$;
5 **if** $|s| < l$ **then**
6 **foreach** $\sigma \in \Gamma(q')$, *one by one* **do**
7 **if** $f(q', \sigma) \in Q \setminus Q_{illeg}$ **then**
8 Set $\Xi \leftarrow \Xi \cup \{(f(q', \sigma), s\sigma)\}$ and unflag $(f(q', \sigma), s\sigma)$ in Ξ;
9 Flag (q', s) in Ξ;
10 **return** $\Xi^* \leftarrow \Xi$.

Theorem 1. *We let Ξ^* be the set of pairs returned by Algorithm 3. Then, $\exists s \in \Xi(q, l)$ such that $f(q, s^i) \in Q \setminus Q_{illeg}$ for $i = 0, 1, \ldots, l$ if and only if $\exists (f(q, s), s) \in \Xi^*$.*

Proof. We denote by Ξ^i the set of nodes obtained at the end of the ith iteration of the repeat-until loop.

($\Rightarrow$) Since $f(q, s^i)!$ and $f(q, s^i) \in Q \setminus Q_{illeg}$, by recursively applying the while-loop on Line 3 of Algorithm 3, we have $(f(q, s^i), s^i) \in \Xi^i$. Therefore, we have $(f(q, s), s) \in \Xi^l = \Xi^*$.

($\Leftarrow$) For any $(q', s) \in \Xi^i$, we prove that $q' = f(q, s)$, $q' \in Q \setminus Q_{illeg}$, and $|s| = i$. The proof is by induction on $i = 0, 1, \ldots, l$. The base case is trivially true since $\Xi^0 = \{(q, \varepsilon)\}$, $q = f(q, \varepsilon)$, $q \in Q \setminus Q_{illeg}$, and $|s| = 0$. The induction hypothesis is that for all $(q', s) \in \Xi^i$ with $i < l$, we have $q' = f(q, s)$, $q' \in Q \setminus Q_{illeg}$, and $|s| = i$. We now prove that the same is also true for all $(q'', s') \in \Xi^{i+1}$. By the while-loop on Line 3 of Algorithm 3, there exist $(q', s) \in \Xi^i$ and $\sigma \in \Sigma$ such that $q'' = f(q', \sigma) \in Q \setminus Q_{illeg}$ and $s' = s\sigma$. Since $q' = f(q, s)$ and $|s| = i$, we have $q'' = f(q, s\sigma) = f(q, s')$ and $|q'| = i + 1$. Thus, for all $(q'', s') \in \Xi^{i+1}$, we have $q'' = f(q, s')$, $q'' \in Q \setminus Q_{illeg}$, and $|s'| = i + 1$. Therefore, for all $(q', s) \in \Xi^l = \Xi$, we have $q' = f(q, s)$, $q' \in Q \setminus Q_{illeg}$, and $|s| = l$. By definition, $s \in \Xi(q, l)$. □

Corollary 1. *For any $s \in \Xi(q, l)$, it is a solution of Problem 1 if $\exists (f(q, s), s) \in \Xi^*$ such that $s^1 \in \Sigma_f$ and there does not exist another $(f(q, t), t) \in \Xi^*$ with $t^1 \in \Sigma_f$ and $C(q, t) < C(q, s)$.*

Proof. For any $q \in Q$, Problem 1 intends to find a $s \in \Xi(q, l)$ having the minimum cost such that (i) $s^1 \in \Sigma_f$ and (ii) $f(q, s^i) \in Q \setminus Q_{illeg}$ for $i = 0, 1, \ldots, |s|$. By Theorem 1, Ξ^* collects all the $(f(q, s), s)$ such that $\exists s \in \Xi(q, l)$ and $f(q, s^i) \in Q \setminus Q_{illeg}$ for $i = 0, 1, \ldots, |s|$. Thus, s is a solution of Problem 1 if and only if $s^1 \in \Sigma_f$ and there does not exsit another $(f(q, t), t) \in \Xi^*$ such that $t^1 \in \Sigma_f$ and $C(q, t) < C(q, s)$. That completes the proof. □

By Corollary 1, we can always obtain a solution of Problem 1 (if it exists) by selecting one $(q, s) \in \Xi^*$ with $s^1 \in \Sigma_f$ that has a minimum cost $C(q, s)$.

Flow diagrams of the algorithm for optimal resource allocation are given in Figure 4. Specifically, we first model the resource automata and process automata as described in Section 2.2. The overall system model G can be obtained by paralleling all these automata as in (1). When the state of the system is updated, we apply Algorithms 1–3 to calculate all the feasible event sequences that start from the current state. By Corollary 1, by calculating

the time cost of all the feasible event sequences, we can always select one having the minimum time cost. Then, we enforce the first event of this sequence at the right time. After the execution of this event, the state of the system is updated again, and we repeat the above process.

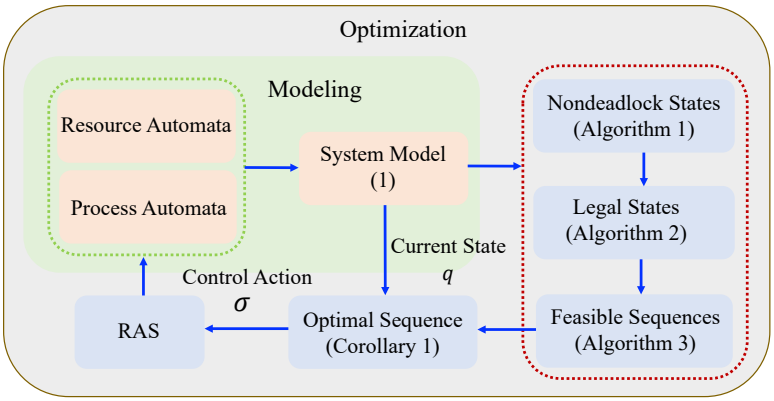

Figure 4. The implementation process of the proposed algorithm.

4. Case Study

In this section, we consider an application of the proposed resource allocation optimization algorithm.

Intersection

Intersection management is essential for safety and traffic efficiency. As shown in Figure 5, the research object of this section is a signal-free intersection connected by eight two-lane roads, which are denoted, respectively, by Roads $1, 2, \ldots, 8$. The intersection is divided into five mutually disjoint Resources a, b, c, d, and e. A vehicle coming from Road 3 and going to Road 7 needs to occupy in the order of Resource b and Resource a to complete the straight movement. A vehicle coming from Road 1 and going to Road 8 needs to occupy in the order of Resource a, Resource e, and Resource c to complete a left turn. A vehicle coming from Road 4 and going to Road 5 needs to occupy Resource d to complete a right turn, and so on in a similar fashion.

Within the intersection, the trajectory of a vehicle is determined by its source road (before the intersection) and its destination road (after the intersection). For example, a vehicle coming from Road 1 (source road) and going to Road 5 (destination road) must occupy in the order of Resources a and d to pass through the intersection. As shown in Table 1, the vehicles arriving at the intersection are classed into 12 different types according to their source roads (SRs) and their destination roads (DRs). Figure 6 visually shows all the vehicle types. For example, as shown in Figure 6a, the 1st type of vehicle is a going-straight vehicle that is coming from Road 1 and going to Road 5, the 3rd type of vehicle is a going-straight vehicle that is coming from Road 3 and going to Road 7, the 6th type of vehicle is a right-turn vehicle that is coming from Road 2 and going to Road 8, and the 12th type of vehicle is a left-turn vehicle that is coming from Road 4 and going to Road 6.

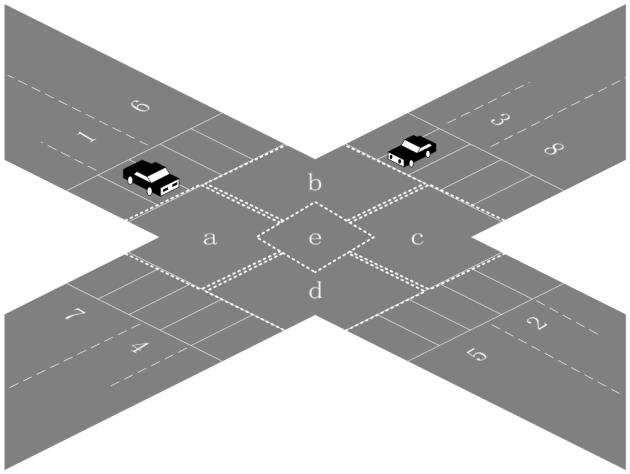

Figure 5. An unsignalized intersection.

Table 1. Types of vehicles.

Type	SR	DR	Type	SR	DR	Type	SR	DR	Type	SR	DR
1	R1	R5	4	R4	R8	7	R3	R6	10	R2	R7
2	R2	R6	5	R1	R7	8	R4	R5	11	R3	R5
3	R3	R7	6	R2	R8	9	R1	R8	12	R4	R6

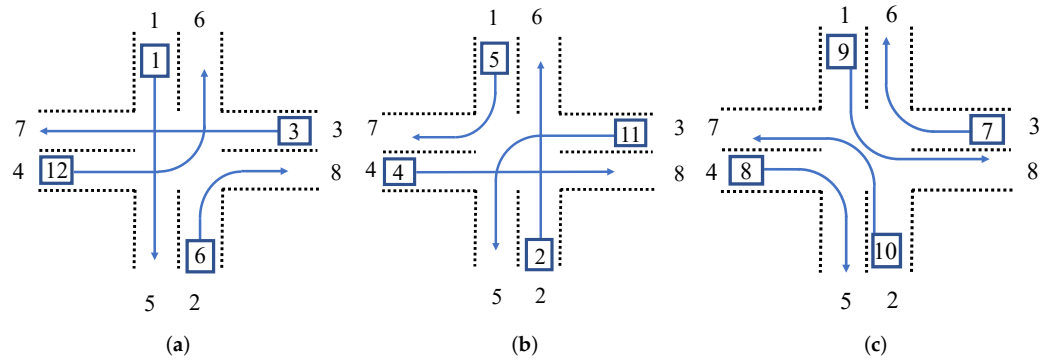

Figure 6. Different vehicle types; (**a**) Vehicle types of 1, 3, 6, and 12; (**b**) Vehicle types of 2, 4, 5, and 11; (**c**) Vehicle types of 7, 8, 9, and 10.

We next discuss how to model processes and resources in the traffic systems. We refer to the process of a vehicle of type i passing through the intersection as Process i. As shown in Figure 7a, Automaton P_1 is used to model the working status of Process 1. The 1st type of vehicle needs to occupy Resources a and d successively to pass through the intersection. Thus, in State 0, the vehicle should first apply for the usage of Resource a. Upon the occurrence of e^1_a, A_1 moves from State 0 to State 1. In State 1, if the vehicle receives an acknowledgement from the CRM, it drives into Resources a, and A_1 moves from State 1 to State 2. After that, the vehicle further sends the usage of Resource d to the CRM ($e^1_{a\to d}$ occurs in State 2). After receiving an acknowledgment, the vehicle drives into Resource d ($\sigma^1_{a\to d}$ occurs in State 3). When the vehicle leaves Resource d (π^1_d occurs in State 4), it passes through the intersection. Similarly, we can construct $P_2, \ldots, P_{12}$.

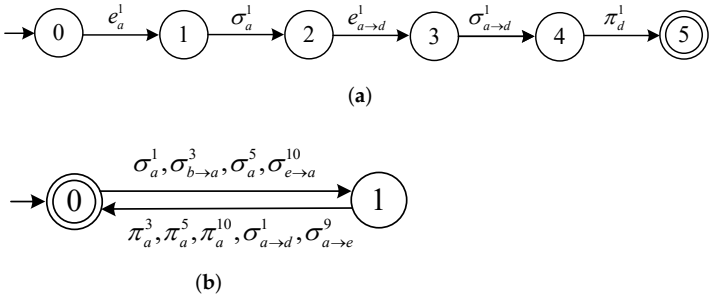

Figure 7. Automata P_1 and R_a; (**a**) Automaton P_1 for Process 1; (**b**) Automaton R_a for Resource a.

Next, we show how to model Resources a. By Figure 6, Resource a can be occupied by the 1st type of vehicle (a vehicle coming from Road 1 and going to Road 5), the 3rd type of vehicle (a vehicle coming from Road 3 and going to Road 7), a 5th type of vehicle (a vehicle coming from Road 1 and going to Road 7), or the 10th type of vehicle (a vehicle coming from Road 2 and going to Road 7). Thus, in Figure 7b, σ_a^1, $\sigma_{b\to a}^3$, σ_a^5, and $\sigma_{e\to a}^{10}$ are defined in State 0 of R_a. When σ_a^1, $\sigma_{b\to a}^3$, σ_a^5, and $\sigma_{e\to a}^{10}$ occurs in State 0, Resource 1 is occupied by a 1st, 3rd, 5th, and 10th type of vehicles, respectively, and Automaton R_a moves from State 0 to State 1. When the 1st, 3rd, 5th, 9th, and 10th type of vehicles leave Resource a, upon the occurrences of $\sigma_{a\to d}^1$, π_a^3, π_a^5, $\sigma_{a\to e}^9$, and π_a^{10}, automaton R_a returns to State 0 from States 1. Similarly, we can construct $R_2, \ldots, R_5$.

Suppose that there are n vehicles arriving at the intersection. Without loss of generality, let vehicle i be a vehicle of type m_i, $i = 1, \ldots, n$. Then, the dynamics of resource occupation of the traffic system can be obtained by $G = (Q, \Sigma, f, \Gamma, q_0, Q_m) = P_{m_1} || \cdots || P_{m_n} || R_a || \cdots || R_e$. Note that the traffic system should be safe, i.e., all the vehicles in an intersection cannot collide. Since a resource can be occupied by one vehicle at a time, the safety of the system has been guaranteed by the system model G. To guide vehicles to leave the intersection efficiently, we define deadlock-free and the efficiency of the traffic system as follows.

- Deadlock-Free: All the vehicles must not block each other so that each of them cannot accomplish the movement. Deadlock-freeness requires that vehicles in an intersection should not block other vehicles. There exists a deadlock in the traffic system if there exists a group of vehicles that block each other and cannot move in the intersection. For each deadlock, there exists a group of vehicles that want to drive into a resource already occupied by another vehicle in this group. For example, it is supposed that Resources a and e are occupied by left-turn vehicles $v1$ (coming from Road 1) and $v2$ (coming from Road 2), respectively. The intersection system is blocked since Vehicle $v1$ is waiting for Vehicle $v2$ to leave Resource e, and Vehicle $v2$ is waiting for Vehicle $v1$ to leave Resource a. When there exists a deadlock, there does not exist an event sequence from the current state to the marked state of G since some vehicles can never leave the intersection.
- Efficiency: the total time for all the vehicles to accomplish their movements is minimal. That is, among all the sequences that start from the current state and end up at the marked state, efficiency requires us to select one with a minimal time cost. In other words, the more efficient the proposed approch is, the less time a vehicle should take for passing through the intersection. Correspondingly, the intersection can enjoy a larger throughout capacity, and the queue length of vehicles delayed at the intersection is smaller.

For a state $q \in Q$ and an event sequence $s \in \Sigma^*$, the required time for the execution of s from q, denoted by $C(q, s)$, needs to be calculated. For example, we consider a sequence $s = \sigma_{3\to b}^3 e_{a\to e}^9 \sigma_{a\to e}^9 e_{b\to a}^3 e_{e\to c}^9 \sigma_{b\to a}^3 \sigma_{e\to c}^9$ that is defined at the current state q. By s, it is not hard to find that two vehicles are traveling in the intersection (a Type 3 vehicle from Road 3 to Road 7 and a Type 9 vehicle from Road 1 to Road 8). To calculate $C(q, s)$, as shown in

Figure 8, we need to first translate s into a "calendar" according to the execution order of events in s. The calendar for the sequence s explicitly describes when each event in s should be executed, and how long the execution lasts. In Figure 8, each line exhibits the movement of a vehicle. The solid vertical bars and hollow rectangles represent uncontrollable events and controllable events, respectively. The continuous parts of each line indicate that the corresponding vehicle is traveling in the intersection. The broken parts of each line indicate that the vehicle has stopped and is waiting at the intersection. The dashed arrows indicate the execution order of the events, i.e., $\sigma^3_{b\to a}$ is executed after $e^9_{e\to c}$. By Figure 8, the time cost for executing s is about 5 s.

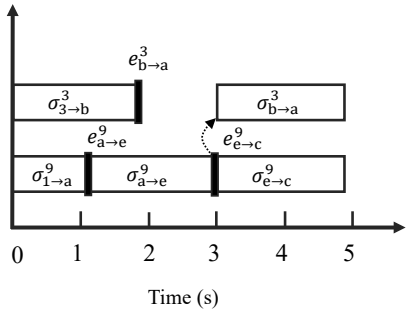

Figure 8. Calendar: the process of two vehicles passing through an intersection.

The structure of the traffic management system is depicted in Figure 9. It is worth noting that in the traffic management system, we have $\Sigma_c = \Sigma_f = \{\sigma^i_j, \sigma^i_{k\to j}, \pi^i_j : i \in \mathcal{P}, j, k \in \mathcal{R}\}$, i.e., all the controllable events are enforceable, and vice versa. For each $q \in Q$ and the prediction depth $l \in \mathbb{N}$, the blue parts of Figure 9 compute all the feasible sequences in $\Xi(q, l)$. By definition, the state of G can be updated following the execution of an enforceable event or an uncontrollable event occurrence. When the state of G is updated, the red parts of Figure 9 return the solution of Problem 1 and execute its first event $\sigma \in \Sigma_f$. The yellow parts of Figure 9 translate σ into appropriate input time-driven signals to the actuators of the vehicles. After the execution of σ, the state of the system is updated again, and we repeat the above process.

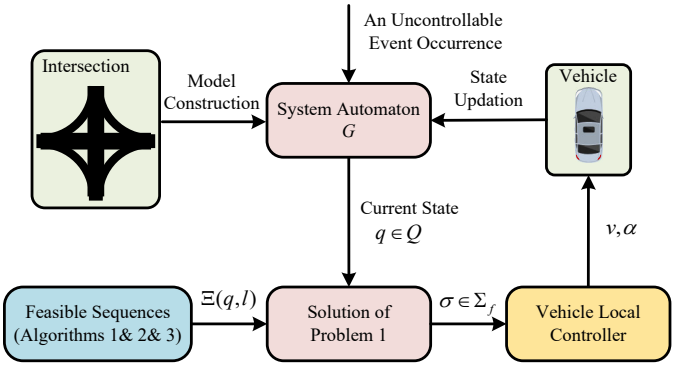

Figure 9. The structure of the traffic management system.

In this section, we adopt an Instant Services Measure (ISM) to implement the proposed approach. Specifically, the ISM immediately considers a vehicle once it arrives at the intersection. In other words, the calculated solution is optimal for all the vehicles arriving at the intersection.

Remark 1. *Note that the system model must be refined upon each new vehicle arrival. Specifically, suppose that system G now is in state $q \in Q$, if a new vehicle of type i approaches the intersection, the system should first be updated to $G(q)||P_i$, where $G(q) \sqsubseteq G$ ("$\sqsubseteq$" denotes subgraph) is the*

accessible part of G from q. Then, an optimal control strategy is calculated using the proposed approach and the system model $G(q)||P_i$.

5. Simulation Results

Simulations are provided in this section to demonstrate the effectiveness of the proposed mathematical model and optimization algorithm. The simulations were run on a Windows 10.0 PC with a 3.8 GHz AMD CPU and 16 GB memory. All of the algorithms were implemented in the Python programming language. In this paper, the arrival of vehicles from each source road satisfies the Poisson distribution, where the parameter of the Poisson distribution λ is 0.1, 0.13, 0.17, and 0.2. Note that λ is also called the vehicle arrival rate (VAR). For example, when the VAR $\lambda = 0.1$, the average time interval between two incoming vehicles is 10 s. On the other hand, it is 5 s when $\lambda = 0.2$. The probability of going straight, turning left, or turning right for each vehicle arriving at the intersection is specified as a uniform distribution.

5.1. Queue Length under Different VARs

First, we test the effectiveness and efficiency of the proposed algorithm by changing the VARs. Figure 10 visualizes the volume of traffic density in terms of the quantity of vehicles arriving at the intersection every 50 seconds. It is observed that a higher λ leads to a higher traffic density. For example, in the second 50 s, there are 11, 12, 18, and 19 vehicles entering the intersection when the arrival rates λ are set as 0.1, 0.13, 0.17, and 0.2, respectively. In Figure 11, the x-axis, y-axis, and z-axis denote, respectively, the simulation time, vehicle queue length, and VAR. As we can see from Figure 11, when $\lambda = 0.1$ and $\lambda = 0.13$, all the vehicles can pass through the intersection freely, and the vehicle queue length is small. As λ increases to 0.17, a minor traffic jam occurs. However, the queue length will not diverge to an infinitely large number, and the intersection still works. When $\lambda = 0.2$, a traffic jam occurs at the intersection due to the limitation of the spatial space.

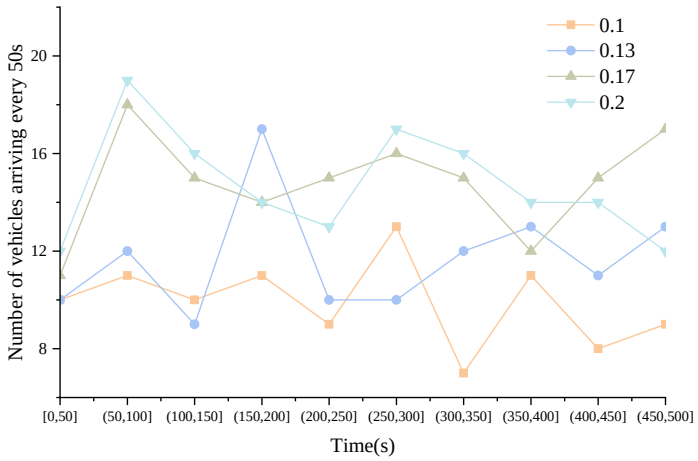

Figure 10. Number of vehicles arriving at the intersection every 50 s when the VARs are 0.1, 0.13, 0.17, and 0.2.

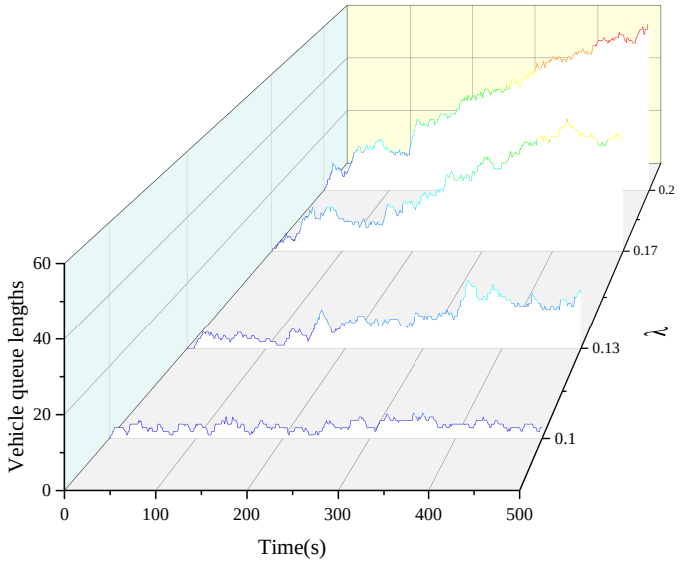

Figure 11. Numbers of vehicles queued at the intersection (queue lengths) when the VARs are 0.1, 0.13, 0.17, and 0.2.

5.2. Comparisons of Different Approaches under Different VARs

In the second simulation, we consider another two implementation measures of the proposed approach, which are denoted by the First-Come-First-Serve Measure (FCFSM) and the Period-Based Measure (PBM). In contrast to ISM, the FCFSM iteratively decides the passing order of vehicles by their earliest arrival times at an intersection. In other words, the earlier a vehicle arrives at the intersection, the higher priority it has going through the intersection. This principle is adopted in [42–45] for scheduling vehicles in an intersection. Different from the FIFOM and the ISM, the PBM first calculates an optimal passing order of a batch of vehicles coming in the current period, then followed by the next period, and so on. Before the first batch of vehicles leaves the intersection, the second batch of vehicles must wait at the intersection. We found that [33–35,37,38,51–53] optimize vehicle scheduling in a period-based principle.

All the measures are evaluated by two criteria: (i) traffic throughput (TT), i.e., the number of vehicles leaving the intersection per unit time, and (ii) traffic smoothness (TS), i.e., the average waiting time when vehicles arrive at and pass through the intersection. Table 2 illustrates the performance of the above two measures under different VARs λ. Particularly, in Table 2, "Mea" stands for "Measure".

Table 2. Comparision of FCFSM, PBM, and ISM under different VARs.

Mea	VAR	TT (unit/s)	TS (s)	Mea	VAR	TT (unit/s)	TS (s)	Mea	VAR	TT (unit/s)	TS (s)
FCFSM	0.10	1.37	42.29	PBM	0.10	1.78	32.25	ISM	0.10	2.02	27.33
	0.13	1.42	40.86		0.13	1.87	33.06		0.13	2.00	31.00
	0.17	1.37	41.94		0.17	1.58	38.42		0.17	1.87	33.41
	0.20	1.73	45.19		0.20	2.08	40.66		0.20	2.18	34.74

As we can see from Table 2, compared with the FCFSM, the ISM increases the TT by 47.45%, 40.85%, 36.50%, and 26.01%, and it decreases the TS by 35.37%, 24.13%, 20.34%, and 23.12%, when the VARs are taken as 0.10, 0.13, 0.17, and 0.20, respectively. The ISM outperforms the FCFSM because the ISM allows more than one vehicle to drive in the intersection as long as they do not collide with each other. However, FCFSM allows only one vehicle to pass through the intersection simultaneously, which leads to extra delay. Next, we compare the ISM with the PBM. As shown in Table 2, the ISM increases the TT by 13.48%, 6.95%, 18.35%, and 4.81%, and it decreases the TS by 16.03%, 6.23%, 13.04%,

and 14.56% when the VARs are taken as 0.10, 0.13, 0.17, and 0.20, respectively. The ISM is more efficient than the PBM since the ISM considers all the vehicles approaching the intersection, whereas the PBM considers only vehicles approaching the intersection in one period. PBM leads to extra delay in the sense that all the vehicles approaching the intersection have to wait at the intersection until the current batch of vehicles finish their journey in the intersection.

5.3. Performance of Different Approaches under Changing VARs

In the above simulations, the performance of the proposed approach is tested under four constant VARs λ. In practice, however, λ cannot be a constant. In other words, the value of λ can change during the simulation. In the third simulation, we use a Markov model to describe the changing conditions of the VAR λ. As depicted in Figure 12, there are two states in the Markov model M. When M is in State 0, the VAR is λ_1; and when M is in State 1, the VAR is λ_2. The state of the Markov model can be updated in each step of the simulation. More specifically, in each step of the simulation, if M is in State 0, then the system may make a state transition to State 1 with a probability of p_2 and make no state transition (still in State 0) with a probability of p_1. If M is in State 1, the system may make a state transition to State 0 with a probability of p_4 and make no state transition (still in State 1) with a probability of p_3. Considering that the VAR cannot be changed in a short space of time, we set $p_1 = 0.9$, $p_2 = 0.1$, $p_3 = 0.9$, and $p_4 = 0.1$. We consider four different combinations of λ_1 and λ_2 in this experiment, as shown in Table 3.

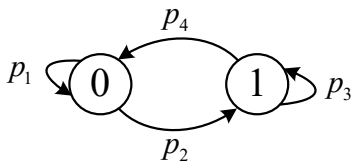

Figure 12. The Markov structure M for modeling the changing VARs.

Table 3. Combinations of the VARs λ_1 and λ_2.

Case	λ_1	λ_2	Case	λ_1	λ_2	Case	λ_1	λ_2	Case	λ_1	λ_2
1	0.10	0.13	2	0.13	0.15	3	0.15	0.17	4	0.17	0.20

Figure 13 shows the changing VARs when we take different combinations of λ_1 and λ_2. We take Case 4 ($\lambda_1 = 0.17$ and $\lambda_2 = 0.2$) as an example. In the first 15 s, it has a VAR of 0.2, and M is in State 0. Then, the Markov model M moves to State 1, and the VAR is changed to 0.17. Next, after approximately 8 s, M returns to State 0, and the VAR is again set to 0.2. Figure 14 shows the queue length of vehicles under different cases. Similar to the first simulation, if Case 1 occurs, i.e., $\lambda_1 = 0.1 \wedge \lambda_2 = 0.13$, or Case 2 occurs, i.e., $\lambda_1 = 0.13 \wedge \lambda_2 = 0.15$, all the vehicles can pass through the intersection freely. If Case 3 occurs, i.e., $\lambda_1 = 0.15 \wedge \lambda_2 = 0.17$, a small traffic jam occurs at the intersection. If Case 4 occurs, i.e., $\lambda_1 = 0.17 \wedge \lambda_2 = 0.2$, a traffic jam occurs. The result reveals that our proposed algorithm is robust to the changing VARs. In Table 4, we compare the ISM with the FCFSM and the PBM under Cases 1, 2, 3, and 4. As shown in Table 4, compared with the FCFSM, the ISM increases the TT by 34.78%, 46.09%, 30.41%, and 39.26%, and it decreases the TS by 55.13%, 53.55%, 33.50%, and 29.05%, when we take cases 1, 2, 3, and 4, respectively. Furthermore, compared with the PBM, the ISM increases the TT by 13.14%, 14.72%, 15.57%, and 19.75%, and it decreases the TS by 43.25%, 37.39%, 23.66%, and 19.72% when we take Cases 1, 2, 3, and 4, respectively. The proposed ISM still outperforms the FCFSM and the PBM under changing VARs.

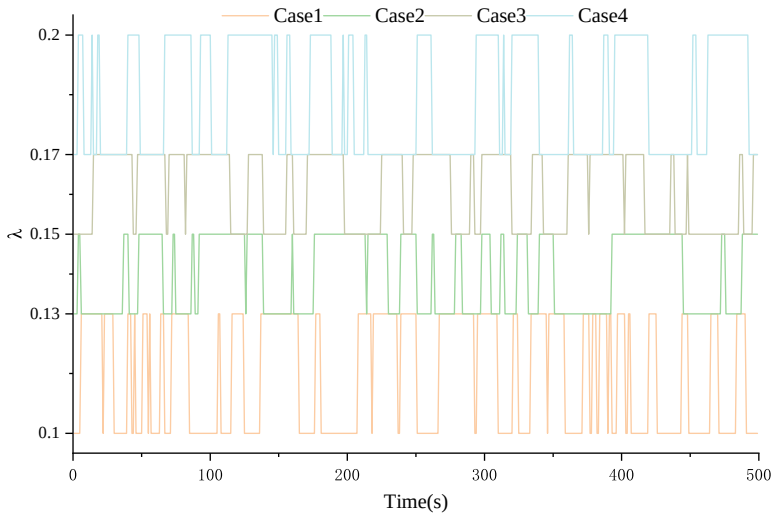

Figure 13. The changing arrival rates for different combinations of λ_1 and λ_2.

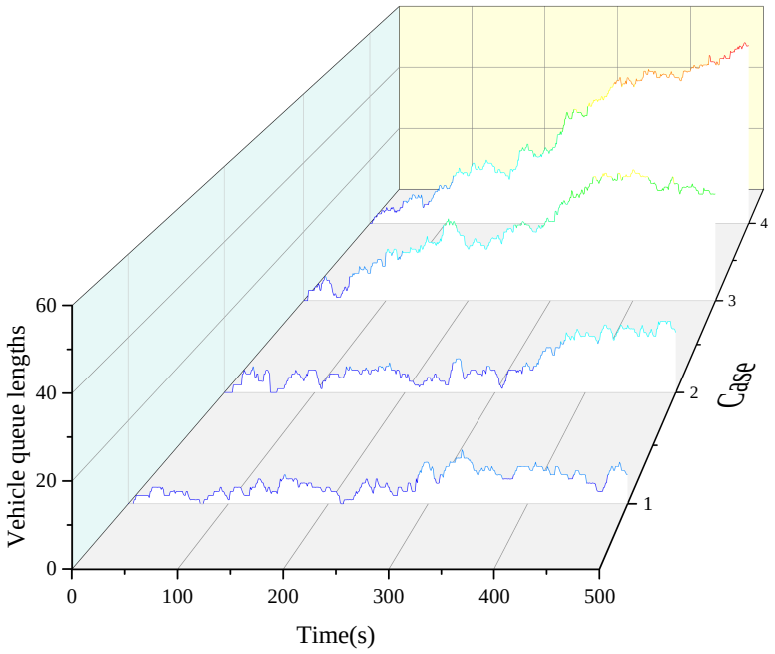

Figure 14. Number of vehicles queued at the intersection (queue lengths) with different combinations of λ_1 and λ_2.

Table 4. Comparision of FCFSM, PBM, and ISM under different cases.

Mea	Case	TT (unit/s)	TS (s)	Mea	Case	TT (unit/s)	TS (s)	Mea	Case	TT (unit/s)	TS (s)
FCFSM	1	1.15	38.24	PBM	1	1.37	30.24	ISM	1	1.55	17.16
	2	1.28	41.03		2	1.63	30.44		2	1.87	19.06
	3	1.48	37.55		3	1.67	32.71		3	1.93	24.97
	4	1.35	41.14		4	1.57	36.36		4	1.88	29.19

5.4. Performance of the ISM under Different Prediction Depths

As shown in Problem 1, an optimal control action can be calculated by looking ahead l steps from the current state and then finding a sequence having the minimun cost and executing the first component of this sequence. Intuitively, the deeper we search from the current state (taking a larger l), the better a solution returns. If we set l to be infinitely large, the returned solution would be the global optimum value. However, to save computing

resources, we would like to take a small l. To balance the computing resources and computing results, we next test the performance of the proposed approach under different l. The performances are measured by the average queue length (AQL) and the number of leaving vehicles (NLV), i.e., the number of vehicles passing through the intersection. The results are shown in Table 5.

Table 5. Performance of ISM under different search depths.

Depth	VAR	AQL (unit)	NLV (unit)	Depth	VAR	AQL (unit)	NLV (unit)
1	0.10	3.45	96.00	3	0.10	3.42	96.00
	0.13	11.18	97.00		0.13	8.78	104.00
	0.17	19.38	106.00		0.17	14.99	117.00
	0.20	33.46	104.00		0.20	30.65	112.00
2	0.10	3.42	96.00	4	0.10	3.42	96.00
	0.13	9.05	104.00		0.13	8.72	106.00
	0.17	17.25	113.00		0.17	14.93	118.00
	0.20	31.92	111.00		0.20	30.65	112.00

By Table 5, when the VAR is 0.10, both the AQL and the NLV have no dramatic changes when l is taken as 1, 2, 3, and 4. That is because when the traffic flow is light, the "greedily optimal" solution approximates the "global optimal" solution. We can look one step from the current state to find an optimal control action to execute. However, when the VAR increases to 0.2, the AQL is 33.46 and the NLV is 104 if $l = 1$, whereas for $l = 4$, the AQL is 30.65 and the NLV is 112. Clearly, if the traffic flow is heavy, a larger l will return a better solution. In addition, the improvement of the performance is insignificant when we take $l = 3$ and $l = 4$ even if the VAR is 0.2. This implies that we can find a near-global-optimal control action by looking ahead only three steps at any instant, which can significantly reduce computing resource consumption.

A video containing the simulation results that can be accessed at all time has been shared on Onedrive (https://1drv.ms/v/s!Av-fOZK0QsIHhTlWwB4HjDlSk2Qd?e=pFw6Ko, accessed on 1 October 2022). In this video, a PID controller is used for longitudinal tracking, and a pure pursuit controller is used for lateral tracking. The execution frequency of the low-level vehicle control module is 40.0 Hz.

6. Discussion

The proposed mathematical model can be applied to other physical systems that involve resource sharing, such as robot scheduling in a warehouse [54,55].

To illustrate this, let us consider a simple warehouse environment depicted in Figure 15a, where two mobile robots named R1 and R2 are serving in the planner environment. The warehouse environment is portioned into eight disjoint cells, and each position is assigned a natural number of $1, \ldots, 8$. We call such a cell as a position. We denote by $\mathcal{P} = \{1, \ldots, 8\}$ the set of all the positions. For a given Position $i \in \mathcal{P}$, we say Position $j \in \mathcal{P}$ is a successor of Position i, denoted by $i \to j$, if there is a route from Position i to Position j. Similarly, we say Position $j \in \mathcal{P}$ is a predecessor of Position i, denoted by $j \to i$, if there is a route from Position j to Position i. Let $Suc(i) = \{j \in \mathcal{P} : i \to j\}$ be the set of all the successors of Position i, and $Pre(i) = \{j \in \mathcal{P} : j \to i\}$ be the set of all the predecessors of Position i. For example, let us consider Position 2 in Figure 15a. Its predecessors are $Pre(2) = \{1, 3\}$, and its successors are $Suc(2) = \{1, 3\}$.

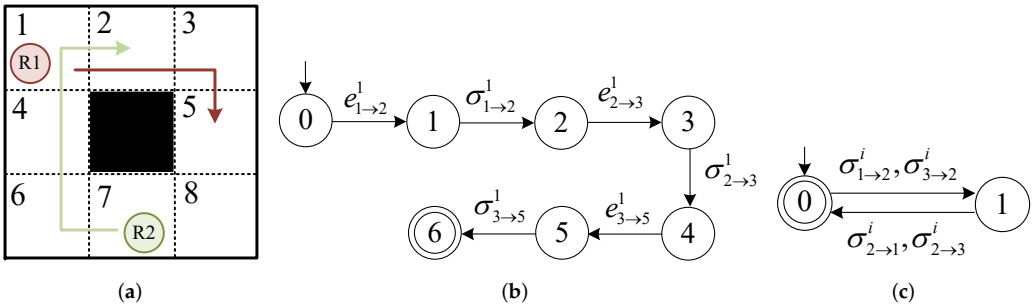

Figure 15. Modeling for warehouse; (**a**) A warehouse; (**b**) Process P_1 for Robot 1; (**c**) Resource R_2 for Position 2.

Each robot is associated with a task, which is characterized as a target position in the warehouse. A robot should move to the target position over a specific trace for completing the task. The "dynamics" of the robot are modeled as a process automaton. In the automaton, the system behaviors are abstracted to kinds of suitable-defined events, such as "a robot moves from Position i to Position j" . For example, we assume that Robot R1 needs to move to Position 5 via the position sequence of $1 \to 2 \to 3 \to 5$ to fulfill its task. The process automaton for Robot R1 is modeled as P_1 in Figure 15b, where $e^1_{i\to j}$ means that Robot R1 sends a request to the CRM for moving from Position i to Position j, and $\sigma^1_{i\to j}$ means that Robot R1 receives an acknowledgement and starts to move from Position i to Position j. Similarly, we can model Robot R2 that moves from Position 7 to Position 2 via a position sequence of $7 \to 6 \to 4 \to 1 \to 2$.

At the same time, the resource automaton R_2 for Position 2 is given in Figure 15c. By Figure 15a, Position 2 can be occupied by robots coming from Position $i \in Pre(2)$, and a robot that is in Position 2 can move to Position $i \in Suc(2)$. Thus, upon the occurrence of $\sigma^i_{j\to 2}$, $j \in Pre(2)$, Position 2 is occupied by Robot i, and R_2 moves to State 1. When R_2 is in State 1, upon the occurrence of $\sigma^i_{2\to j}$, $j \in Suc(2)$, Position 2 becomes idle again, and R_2 returns to State 0. Similarly, we can model Positions 1 and $3 \sim 8$.

The overall system can be obtained by $G = P_1||P_2||R_1||\cdots||R_8$. By applying the proposed algorithm, we can always find an optimal sequence to guide Robots 1 and 2 to finish their tasks safely (without collision and deadlock) and efficiently (with the minimum time cost).

7. Conclusions

In this paper, we have proposed an automaton model for resource allocation optimization. This model is general in the sense that it can be applied to different types of resource allocation problems. With the proposed model, we have developed an optimal control mechanism that positively drives the system to a marked state in the most efficient way. That is, among all the deadlock-free sequences, we develop algorithms to select one having the minimum cost and then execute it by disabling and enforcing events. This is accomplished in two steps, the first of which computes all the legal states from which we can always arrive at a marked state, and the second step repeatedly enforces controllable events by looking ahead so that only legal states can be reached and the cumulative cost for executing these control decisions is minimized. As a case study, we have considered the vehicle scheduling problem in a signal-free intersection. The intersection is divided into several resources according to the crossing location of two different vehicles' trajectories. The algorithm is sufficiently efficient to consider each arrived vehicle in real time. The simulation results demonstrated the effectiveness of the proposed model and algorithm.

One direction for future research is currently underway to validate the proposed approach through physical experiments. Another direction in which one can enhance the application scope of the proposed approach by accommodating partially observable events in the system model, as some events are often unobservable by nature. Furthermore,

according to the practical applications, other notions of optimality could also be specified, and new algorithms for obtaining new type of the optimal controller may be developed.

Author Contributions: Conceptualization, Y.H. and W.L.; methodology, Y.H. and W.L.; software, Y.M.; validation, Y.Z., Y.J. and Q.L.; formal analysis, Y.H. and Y.M.; investigation, W.L.; resources, Y.J. and W.L.; data curation, Y.Z.; writing—original draft preparation, Y.H. and Y.M.; writing—review and editing, Y.H., Y.M. and W.L.; visualization, Y.Z. and Y.J.; supervision, Q.L. and W.L.; project administration, Q.L. and W.L.; funding acquisition, Q.L. and W.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by National Natural Science Foundation of China under Grant 92048205 and the Pujiang Talents Plan of Shanghai under Grant 2019PJD035.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Simatic, M. *Operating System Concepts*; Addison-Wesley Pub. Co.: Boston, MA, USA, 2005.
2. Wang, N.; Wang, C.; Niu, Y.; Yang, M.; Yu, Y. A Two-Stage Charging Facilities Planning Method for Electric Vehicle Sharing Systems. *IEEE Trans. Ind. Appl.* **2021**, *57*, 149–157. [CrossRef]
3. Nguyen, S.; Thiruvady, D.; Zhang, M.; Alahakoon, D. Automated Design of Multipass Heuristics for Resource-Constrained Job Scheduling With Self-Competitive Genetic Programming. *IEEE Trans. Cybern.* **2022**, *52*, 8603–8616. [CrossRef]
4. Di Febbraro, A.; Giglio, D.; Sacco, N. A deterministic and stochastic Petri net model for traffic-responsive signaling control in urban areas. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 510–524. [CrossRef]
5. Lin, Y.; Hsu, H.; Lin, S.; Lin, C.; Jiang, I.H.; Liu, C. Graph-based modeling, scheduling, and verification for intersection management of intelligent vehicles. *ACM Trans. Embed. Comput. Syst. (TECS)* **2019**, *18*, 1–21. [CrossRef]
6. Steere, D.; Shor, M.; Goel, A.; Walpole, J.; Pu, C. Control and modeling issues in computer operating systems: Resource management for real-rate computer applications. In Proceedings of the 39th IEEE Conference on Decision and Control (CDC), Sydney, Australia, 12–15 December 2000; Volume 3; pp. 2212–2221.
7. Huang, G.; Gong, W.; Zhang, B.; Li, C.; Li, C. An Online Buffer-Aware Resource Allocation Algorithm for Multiuser Mobile Video Streaming. *IEEE Trans. Veh. Technol.* **2020**, *69*, 3357–3369. [CrossRef]
8. Dijkstra, E.W. *Cooperating Sequential Processes*; Springer: New York, NY, USA, 1968.
9. Habermann, A. Prevention of system deadlocks. *Comm. ACM* **1969**, *12*, 373–377. [CrossRef]
10. Havender, J.W. Avoiding deadlock in multitasking systems. *IBM Syst. J.* **1968**, *7*, 74–84. [CrossRef]
11. Holt, R.C. Some Deadlock Properties of Computer Systems. *ACM Comput. Surv.* **1972**, *4*, 179–196. [CrossRef]
12. Ibrahim, M.; Reveliotis, S.; Nazeem, A. Maximal Linear Deadlock Avoidance Policies for Sequential Resource Allocation Systems: Characterization, Computation, and Approximation. *IEEE Trans. Autom. Control* **2021**, *66*, 3906–3921. [CrossRef]
13. Park, J.; Reveliotis, S. Algebraic synthesis of efficient deadlock avoidance policies for sequential resource allocation systems. *IEEE Trans. Robot. Autom.* **2000**, *16*, 190–195. [CrossRef]
14. Reveliotis, S.; Fei, Z. Robust Deadlock Avoidance for Sequential Resource Allocation Systems With Resource Outages. *IEEE Trans. Autom. Sci. Eng.* **2017**, *14*, 1695–1711. [CrossRef]
15. Tajer, A.; Zohdy, M.; Alnajjar, K. Resource Allocation Under Sequential Resource Access. *IEEE Trans. Commun.* **2018**, *66*, 5608–5620. [CrossRef]
16. Reveliotis, S.; Ferreira, P. Deadlock avoidance policies for automated manufacturing cells. *IEEE Trans. Robot. Autom.* **1996**, *12*, 845–857. [CrossRef]
17. Li, N.; Wang, Z.; Pei, Z. Sequential Resource Planning Decisions in an Epidemic Based on an Innovative Spread Model. *IEEE Trans. Autom. Sci. Eng.* **2022**, *19*, 677–691. [CrossRef]
18. Qi, N.; Huang, Z.; Zhou, F.; Shi, Q.; Wu, Q.; Xiao, M. A Task-driven Sequential Overlapping Coalition Formation Game for Resource Allocation in Heterogeneous UAV Networks. *IEEE Trans. Mob. Comput.* **2022**. [CrossRef]
19. Theunissen, R.; Petreczky, M.; Schiffelers, R.; Beek, D.V. Application of Supervisory Control Synthesis to a Patient Support Table of a Magnetic Resonance Imaging Scanner. *IEEE Trans. Autom. Sci. Eng.* **2014**, *11*, 20–32. [CrossRef]
20. Zhao, B.; Lin, F.; Wang, C.; Zhang, X.; Polis, M.P.; Wang, L.Y. Supervisory Control of Networked Timed Discrete Event Systems and Its Applications to Power Distribution Networks. *IEEE Trans. Control Netw. Syst.* **2017**, *4*, 146–158. [CrossRef]
21. Grichi, H.; Mosbahi, O.; Khalgui, M.; Li, Z. An Extended Object Constraint Language for Adaptive Discrete Event Systems With Application to Reconfigurable Wireless Sensor Networks. *IEEE Trans. Syst. Man Cybern. Syst.* **2020**, *50*, 3562–3576. [CrossRef]
22. Vieira, A.D.; Santos, E.A.P.; de Queiroz, M.H.; Leal, A.B.; de Paula Neto, A.D.; Cury, J.E.R. A Method for PLC Implementation of Supervisory Control of Discrete Event Systems. *IEEE Trans. Control Syst. Technol.* **2017**, *25*, 175–191. [CrossRef]

23. Ramadge, P.J.; Wonham, W.M. Supervisory Control of a Class of Discrete Event Processes. *SIAM J. Control Optim.* **1987**, *25*, 206–230. [CrossRef]
24. Lin, F.; Wonham, W.M. On observability of discrete-event systems. *Inf. Sci.* **1988**, *44*, 173–198. [CrossRef]
25. Heymann, M.; Lin, F. On-line control of partially observed discrete event systems. *Discret. Event Dyn. Syst. Theory Appl.* **1994**, *4*, 221–236. [CrossRef]
26. Hadjalouane, N.B.; Lafortune, S.; Lin, F. Centralized and distributed algorithms for on-line synthesis of maximal control policies under partial observation. *Discret. Event Dyn. Syst. Theory Appl.* **1996**, *6*, 379–427. [CrossRef]
27. Takai, S.; Ushio, T. Effective computation of an $L_m(G)$-closed, controllable, and observable sublanguage arising in supervisory control. *Syst. Control Lett.* **2002**, *49*, 191–200. [CrossRef]
28. Cai, K.; Zhang, R.; Wonham, W.M. Relative Observability of Discrete-Event Systems and Its Supremal Sublanguages. *IEEE Trans. Autom. Control* **2015**, *60*, 659–670. [CrossRef]
29. Yin, X.; Lafortune, S. Synthesis of Maximally Permissive Supervisors for Partially-Observed Discrete-Event Systems. *IEEE Trans. Autom. Control* **2016**, *61*, 1239–1254. [CrossRef]
30. Alves, M.V.S.; Carvalho, L.K.; Basilio, J.C. New Algorithms for Verification of Relative Observability and Computation of Supremal Relatively Observable Sublanguage. *IEEE Trans. Autom. Control* **2017**, *62*, 5902–5908. [CrossRef]
31. Hou, Y.; Wang, W.; Zang, Y.; Lin, F.; Yu, M.; Gong, C. Relative Network Observability and Its Relation With Network Observability. *IEEE Trans. Autom. Control* **2020**, *65*, 3584–3591. [CrossRef]
32. Ji, Y.; Yin, X.; Lafortune, S. Local Mean Payoff Supervisory Control for Discrete Event Systems. *IEEE Trans. Autom. Control* **2021**. [CrossRef]
33. Zhao, X.; Wang, J.; Chen, Y.; Yin, G. Multi-objective Cooperative Scheduling of CAVs at Non-Signalized Intersection. In Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, USA, 4–7 November 2018; pp. 3314–3319.
34. Levin, M.W.; Rey, D. Conflict-point formulation of intersection control for autonomous vehicles. *Transp. Res. Part Emerg. Technol.* **2017**, *85*, 528–547. [CrossRef]
35. Mirheli, A.; Hajibabai, L.; Hajbabaie, A. Development of a signal-head-free intersection control logic in a fully connected and autonomous vehicle environment. *Transp. Res. Part Emerg. Technol.* **2018**, *92*, 412–425. [CrossRef]
36. Bichiou, Y.; Rakha, H.A. Developing an Optimal Intersection Control System for Automated Connected Vehicles. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 1908–1916. [CrossRef]
37. Kamal, M.A.S.; Imura, J.i.; Hayakawa, T.; Ohata, A.; Aihara, K. A Vehicle-Intersection Coordination Scheme for Smooth Flows of Traffic Without Using Traffic Lights. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 1136–1147. [CrossRef]
38. Liu, C.; Lin, C.W.; Shiraishi, S.; Tomizuka, M. Distributed Conflict Resolution for Connected Autonomous Vehicles. *IEEE Trans. Intell. Veh.* **2018**, *3*, 18–29. [CrossRef]
39. Pandit, K.; Ghosal, D.; Zhang, H.M.; Chuah, C.N. Adaptive traffic signal control with vehicular ad hoc networks. *IEEE Trans. Veh. Technol.* **2013**, *62*, 1459–1471. [CrossRef]
40. Huang, Y.; Weng, Y.; Zhou, M. Modular design of urban traffic-light control systems based on synchronized timed Petri nets. *IEEE Trans. Intell. Transp. Syst.* **2013**, *15*, 530–539. [CrossRef]
41. Wang, J.; Yan, J.; Li, L. Microscopic Modeling of a Signalized Traffic Intersection Using Timed Petri Nets. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 305–312. [CrossRef]
42. Carlino, D.; Boyles, S.D.; Stone, P. Auction-based autonomous intersection management. In Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), Hague, The Netherlands, 6–9 October 2013; pp. 529–534.
43. Bashiri, M.; Fleming, C.H. A platoon-based intersection management system for autonomous vehicles. In Proceedings of the 2017 IEEE Intelligent Vehicles Symposium (IV), Los Angeles, CA, USA, 11–14 June 2017; pp. 667–672.
44. Du, Z.; Homchaudhuri, B.; Pisu, P. Hierarchical distributed coordination strategy of connected and automated vehicles at multiple intersections. *J. Intell. Transp. Syst.* **2018**, *22*, 144–158. [CrossRef]
45. Ahmane, M.; Abbas-Turki, A.; Perronnet, F.; Jia, W.; Moudni, A.E.; Buisson, J.; Zeo, R. Modeling and controlling an isolated urban intersection based on cooperative vehicles. *Transp. Res. Part Emerg. Technol.* **2013**, *28*, 44–62. [CrossRef]
46. Li, B.; Zhang, Y.; Zhang, Y.; Jia, N.; Ge, Y. Near-Optimal Online Motion Planning of Connected and Automated Vehicles at a Signal-Free and Lane-Free Intersection. In Proceedings of the 2018 IEEE Intelligent Vehicles Symposium (IV), Suzhou, China, 26–30 June 2018; pp. 1432–1437.
47. Li, B.; Zhang, Y. Fault-Tolerant Cooperative Motion Planning of Connected and Automated Vehicles at a Signal-Free and Lane-Free Intersection. *IFAC PapersOnLine* **2018**, *51*, 60–67. [CrossRef]
48. Li, B.; Zhang, Y.; Acarman, T.; Ouyang, Y.; Yaman, C.; Wang, Y. Lane-free Autonomous Intersection Management: A Batch-processing Framework Integrating Reservation-based and Planning-based Methods. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Philadelphia, PA, USA, 23–27 May 2021; pp. 7915–7921.
49. Zohdy, I.H.; Rakha, H.A. Game theory algorithm for intersection-based cooperative adaptive cruise control (CACC) systems. In Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, AK, USA, 16–19 September 2012; pp. 1097–1102.
50. Wuthishuwong, C.; Traechtler, A.; Bruns, T. Safe trajectory planning for autonomous intersection management by using vehicle to infrastructure communication. *EURASIP J. Wirel. Commun. Netw.* **2015**, *2015*, 33. [CrossRef]

51. Zohdy, I.H.; Rakha, H.A. Intersection management via vehicle connectivity: The intersection cooperative adaptive cruise control system concept. *J. Intell. Transp. Syst.* **2016**, *20*, 17–32. [CrossRef]
52. Fayazi, S.A.; Vahidi, A.; Luckow, A. Optimal scheduling of autonomous vehicle arrivals at intelligent intersections via MILP. In Proceedings of the American Control Conference, Seattle, WA, USA, 24–26 May 2017; pp. 4920–4925.
53. Cassandras, C.G.; Lafortune, S. *Introduction to Discrete Event Systems*, 2nd ed.; Springer: New York, NY, USA, 2007.
54. Tatsumoto, Y.; Shiraishi, M.; Cai, K.; Lin, Z. Application of online supervisory control of discrete-event systems to multi-robot warehouse automation. *Control Eng. Pract.* **2018**, *81*, 97–104. [CrossRef]
55. Deplano, D.; Franceschelli, M.; Ware, S.; Su, R.; Giua, A. A discrete event formulation for multi-robot collision avoidance on pre-planned trajectories. *IEEE Access* **2020**, *8*, 92637–92646. [CrossRef]