Discrete Event Dynamic Systems
https://doi.org/10.1007/s10626-020-00336-8

Contention-resolving model predictive control for an intelligent intersection traffic model

Ningshi Yao[1] · **Fumin Zhang**[1]

Received: 1 March 2020 / Accepted: 29 December 2020 / Published online: 01 March 2021

Abstract

We address the problem of optimally scheduling automated vehicles crossing an intelligent intersection by assigning vehicles with priorities and desired speed. An idealized intersection traffic model is established for the development and verification of the required algorithms. We formulate the intersection scheduling problem as a mixed integer programming (or MIP) problem which co-designs the priority and traveling speed for each vehicle. The co-design aims to minimize the vehicle waiting time at the intersection area, under a set of safety constraints. We derived a contention-resolving model predictive control (or MPC) algorithm to dynamically assign priorities and compute the vehicles' traveling speeds. A branch cost formulation is proposed for the decision tree constructed by contention-resolving MPC based on time instants when collisions might occur among vehicles. Based on the priority assignments, a decentralized control law is designed to control each vehicle to travel with an optimal speed given a specific priority assignment. The optimal priority assignment can be determined by searching the lowest cost path in the decision tree. The solution computed by contention-resolving MPC is proved to be optimal given the condition of immediate access (or CIA) required in real-time scheduling. The effectiveness of the proposed method is verified through simulation and compared with the first-come-first-serve (or FCFS) and highest-speed-first (or HSF) scheduling strategies.

Keywords Intersection scheduling · Model predictive control · Mixed integer optimization · Optimal control · Event triggered system

This article belongs to the Topical Collection: *Smart Cities*
Guest Editors: (Samuel) Qing-Shan Jia, Mariagrazia Dotoli, and Qian-chuan Zhao

✉ Fumin Zhang
fumin@gatech.edu

Ningshi Yao
nyao6@gatech.edu

[1] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30308, USA

1 Introduction

Urban cities are becoming overcrowded with automobiles, leading to growing traffic congestion at traffic intersections. The traffic light has been the most commonly used device for intersection scheduling since 1868. However, while traffic lights ensure the safety of conflicting movements at intersections, they also cause increased delays, fuel consumption, tailpipe emissions and even fatal accidents (Dimitrakopoulos and Demestichas 2010). Frequent stops and starts caused by traffic lights also frustrate drivers and passengers. Smarter intersection scheduling is needed to better control vehicles at intersections (Rios-Torres and Malikopoulos 2017), which is part of an overall intelligent transportation and smart cities for the future.

Connected and automated vehicles provide significant new opportunities for improving intersection efficiency. A recent study showed that changing the intersection scheduling from traffic lights to coordinating automated vehicles has the potential of doubling the intersection capacity and reducing traffic delays (Tachet et al. 2016). With the introduction of vehicle-to-vehicle as well as vehicle-to-infrastructure communication, automated vehicles can assist drivers with better decision making and reduce fuel consumption, emissions, and traffic congestion. Numerous research efforts have explored the scheduling and control of automated vehicles (Ahn et al. 2014; Zhang et al. 2015, 2017; Dallal et al. 2017; Meng et al. 2018). Many works assume that the arrival times of automated vehicles at an intersection satisfy a certain random process. Then based on the arrival times, they utilize a first-come-first-serve (or FCFS) scheduling mechanism, so that controllers can be designed to coordinate the crossing speed of vehicles (Lee and Park 2012; Zhang et al. 2016; Malikopoulos et al. 2018). However, the FCFS mechanism may lead to poor scheduling and possible congestion. For instance, FCFS schemes can give crossing orders in which several faster vehicles must slow down to favor a slow vehicle, which may not be optimal in the context of total traveling time or energy consumption. FCFS is conservative in the sense that it prevents the intersection scheduler from reordering the entrance of automated vehicles to the intersection. In those cases, the highest-speed-first (or HSF) scheduling, which schedules the vehicles with higher speed to pass the intersection first, is another strategy for intersection scheduling.

Optimization-based approaches have been proposed to compute the optimal schedule for coordinating automated vehicles. The works in Jiao et al. (2014) and Lu and Kim (2017) utilized a genetic algorithm to dynamically coordinate traffic at intersections and their results were verified through simulation using real traffic data. In Hult et al. (2018), the authors used mixed integer quadratic program (or MIQP) to compute an approximate solution to schedule the order of vehicles crossing the intersection. In Fayazi and Vahidi (2018), the intersection scheduling problem was formulated as a mixed integer linear program (or MILP) problem, and was solved by the IBM CPLEX optimization package. Although these methods can obtain an optimal or a local optimal solution for intersection scheduling, the major disadvantage is the computation requirement. The optimization problem formulated in intersection scheduling is usually nonlinear and non-convex, which is inherently difficult to solve and takes long time for the optimization solvers to find the optimal solution.

Recent works showed encouraging results by using model predictive control (or MPC) approach to coordinate vehicles (Lin et al. 2011; Frejo and Camacho 2012; Fukushima et al. 2013; Kim and Kumar 2014; Qian et al. 2015). MPC performs prediction-optimization iteratively to a predefined cost function (usually considering efficiency, ecological, and safety objectives) while receding a finite optimization time horizon. In our previous work (Yao and Zhang (2018) and Yao et al. (2017, 2019, 2020)), we proposed a contention-resolving MPC

method, a sampling based algorithm based on the crucial events when contentions occur, to co-design priorities and controls for networked control systems and traffic intersection. The co-design problem is formulated as a mixed integer programming (or MIP) problem and our method can provide a solution to this optimization problem with reduced demand on computing resources. In this paper, we extend our previous work and provide more rigorous analysis for the solution obtained by contention-resolving MPC. We discovered that the classical preemptive-repeat task model (Conway et al. 2003) can model the intersection scheduling behavior if the vehicle's earliest arrival time is defined as the task request time in scheduling theory. Based on this discovery, the main contributions in this work are summarized as follows:

1) A new analytical timing model for preemptive-repeat tasks is developed to compute timing states for the traffic intersection scheduling system. Based on the timing models, we present a sufficient and necessary condition to determine the time instants when contentions occur and compute the significant moments when a vehicle actually enters the intersection and when the intersection is not occupied by any vehicle. In Hult et al. (2018) and Fayazi and Vahidi (2018), these significant moments are determined by MIP solvers. Based on these significant moments, the priority assignment and vehicle speed control law design can be decoupled and we can construct a decision tree to efficiently search all of the possible priority assignments.
2) Enabled by the significant moments, the infinite dimensional priority and control co-design problem can be converted into a path planning problem on a decision tree. Contention-resolving MPC algorithm assigns priorities only at the significant moments when contentions occur, which are a finite number of time instances on the MPC optimization horizon, therefore, the decision tree contains a finite number of branches and each branch corresponds to one possible priority assignment. The optimal control law design is embedded in the computation of branch costs. An optimal solution of priority assignments and vehicle speed control for the original co-design problem must be a path from the root of the decision tree to one of the terminal leaves. Different from the work which use a genetic algorithm (Yan et al. 2013; Jiao et al. 2014; Lu and Kim 2017) or an MIP solver (Fayazi and Vahidi 2018; Hult et al. 2018) which can also find the optimal solution, our method searches through a greatly reduced number of possible paths in the decision tree, which provides scalable methods that eliminate the need for an exhaustive search to find the optimal solution. To the best of our knowledge, the use of a decision tree to design the schedule and control for vehicles at intersection has not previously been documented in the literature.
3) We present a new formula to compute branch costs in the decision tree that is constructed by contention-resolving MPC. The branch cost function can handle cases where a vehicle's entrance to the intersection can be delayed multiple times. In addition, we find an analytical solution to distributively compute the optimal speed for each vehicle, which tremendously reduces the computation workloads, because the branch cost can be directly calculated based on the analytical solution instead of solving another optimization problem as in Yao et al. (2020).
4) Assuming the vehicle scheduling behavior in a traffic intersection system follows basic requirements and the condition of immediate access (or CIA) in real-time scheduling theory, we prove that CIA is a necessary condition for the contention-resolving MPC to find the global optimal solution for the formulated scheduling and control co-design optimization problem. In addition, we also study the case where CIA assumption is violated for the intersection scheduling. We provide both numerical examples and

theoretical analysis to discuss the optimality of the solution computed by contention-resolving MPC without CIA condition. This contribution provides more insights about when the contention-resolving MPC algorithm can find the optimal solution.

Comparing with our previous works, the CIA condition is a new discovery which has not been addressed in any of our previous work. Moreover, in work Yao et al. (2019), we only compared the solution computed by contention-resolving MPC with FCFS and illustrated how our method can find a better solution than FCFS, but the performance was the same as the priority assignment under the HSF strategy for the specific case studied. In this paper, the contention-resolving MPC method is further compared with HSF scheduling strategy in more cases. Simulation results illustrate significant improvements using our proposed method compared to *both* FCFS and HSF.

This paper is organized as follows. In Section 2, the coupled priority assignment and MPC design problem is formulated. In Section 3, we introduce an analytical timing model to compute the significant moments of the crucial events. In Section 4, we present the contention-resolving MPC algorithm to design priority assignments and vehicle speed control. Simulated case study results using the proposed method are presented in Section 5. Section 6 is the conclusion.

2 Problem formulation

We propose an idealized traffic model at a simplified intelligent intersection to serve as the launchpad for the development of control and scheduling co-design methods. Consider N automated vehicles traveling along a single lane closed track that has a "figure-eight" shape as shown in Fig. 1. Vehicles in the closed track travel in the direction marked by the arrows in the middle of the lanes. The track consists of a shaded intersection area where two straight line tracks intersects. The four ends of the intersections are marked as two entrances and two exits depending on the direction of travel. Each exit is connected back to an entrance by a circular arc. Let S represent the length of the straight line between an entrance and the corresponding exit, D represent the length of each of the two circular arcs, and L represent the length of a vehicle. Note that there is no restrictions on the radius of the circular arc. The circular arc can also be replaced by other shapes (no self intersection) with equal length that connect an exit with an entrance.

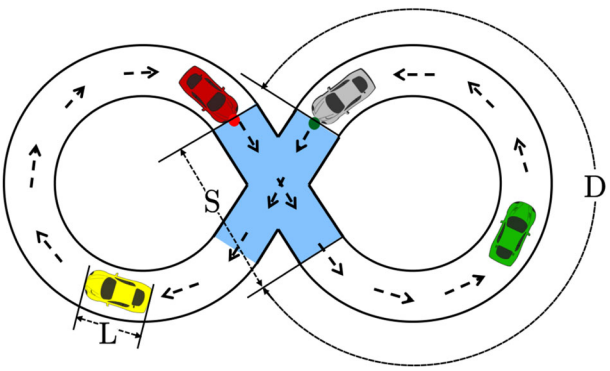


Fig. 1 Example of a one lane intersection. Vehicles follow directions indicated by arrows

The motion of the i-th vehicle, for $i = 1, ..., N$, is modeled as

$$\dot{x}_i(t) = u_i(t),\ x_i(t_0) = x_i^0 \tag{1}$$

where $x_i(t)$ and $u_i(t)$ denote the position and speed of vehicle i respectively at each time t. The initial position is x_i^0, where $x_i^0 = 0$ is when the front end of a vehicle is at one of the two entrances of the intersection, which is assigned to each vehicle. The orientation of vehicle is assumed to be automatically controlled by a heading controller onboard the vehicle, which is not considered in this paper. We assume that vehicle speed is continuous and can be directly controlled outside of the intersection area, but the value of the vehicle speed should be bounded as

$$0 \le u_i(t) \le u_{i,\max} \tag{2}$$

where $u_{i,\max} > 0$ is the maximum speed limit. To further simplify the problem setup, we require each vehicle passes through the intersection with a constant speed $u_{i,\text{int}}$ satisfying $u_{i,\text{int}} < u_{i,\max}$ for all i. The speeds outside of the intersection can be controlled and can have sudden jumps from $u_i(t)$ to $u_{i,\text{int}}$ when vehicles arrive at the intersection, and from $u_{i,\text{int}}$ to $u_i(t)$ when vehicles exit the intersection.

When the position of any portion of a vehicle is in the intersection area, the intersection is considered occupied by that vehicle. When multiple vehicles occupy the intersection at the same time, a contention occurs. We make the following assumption when a contention occurs:

Assumption 1 *At any given time, only one vehicle can occupy the intersection.*

This assumption guarantees that no collisions occur among vehicles when they are passing the intersection.

Remark 1 Even though the "figure eight" track in this paper may appear to be oversimplified and limited for real-world traffic intersections, we have found it quite useful for the design of the optimal priorities and control laws. Such idealized models are not untypical in control systems research, such as the broadly adopted "inverted pendulum" models. The main advantage that the idealized intersection traffic model provides is that all the timings of contention events are deterministic and hence predictable with an analytical model we proposed. Before we can address real world traffic where the arrival of vehicles are in general stochastic, we may need to completely solve the co-design problem in such a deterministic setting. We will discuss possible extensions in the "conclusions and future work" section.

2.1 Priority-based scheduling

A *half cycle* is a duration of time between the event when a vehicle's front end enters an intersection through one entrance and the next event when the same vehicle enters the intersection again, but through a different entrance. Let $\alpha_i[k_i]$ denote the time when vehicle i enters the intersection after having traveled through k_i half cycles. For any index k_i, C_i is the amount of time from the instant when the front end of the vehicle arrives at one entrance to the time instant when the rear end of the vehicle i leaves the intersection throught the corresponding exit, i.e. $C_i = \frac{\text{S+L}}{u_{i,\text{int}}}$. An time $\gamma_i[k_i]$ is the time instant when the rear end of vehicle i leaves the intersection for the k_i-th half cycle, satisfying

$$\gamma_i[k_i] = \alpha_i[k_i] + C_i. \tag{3}$$

When there is no contention among vehicles, it is trivial that all the vehicles should travel with their maximal speed in order to minimize the total traveling time. The following equation needs to be satisfied if no contention occurs:

$$\alpha_i[k_i] = \gamma_i[k_i - 1] + T_i, \tag{4}$$

where T_i is the least amount of time for vehicle i to travel through the k_i-th half cycle, namely,

$$T_i = \frac{\mathrm{D} - \mathrm{L}}{u_{i,\max}}. \tag{5}$$

When a contention occurs to vehicle i while it is passing the intersection for the k_i-th time, Eq. 4 will not hold because it may be interrupted by other vehicles. We introduce the delay variable $\delta_i[k_i]$ to represented the total amount of time that vehicle i is delayed caused by yielding to other vehicles for the intersection. We also define $\tilde{\alpha}_i[k_i]$ as the earliest time that vehicle i can arrive at the intersection, i.e., the vehicle travels with its maximal speed $u_{i,\max}$ without considering possible contentions with other vehicles. The initial values for $\tilde{\alpha}_i[1]$ are $x_i^0/u_{i,\max}$. The delayed arrival time and the earliest arrival time then satisfy the following equations:

$$\alpha_i[k_i] = \tilde{\alpha}_i[k_i] + \delta_i[k_i],\ \ \tilde{\alpha}_i[k_i + 1] = \alpha_i[k_i] + C_i + T_i. \tag{6}$$

Using the concept of the earliest arrival time, we can then mathematically define the crucial event of contention between vehicle i and j.

Definition 1 If there exist indices i and j such that the intersection between two time intervals, $[\tilde{\alpha}_i[k_i], \tilde{\alpha}_i[k_i] + C_i)$ and $[\tilde{\alpha}_j[k_j], \tilde{\alpha}_j[k_j] + C_j)$, is not empty, then a contention between vehicle i and vehicle j will occur at a moment during the intersection between the two time intervals.

In other words, when there is no delay, if the intersection occupation time of vehicle i overlaps with that of vehicle j, then a contention occurs between vehicles i and j. One of the vehicles needs to be delayed to avoid the contention.

When contentions occur, priorities are needed to determine which vehicle enters the intersection first. Each vehicle i is assigned a unique priority number $p_i(t)$, in which case contentions can be resolved by comparing the priorities p_i among all vehicles that are competing for the access to the intersection.

Definition 2 A priority assignment is a tuple $\mathbf{P}(t) = (p_1(t), ..., p_i(t), ..., p_N(t)) \in \mathcal{P}(\{1, ..., N\})$, where $p_i(t)$ is the priority assigned to vehicle i at time t and such that for each i and j in $\{1, ..., N\}$, we have $p_i(t) < p_j(t)$ if and only if vehicle i is assigned higher priority than vehicle j at t. Here $\mathcal{P}(\{1, ..., N\})$ is the set of all permutations of $\{1, ..., N\}$, so the value of $p_i(t)$ is a positive integer in $\{1, \ldots, N\}$, such that $p_i(t) \neq p_j(t)$ if $i \neq j$.

When a contention occurs and a specific priority assignment is given, we make the following assumption.

Assumption 2 *When a contention occurs at time t, the vehicle with the smallest $p_i(t)$ gets access to and enter the intersection at time t.*

This assumption follows the convention in the scheduling literature of giving smaller numbers to the higher prioritized tasks (Conway et al. 2003). Based on the Definition 2, each vehicle has a unique priority number. Therefore, there exist no tie among the priority assignments when a contention occurs.

The intersection access times of lower prioritized vehicles are delayed by higher prioritized vehicles. A lower prioritized vehicle can enter the intersection right *at* or *after* the time instant when all the higher prioritized contended vehicles pass the intersection. We make the following assumption that

Assumption 3 (the CIA Condition) *If a contention occurs at time t and $p_i(t)+1=p_j(t)$, then vehicle j enters the intersection at time $\gamma_i[k_i]$, where k_i is the number of half cycles vehicles i has passed through during the time interval $[t_0, t]$.*

With Assumption 3, the lower prioritized vehicle j cannot enter the intersection *after* the time instant $\gamma_i[k_i]$. This assumption is also used in the priority-based real-time scheduling mechanism from Conway et al. (2003), where no inserted idle time should be allowed if there are one or more tasks waiting to use the shared resource. FCFS and HSF are also based on this assumption. For the convenience of later references, we call this assumption the *condition of immediate access (or CIA)*. We will show that the CIA is a necessary condition for finding a global optimal solution for the co-design problem.

Consider the example in Fig. 2 where the earliest arrival times of all three vehicles are the same and their intersection occupation time interval overlaps with each other, which means a contention occurs. Let the priority assignment be $p_1(\tilde{\alpha}_1[1])=1$, $p_2(\tilde{\alpha}_2[1])=2$ and

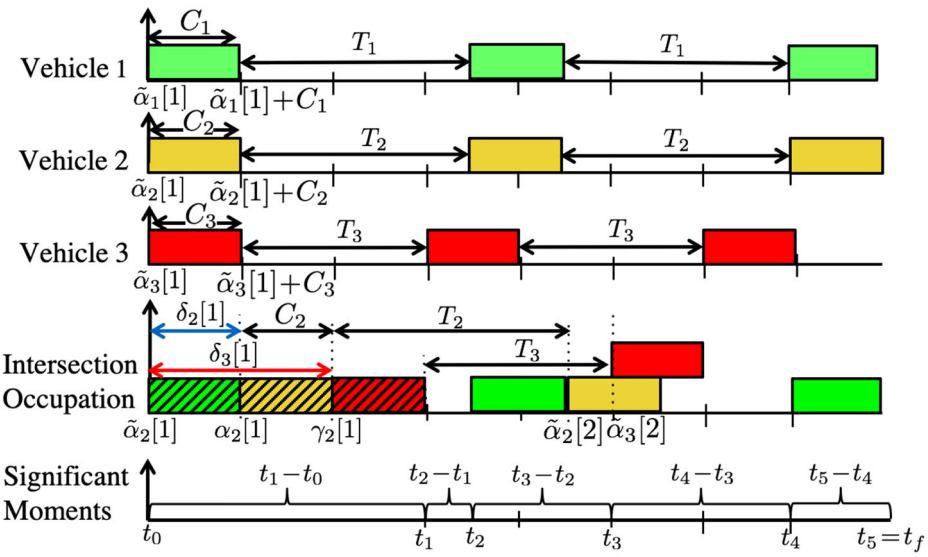


Fig. 2 Illustration of scheduling three vehicles. The upper three sub-figures show the earliest arrivals for each vehicle when contentions are not considered. The forth sub-figure shows the intersection occupation time after priorities are assigned to resolve the contention that occurs at $\tilde{\alpha}_2[1]$. The colored rectangles represent intersection occupation intervals when there is no delay and the shaded colored rectangles represent the true intersection occupation intervals. The bottom sub-figure shows the significant moments t_w, which will be introduced in Section 3

$p_3(\tilde{\alpha}_3[1]) = 3$. Because of the intersection occupation of vehicle 1, vehicle 2 has a time delay $\delta_2[1]$, shown by the blue arrow. The time instants $\alpha_2[1]$ and $\tilde{\alpha}_2[2]$ are changed according to Eq. 6. Also, because of the intersection occupation of vehicle 1 and 2, vehicle 3 has the longest time delay $\delta_3[1]$, shown by the red arrow. The time delay $\delta_3[1]$ increases if vehicle 3 travels with a lower speed between time $\tilde{\alpha}_2[1]$ and $\gamma_2[1]$. In other words, the value of $\delta_i[k_i]$ is a function of vehicle speed $u_i(t)$ satisfying $\int_{\gamma_i[k_i-1]}^{\gamma_i[k_i-1]+T_i+\delta_i[k_i]} u_i(t)\mathrm{d}t = \mathrm{D} - \mathrm{L}$. If $u_i(t)$ is a constant within time interval $[\gamma_i[k_i-1],\ \gamma_i[k_i-1]+T_i+\delta_i[k_i]]$, then $\delta[k_i] = (\mathrm{D} - \mathrm{L})/\mathrm{u_i(t)}-\mathrm{T_i}$. The time delays $\delta_i[k_i]$ also depend on priority assignments $\mathbf{P}(t)$. One example is that if we exchange the priority assignments between system 1 and 3, then vehicle 1 has the longest time delay.

In summary, the time delay variables $\delta_i[k_i]$ are implicit functions of C_i, T_i, $u_i(t)$ and $\mathbf{P}(t)$. In Section 3, we will present a timing model which can accurately compute $\delta_i[k_i]$ and update $\alpha_i[k_i]$ given a specific priority assignment.

2.2 Formulation of model predictive control

We formulate a contention-resolving model predictive control problem to compute optimal priority assignments $\mathbf{P}^*(t) = (p_1^*(t), ..., p_N^*(t))$ and an optimal vehicle speed $\mathbf{u}^*(t) = (u_1^*(t), ...u_N^*(t))$ on a time interval $[t_0, t_f]$. The times t_0 and t_f are the starting and ending points of the MPC time horizon, respectively, and t_0 and t_f will move forward in time when the MPC is initiated. Given initial states $\mathbf{x}(t_0) = (x_1(t_0), ..., x_N(t_0))$ and initial controls $\mathbf{u}(t_0) = (u_1(t_0), ..., u_N(t_0))$, the co-design method is to find values for the optimal $\mathbf{P}^*(t)$ and $\mathbf{u}^*(t)$ by solving the optimization problem

$$\min_{\mathbf{P}(t),\mathbf{u}(t)} \sum_{i=1}^{N}\sum_{k_i=1}^{K_i} \int_{\gamma_i[k_i-1](\mathbf{P}(t),\mathbf{u}(t))}^{\alpha_i[k_i](\mathbf{P}(t),\mathbf{u}(t))} \frac{1}{2}\left[u_{i,\max} - u_i(t)\right]^2 \mathrm{d}t \tag{7}$$

$$\text{s.t. (1), (2) } \forall t \in [t_0, t_f], \text{ (3), (6),}$$

$$\delta_i[k_i] \le \Delta t, \tag{8}$$

$$x_i(\gamma_i[k_i-1]) = \mathrm{S}+\mathrm{L}+(k_i-1)\cdot(\mathrm{S}+\mathrm{D}) + x_i^0,$$

$$x_i(\alpha_i[k_i]) = k_i \cdot (\mathrm{S}+\mathrm{D}) + x_i^0 \text{ for all } i \text{ and } k_i \tag{9}$$

where K_i is the largest index of the half cycle which vehicle i has traveled satisfying $\gamma_i[K_i] \le t_f$. The notations $\gamma_i[k_i-1](\mathbf{P}(t), \mathbf{u}(t))$ and $\alpha_i[k_i](\mathbf{P}(t), \mathbf{u}(t))$ represent that these time instants are implicit functions of priority assignment $\mathbf{P}(t)$ and vehicle speed $\mathbf{u}(t)$. The cost function aims to increase the speed as much as possible to increase the intersection capacity. If a contention happens and a vehicle needs to slow down or stop, then the cost increases. The interval $[\alpha[k_i], \gamma_i[k_i]]$ is not included in the formulation because $u_i(t)$ is fixed to be $u_{i,\mathrm{int}}$. A set of constraints (1), (2), (3) and (6) need to be satisfied for all times $t \in [t_0, t_f]$. To ensure fairness, Eq. 7 represents that we require all the time delay variables $\delta_i[k_i]$ to be less than or equal to a constant Δt, where Δt is the maximal time delay that can be tolerated by a vehicle. Equation 8 is the boundary condition by the definitions of $\gamma_i[k_i]$ and $\alpha_i[k_i]$.

Since $\mathbf{u}(t)$ is a vector of real numbers and $\mathbf{P}(t)$ is a vector of integers at each time t, the contention-resolving MPC problem is a mixed integer optimization problem (or MIP). It is a nonlinear and non-convex optimization problem that is difficult to solve. And the MIP associated with contention-resolving MPC is hard to be solved using existing optimization techniques, for two reasons. First, the priority assignments $p_i(t)$ cannot be replaced by constant real numbers, because the relaxation will not add new freedom to the search for

optimal solutions. Second, the cost function for each i is not an explicit function of the priority assignment $\mathbf{P}(t)$, hence convex optimization cannot be applied.

3 Analytical timing model

Even though the traffic system evolves continuously in time, due to the constrained occupation of the intersection, there are certain moments in time that are more significant than other moments. The earliest arrival time $\tilde{\alpha}_i[k_i]$, the intersection entrance time $\alpha_i[k_i]$, and intersection exit time $\gamma_i[k_i]$ are called significant moments. In our proposed framework of contention-resolving MPC, the earliest arrival time $\tilde{\alpha}_i[k_i]$ will initiate or re-initiate the computation of contention-resolving MPC controllers to determine the control effort $u_i(t)$ and the priority value $p_i(t)$ for all i. And $\gamma_i[k_i]$ and $\alpha_i[k_i]$ are the lower and upper integral limits of the cost function of MPC. In order to obtain the significant moments, it is necessary to compute $\delta_i[k_i]$, which is not easy since we need to consider how many vehicles are competing for the intersection and whether they will be delayed. We leverage the task scheduling theory and model the prioritized intersection scheduling as one of the preemptive-repeat problems.

3.1 Preemptive-repeat tasks

In task scheduling theory (Conway et al. 2003), if a task's occupation of a shared resource can be interrupted by the arrival of another task, then the task is called a *preemptive* task. Suppose the interrupted task needs to occupy the shared resource for a length of time C. If the task can continue its occupation of the resource until a total occupation time of C, then the task belongs to the type of *preemptive-resume*. On the other hand, if the task has to restart its occupation of the resource after it is preempted, regardless how much time it has occupied the resource before the preemption, then the task belongs to the type of *preemptive-repeat*. A preemptive-repeat task will finish request the shared resource only when it occupies the shared resource for an uninterrupted time window of length C.

We model the earliest arrival time $\tilde{\alpha}_i[k_i]$ as the resource requesting time of a task in scheduling theory and define the total process time of this task to be the intersection occupation time C_i. If there are no other vehicles which have the same earliest arrival time as $\tilde{\alpha}_i[k_i]$, this task can start processing. If there are no other task j having contention with task i, i.e. no contention occurs to vehicle i within its k_i-th half cycle, then there is no preemption and task i can finish the processing. If there exists a task j such that i and j have a contention and j is assigned a higher priority than i, then the processing of task i is preempted by j and the whole processing of task i will be repeated at $\gamma_j[k_j]$. Task i may be preempted by the arrival of another higher prioritized task again until there is an uninterrupted time window of length C_i for task i. This time window is the true intersection occupation time interval of vehicle i. The preemption mechanism guarantees that the order of entering the intersection follows the priority assignment and satisfies the CIA assumption. The task repeating mechanism guarantees the physical constraint that there is no interruption once a vehicle enters and occupies the intersection.

3.2 The timing states

In our previous work (Shi and Zhang 2015; Shi et al. 2017), we developed the significant moment analysis (or SMA) method and established timing models that can compute $\delta_i[k_i]$

for the preemptive-resume tasks and the non-preemptive tasks, which are different from the preemptive-repeat tasks for the traffic intersection scheduling problem. Here we use SMA to derive a new analytical timing model for the preemptive-repeat tasks, which has not been presented in Shi and Zhang (2015) and Shi et al. (2017).

Definition 3 At each time t within the time horizon $[t_0, t_f]$ of contention-resolving MPC, we define the timing state variable $Z(t) = (D(t), R(t), O(t))$ as follows. The dynamic deadline variable is $D(t) = (d_1(t), ..., d_i(t), ..., d_N(t))$, where $d_i(t)$ denotes how long after time t the next request of task i will be generated. The remaining time variable is $R(t) = (r_1(t), ..., r_i(t), ..., r_N(t))$, where $r_i(t)$ is the amount of time needed after time t to complete the processing of current task i. The dynamic response time variable is $O(t) = (o_1(t), ..., o_i(t), ..., o_N(t))$, where $o_i(t)$ denotes the length of time from the time when the most recent request from task i is generated to the minimum of (a) the time when the most recent request from task i is completed and (b) the current time t, i.e.,

$$o_i(t) = \min\{\gamma_i[k_i], t\} - \alpha_i[k_i], \text{ if } t \in [\alpha_i[k_i], \alpha_i[k_i+1]). \tag{10}$$

The initial values for the timing states $Z(t)$ are

$$d_i(t_0) = \begin{cases} T_i + C_i, & \text{if } \tilde{\alpha}_i[1] = 0 \\ 0, & \text{otherwise} \end{cases}, \quad r_i(t_0) = \begin{cases} C_i, & \text{if } \tilde{\alpha}_i[1] = 0 \\ 0, & \text{otherwise} \end{cases}, \quad \text{and } o_i(t_0) = 0.$$

3.3 Timing model for intersection scheduling

We divide $[t_0, t_f]$ into a set of disjoint sub-intervals $[t_w, t_{w+1})$ such that resource requests from any task are only generated at t_w, but not at any other time point within (t_w, t_{w+1}). Hence t_w equals to $\tilde{\alpha}_i[k_i]$ for some i. Since the previous request from task i has to be fulfilled before a new request can be generated, the dynamic deadline must satisfies $d_i(t_w^-) = 0$ where t_w^- denotes $t \to t_w$ while $t < t_w$ e.g. the limit when t approaches t_w from the left. If the values of the dynamic deadlines are known at t_w, then t_{w+1} can be computed by

$$t_{w+1} = t_w + \min\left\{d_1(t_w), ..., d_N(t_w), t_f - t_w\right\}. \tag{11}$$

An illustration of computed t_w is shown in Fig. 2.

The evolution of $Z(t)$ within any sub-interval $[t_w, t_{w+1})$ can be derived as follows:

At time $t = t_w$: We first discuss the value of $[d_i(t), r_i(t), o_i(t)]$ at times t_w. For any task i, the values of the state vector at time t_w, i.e. $[d_i(t_w), r_i(t_w), o_i(t_w)]$, depend on whether an new request of task i is generated at t_w.

(1) If task i generates a new request at t_w, then we have that $d_i(t_w^-) = 0$. In this case, the state vector $[d_i(t), r_i(t), o_i(t)]$ is updated as

$$d_i(t_w) = T_i + C_i, \; r_i(t_w) = C_i, \; o_i(t_w) = 0. \tag{12}$$

(2) If task i does not generate a new request at t_w, i.e. $d_i(t_w^-) \neq 0$, then there is at least a task j such that $d_j(t_w^-) = 0$, $j \neq i$. There are three different cases that need to be discussed. The first case (2a) is that task i has completed processing, i.e. $r_i(t_w^-) = 0$. The second case (2b) is that task i is being processed, i.e. $r_i(t_w^-) > 0$ and task i has higher priority than task j, i.e. $p_i(t_w^-) < p_j(t_w^-)$. Then the timing states of task i is not affected by the new request from task j. Hence, there exist no jumps for the value of timing states of task i,

$$d_i(t_w) = d_i(t_w^-), \; r_i(t_w) = r_i(t_w^-), \; o_i(t_w) = o_i(t_w^-). \tag{13}$$

The third case (2c) is that task i is being processed, i.e. $r_i(t_w^-) > 0$, but task i has lower priority than task j, i.e. $p_i(t_w^-) > p_j(t_w^-)$. In this case, the new request from task j preempts task i, so that the timing states of task i are reset to

$$d_i(t_w) = T_i + C_i,\ r_i(t_w) = C_i,\ o_i(t_w) = o_i(t_w^-). \tag{14}$$

In particular, $d_i(t_w)$ and $r_i(t_w)$ are reset to their initial values as in Eq. 12.

We define $S_{i,j} = \{t \in [t_0, t_f] : p_i(t) < p_j(t)\}$ which is a set containing all the time instants when task i is assigned with higher priority than task j. The complement of set $S_{i,j}$, denoted as $S_{i,j}^c = \{t \in [t_0, t_f] : p_i(t) > p_j(t)\}$ is the set containing all the time instants when task i is assigned with lower priority than task j. Following from Eqs. 12–14, we can express the evolution of the timing states $d_i(t)$ for task i from t_w^- to t_w as:

$$\begin{aligned} d_i(t_w) = {} & [1 - \operatorname{sgn}(d_i(t_w^-))](T_i + C_i) + \operatorname{sgn}(d_i(t_w^-))\operatorname{sgn}(r_i(t_w^-))\mathbb{1}_{S_{i,j}^c}(t_w^-)(T_i + C_i) \\ & + \operatorname{sgn}(d_i(t_w^-))\operatorname{sgn}\left[(1 - \operatorname{sgn}(r_i(t_w^-))) + \mathbb{1}_{S_{i,j}}(t_w^-)\right] d_i(t_w^-), \end{aligned} \tag{15}$$

where sgn is defined by $\operatorname{sgn}(q) = 1$ if $q \geq 0$ and $\operatorname{sgn}(q) = 0$ if $q < 0$ and the notation $\mathbb{1}_S(t)$ is defined to be 1 if a time instant $t \in S$ and 0 if $t \notin S$ for any set S. The meaning of Eq. 15 is as follows. For case (1) where $d_i(t_w^-) = 0$, we have $\operatorname{sgn}(d_i(t_w^-)) = 0$ and $1 - \operatorname{sgn}(d_i(t_w^-)) = 1$. Then the first term $[1 - \operatorname{sgn}(d_i(t_w^-))](T_i + C_i)$ in Eq. 15 equals $T_i + C_i$ while the second and third terms in Eq. 15 equal 0, which agrees with Eq. 12. For the case (2c) where $d_i(t_w^-) > 0$, $r_i(t_w^-) > 0$ and $p_i(t_w^-) > p_j(t_w^-)$, we have equivalent representations of these three conditions as $\operatorname{sgn}(d_i(t_w^-)) = 1$, $\operatorname{sgn}(r_i(t_w^-)) = 1$ and $\mathbb{1}_{S_{i,j}^c}(t_w^-) = 1$ while both $1 - \operatorname{sgn}(r_i(t_w^-)) = 0$ and $\mathbb{1}_{S_{i,j}}(t_w^-) = 0$ resulting in $\operatorname{sgn}\left[(1 - \operatorname{sgn}(r_i(t_w^-))) + \mathbb{1}_{S_{i,j}}(t_w^-)\right] = 0$ Therefore, when case (2c) occurs, the second term in Eq. 15 equals $T_i + C_i$ while the first and third terms are 0, which agrees with Eq. 14. As for the case (2a) where $d_i(t_w^-) > 0$, $r_i(t_w^-) = 0$, we have $\operatorname{sgn}(d_i(t_w^-)) = 1$ and $1 - \operatorname{sgn}(r_i(t_w^-)) = 1$. For the case (2b) where $d_i(t_w^-) > 0$, $r_i(t_w^-) > 0$ and $p_i(t_w^-) < p_j(t_w^-)$, we have $\operatorname{sgn}(d_i(t_w^-)) = 1$, $\operatorname{sgn}(r_i(t_w^-)) = 1$ and $\mathbb{1}_{S_{i,j}}(t_w^-) = 1$. Either (2a) or (2b) occurs, we can see that $\operatorname{sgn}(d_i(t_w^-)) = 1$ and $\left[(1 - \operatorname{sgn}(r_i(t_w^-))) + \mathbb{1}_{S_{i,j}}(t_w^-)\right] = 1$, resulting in $d_i(t_w) = d_i(t_w^-)$, as Eq. 13. Similarly, we can derive the evolution of the timing states $r_i(t)$ and $o_i(t)$ as

$$\begin{aligned} r_i(t_w) = {} & [1 - \operatorname{sgn}(d_i(t_w^-))]C_i + \operatorname{sgn}(d_i(t_w^-))\operatorname{sgn}(r_i(t_w^-))\mathbb{1}_{S_{i,j}^c}(t_w^-)\, C_i \\ & + \operatorname{sgn}(d_i(t_w^-))\operatorname{sgn}\left[(1 - \operatorname{sgn}(r_i(t_w^-))) + \mathbb{1}_{S_{i,j}}(t_w^-)\right] r_i(t_w^-), \end{aligned} \tag{16}$$

$$o_i(t_w) = \operatorname{sgn}(d_i(t_w^-))o_i(t_w^-), \tag{17}$$

At time $t \in (t_w, t_{w+1})$: For the deadline variable $d_i(t)$, it decreases constantly with the rate $\dot{d}_i(t) = 0$ if there are another higher prioritized tasks occupying the shared resource while task i is waiting to use the share resource, i.e., $r_i(t_w) > 0$. The decreasing rate $\dot{d}_i(t)$ equals -1 otherwise. The total amount of time when $\dot{d}_i(t) = 0$ within $[t_w, t_{w+1})$ can be computed

$$\operatorname{sgn}(r_i(t_w)) \sum_{q \in \mathrm{HP}_i(t_w)} r_q(t_w),$$

where $\mathrm{HP}_i(t_w) = \{j \in \{1,\ldots,N\} : p_j(t_w) < p_i(t_w)\}$ is the set of all indices of vehicles which have higher priorities than vehicle i at time t_w. Therefore, the deadline variable at time t is

$$d_i(t) = d_i(t_w) - \max\left\{0, t - t_w - \mathrm{sgn}(r_i(t_w)) \sum_{q \in \mathrm{HP}_i(t_w)} r_q(t_w)\right\}, \tag{18}$$

where the function max guarantees that the amount of time when $\dot{d}_i(t) = -1$ will not be a negative value. This equation is different from the equations for preemptive-resume and non-preemptive tasks published in our previous work (Yao et al. 2020). As for the remaining time and dynamic response variables, the evolution rules are

$$\begin{aligned} r_i(t) &= \max\left\{0, r_i(t_w) - \max\left\{0, t - t_w - \sum_{q \in \mathrm{HP}_i(t_w)} r_q(t_w)\right\}\right\}, \\ o_i(t) &= o_i(t_w) + \mathrm{sgn}(r_i(t_w)) \min\left\{t - t_w, r_i(t_w) + \sum_{q \in \mathrm{HP}_i(t_w)} r_q(t_w)\right\} \end{aligned} \tag{19}$$

These equations are identical to the equations for preemptive-resume tasks, which has been addressed by our previous work (Yao et al. 2020).

Combining all of the evolution rules leads to the timing model of intersection scheduling, which computes the value of $Z(t)$ at time t, given the initial state variable $Z(t_0)$, the vehicle timing parameters (C_i, T_i) for all i and a specific priority assignment $\mathbf{P}(t_0 \sim t)$, where $\mathbf{P}(t_0 \sim t)$ is a simplified notation to represent the priority assignment for all vehicles during the time interval $[t_0, t]$

$$Z(t) = \mathbb{H}(t; Z(t_0), (\tilde{\alpha}_i[k_i], C_i, T_i)_{i=1,\ldots,N}, \mathbf{P}(t_0 \sim t)). \tag{20}$$

Here we use the function $\mathbb{H}(\cdot)$ to represent the timing model, which consists of a set of analytical algebraic and differential Eq. 11 and Eqs. 15–19.

The timing model is then used to calculate the time delay $\delta_i[k_i]$ needed in Eq. 7 for the MPC problem formulation. Furthermore, $\delta_i[k_i]$ is also needed in Eq. 6 to compute $\alpha_i[k_i]$ and the $\gamma_i[k_i]$ which are needed in Eq. 8. By the definition of the variable $O(t)$, we have $\delta_i[k_i] = Z_{2N+i}(\tilde{\alpha}_i[k_i + 1]^-) - C_i$, where $Z_{2N+i}(\tilde{\alpha}_i[k_i+1]^-)$ denotes the $(2N+i)$-th element of $Z(\alpha_i[k_i+1]^-)$.

4 Contention-resolving MPC algorithm

The formulated co-design problem (7) is a nonlinear and non-convex problem that is difficult to solve. We convert this difficult problem into a path planning problem that can be solved iteratively by first generating all the possible priorities. The conversion is based on the insight that priorities only need to be assigned when a contention occurs, which only happens at the significant moments $\tilde{\alpha}_i[k_i]$. Then the optimal vehicle speed control are designed based on the assigned priorities and significant moments. In this approach, the co-design of priorities and vehicle control can be decoupled. Then using sampling based methods, such as the A-star algorithm, the priority assignment can be determined.

4.1 Construction of decision tree

We use the timing model to determine when contentions occur by checking the following Proposition,

Proposition 1 *Contention happens at time t if and only if the following conditions hold:*

$$\sum_{i=1}^{N} sgn(r_i(t)) \geq 2,\ \sum_{i=1}^{N} sgn(r_i(t^-)) \leq 1 \text{ and } t = \tilde{\alpha}_i[k_i] \text{ for some } i \text{ and } k_i \tag{21}$$

where t^- represents limit from the left.

Proof Based on Definition 3, if a vehicle i has not finished the current intersection occupation at t, then $r_i(t) > 0$ and $\text{sgn}(r_i(t)) = 1$. Since $r_i(t)$ is always non-negative, $\text{sgn}(r_i(t)) \geq 0$ for all t. Therefore, $\sum_{i=1}^{N} \text{sgn}(r_i(t)) \geq 2$ is equivalent to two or more vehicles wanting to access the intersection, which means a contention is occurring at time t. Since $\sum_{i=1}^{N} \text{sgn}(r_i(t^-)) \leq 1$ means that no contention happens at time instants before t that are close to t, the result follows. □

Based on the contention times, we can construct a decision tree. Figure 3 shows an example of decision tree. In the decision tree, each leaf represents a contention time satisfying Proposition 1. We denote the contention time by t_l^c where l is the index of its corresponding leaf. At each contention time, there are only a finite number of vehicles competing for the access to the intersection. Each possible assignment of the priority to the finite number of vehicles will produce a branch of a decision tree. The construction of the entire decision tree is not necessary for contention-resolving MPC algorithm. However, for the purpose of clearly presenting the concept for the sampling based optimization method, we now briefly describe how the tree can be fully constructed.

The decision tree construction starts from the root v_0 associated with the MPC starting time t_0. The construction is performed iteratively. During the construction, if a leaf has no branches pointing out from it, then it is called *unexpanded*. At each iteration, new branches are generated from unexpanded leaves and new leaves are generated at the end of each branch. For an unexpanded leaf l, let $\Lambda(t_l^c)$ denote the set of vehicles having contentions at a contention time t_l^c. And we define M as the number of elements of $\Lambda(t_l^c)$. And let $\mathbf{P}_m$ denote the m-th permutation in $\mathcal{P}(\{1, ..., M\})$, so $m \in \{1, 2, .., M!\}$. For leaf l, we generate $M!$ branches from it. Each branch corresponds to a unique choice of the priority assignment in $\mathcal{P}(\{1, ..., M\})$. The m-th branch expands from v_l and connects to a new leaf v_{j+m} based on $\mathbf{P}_m$, where j is the number of existing leaves in the tree before we generate new branches from leaf v_l. The contention time represented by leaf $v_{l,j+m}$ is the next contention time occurs after t_l^c scheduled by priority assignment $\mathbf{P}_m$. Different branches may end with different next contention times after t_l^c. The iterative construction terminates when the contention times of all unexpanded leaves are greater or equal to t_f. And we call these unexpanded leaves *terminal leaves* and assign t_f to them as their contention times.

An example of scheduling three vehicles for four consecutive contentions is shown in Fig. 3. The upper part of the figure shows the constructed decision tree and the lower part shows the intersection occupation scheduled by the priority assignments along the path with green arrows. Notice that in this example, the second earliest arrival of vehicle 1 is delayed by vehicle 2 at t_2^c and vehicle 3 at t_5^c. This is because at these two contention times, vehicle 1 are assigned with lower priority. This is a scenario which was not considered in our previous

work (Yao and Zhang 2018) and motivates us to propose a new formula for branch costs, which will be presented in the next section.

4.2 Branch cost

After constructing the decision tree, we define a cost for each branch. Along one branch (l, j) whose associated priority assignment is $\mathbf{P}_m$, we first calculate the significant moments $\alpha_i[k_i]$ and $\gamma_i[k_i]$ for all i and k_i such that $t_l^c \leq \alpha_i[k_i] \leq t_j^c$,

$$\begin{aligned} Z(t) &= \mathbb{H}(t;\, Z(t_l^c),\, (\tilde{\alpha}_i[k_i],\, C_i,\, T_i)_{i=1,\ldots,N},\, \mathbf{P}_m), \\ \delta_i[k_i] &= Z_{2N+i}(\tilde{\alpha}_i[k_i+1]^-) - C_i,\ \ \alpha_i[k_i] = \tilde{\alpha}_i[k_i] + \delta_i[k_i], \\ \gamma_i[k_i] &= \alpha_i[k_i] + C_i,\ \ \tilde{\alpha}_i[k_i+1] = \gamma_i[k_i] + T_i. \end{aligned} \tag{22}$$

Based on the computed $\delta_i[k_i]$ from Eq. 22, if there exists any i and k_i such that $\delta_i[k_i] > \Delta t$, then the priority assignment $\mathbf{P}_m$ is infeasible because under such priority assignment, the constraint Eq. 8 is violated. We define the corresponding branch cost $w_{l,j} = +\infty$ if $\mathbf{P}_m$ is infeasible. If $\mathbf{P}_m$ is feasible, then branch cost $w_{l,j}$ is defined as

$$w_{l,j} = \sum_{i=1}^{N} w_{l,j}^i \tag{23}$$

where $w_{l,j}^i$ is the cost of vehicle i and it can be computed by solving the following optimization problem based on the significant moments calculated along a branch. Let $\underline{k}_i$ be the

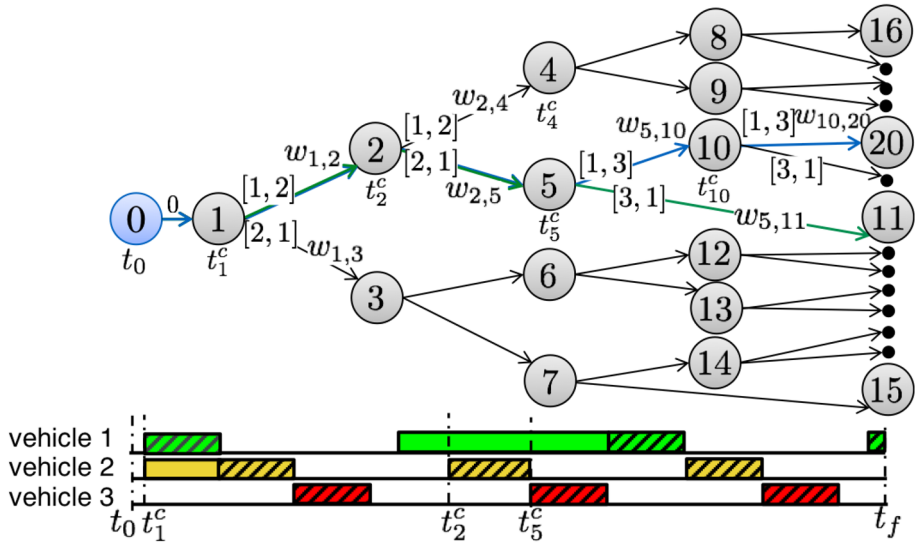


Fig. 3 Decision tree to solve the co-design problem for a finite time window. The blue circle represents the root v_0, grey circles and dots represent internal leaves. The decision tree is expanded in the direction of the arrows, which represent the branches. The colored rectangles in the lower part of the figure represent the time delay δ_i. The starting time of the colored rectangles is the earliest arrival time $\tilde{\alpha}$. The shaded colored rectangles represent the intersection occupation time of each vehicle under the assigned priorities

smallest index satisfying $\gamma_i[k_i] > t_l^c$ and $\bar{k}_i$ be the largest index satisfying $\gamma_i[k_i] \leq t_j^c$. If $\underline{k}_i \leq \bar{k}_i$, then

$$w_{l,j}^i = \sum_{k_i=\underline{k}_i}^{\bar{k}_i} \min_{u_i(t)} \int_{\gamma_i[k_i-1]}^{\alpha_i[k_i]} \frac{1}{2}\left[u_{i,\max} - u_i(t)\right]^2 \mathrm{d}t \tag{24}$$

$$\text{s.t. (1), (2), (9) and given } \gamma_i[k_i-1],\ \alpha_i[k_i]$$

where $r_i[0]$ is defined to be t_0 for all i. The meaning of Eq. 24 is as follows. If the k_i-th intersection occupation of vehicle i is completed between the contention times t_l^c and t_j^c, i.e. $\gamma_i[k_i] \in (t_l^c, t_j^c]$, then the cost of the (k_i-1)-th half cycle, traveled between $[\gamma_i[k_i-1], \alpha_i[k_i]]$, is included in the branch cost $w_{l,j}$. If no intersection occupation of vehicle i is completed within $[t_l^c, t_j^c]$, i.e. $\underline{k}_i > \bar{k}_i$, then $w_{l,j} = 0$. This branch cost formulation ensures that all costs included in one branch are determined and will not be changed by the priority assignments at or after time t_j^c. The cost of the incompleted $(\bar{k}_i+1)$-th half cycle will be included by the branches following the branch (l, j).

Figure 4 shows an illustration of the defined branch cost for the blue and green paths in Fig. 3. The different priority assignments at t_5^c cause different branch cost computations. In the blue path, the cost of vehicle 1 consists of two intervals because another contention occurs at t_{10}^c. In the green path, the cost of vehicle 1 involves only one interval because the next contention occurs after time t_f. The optimal vehicle control design is embedded in the branch cost calculation. We need to solve the optimization problem (24) to obtain the optimal control law $u_i^*(t)$. In the next section we present an analytical solution for this optimal control problem.

Remark 2 Along any arbitrary path in the decision tree, all the significant moments are deterministic and can be computed by the timing model. For any $\gamma_i[k_i]$ along this path, we can always find the consecutive contention times t_l^c and t_j^c such that $\gamma_i[k_i] \in [t_l^c, t_j^c)$ and the cost of the half cycle before $\gamma_i[k_i]$ is added in the branch cost. This guarantees that no cost is left out between branches.

The optimal vehicle control design is embedded in the branch cost calculation. We need to solve the optimization problem (24) to obtain the optimal control law $u_i^*(t)$. In the next section we present an analytical solution for this optimal control problem.

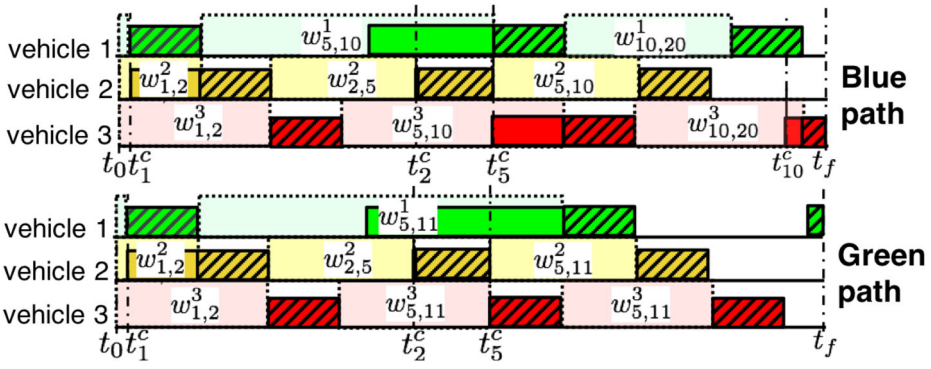


Fig. 4 Illustration of branch costs along paths. The ending time of the shaded colored rectangles represents γ_i. The dashed rectangles represent the time intervals $[\gamma_i[k_i], \alpha_i[k_i+1])$ to compute the branch costs $w_{l,j}^i$

4.3 Analytical solution of the optimal vehicle control

For a vehicle i at the time instant $\gamma_i[k_i-1]$, if this vehicle will not have any contention with other vehicles within its k_ith half cycle, then the actual arrival time $\alpha_i[k_i]=\tilde{\alpha}_i[k_i]$. Therefore, vehicle travels with its maximal speed limit $u_{i,\max}$ within the k_ith half cycle. Consider the case when vehicle i will have a contention with other vehicles, but vehicle i is assigned with the highest priority, i.e., $p_i(t_l^c)=1$. To minimize the cost in Eq. 24, vehicle i should travel through the intersection with its maximal speed $u_{i,\max}$ because it has the highest priority when contention occurs and no other vehicles can delay it from entering the intersection. If vehicle i will have a contention with other vehicles, but vehicle i is assigned with lower priority, i.e., $p_i(t_l^c)>1$, it can only enters the intersection after all vehicles with higher priorities leave the intersection area. The optimal speed for vehicle i can be computed through solving a constrained optimization problem:

$$\min_{u_i(t)} \int_{\gamma_i[k_i-1]}^{\alpha_i[k_i]} \frac{1}{2}\left[u_{i,\max}-u_i(t)\right]^2 \mathrm{d}t \quad \text{with constraints (1), (2), (9).} \tag{25}$$

We first show that this problem has feasible solutions.

Lemma 1 *Given $\gamma_i[k_i-1]$ and $\alpha_i[k_i]$, a feasible solution always exists under constraints (1), (2) and (8).*

Proof If $\alpha_i[k_i]=\tilde{\alpha}_i[k_i]$, the unique feasible solution is that a vehicle travels with the maximal speed $u_{i,\max}$. If $\alpha_i[k_i]>\tilde{\alpha}_i[k_i]$, vehicle i can travel with a lower speed, $u_i^0=(\mathrm{D}-\mathrm{L})/(\alpha_i[\mathrm{k_i}]-\gamma_i[\mathrm{k_i}-1])$. Since (3) and (6), we have $\tilde{\alpha}_i[k_i]=\alpha_i[k_i-1]+C_i+T_i=\gamma_i[k_i-1]-C_i+C_i+T_i=\gamma_i[k_i-1]+T_i$. Therefore, we have $T_i=\tilde{\alpha}_i[k_i]-\gamma_i[k_i-1]$, which gives us the result $u_{i,\max}=(\mathrm{D}-\mathrm{L})/T_i=(\mathrm{D}-\mathrm{L})/(\tilde{\alpha}_i[k_i]-\gamma_i[k_i-1])$. Because $\alpha_i[k_i]>\tilde{\alpha}_i[k_i]$, we have $u_i^0<u_{i,\max}$. Therefore, $u_i^0=(\mathrm{D}-\mathrm{L})/(\alpha_i[\mathrm{k_i}]-\gamma_i[\mathrm{k_i}-1])$ is a feasible solution which is continuous within the time interval $[\gamma_i[k_i-1],\alpha_i[k_i]]$ and satisfies (2). □

Then we can compute the optimal control law for vehicle i within the time window $(\gamma_i[k_i-1],\alpha_i[k_i])$.

Theorem 1 *The optimal solution for (24) must satisfy*

$$u_i^*(t)=\frac{\mathrm{D}-\mathrm{L}}{\alpha_i[k_i]-\gamma_i[k_i-1]}=\frac{\mathrm{D}-\mathrm{L}}{T_i+\delta_i[k_i]},\ t\in[\gamma_i[k_i-1],\alpha_i[k_i]] \text{ for some } k_i\geq 1. \tag{26}$$

Proof We use a constrained optimization argument based on Sethi and Thompson (2000). Using the cost function (24), the system dynamics (1), and the control constraints (2), it follows that for each vehicle i, the Hamiltonian is defined as

$$H_i(t,\ x_i,u_i)=\tfrac{1}{2}\left(u_{i,\max}-u_i\right)^2+\lambda_i(t)\cdot u_i+\mu_{i,1}(u_i-u_{i,\max})+\mu_{i,2}(-u_i) \tag{27}$$

where $\lambda_i(t)$ is the Lagrange multiplier, and $\mu_{i,1}\geq 0$ and $\mu_{i,2}\geq$ are the constant Kuhn Tucker multipliers for the inequality constraints $u_i-u_{i,\max}\leq 0$ and $-u_i\leq 0$. The terminal cost $\Psi_i(t,x_i,u_i)$ is 0 since there is no terminal cost in the original cost function in Eq. 24. Therefore, the Euler-Lagrange condition becomes

$$\dot{\lambda}_i=-\frac{\partial H_i}{\partial x_i}=0,\ \lambda_i(\alpha_i[k_i])=\left.\frac{\partial \Psi_i}{\partial x_i}\right|_{\alpha_i[k_i]}=0.$$

Then $\lambda_i(t)=0$ for all $t \in [\gamma_i[k_i-1], \alpha_i[k_i]]$. The necessary condition for optimality is

$$\frac{\partial H_i}{\partial u_i} = -u_{i,\max}+u_i(t)+\lambda_i(t)+\mu_{i,1}-\mu_{i,2}=0.$$

Therefore, the optimal vehicle speed is a constant given by $u_i^*(t)=u_{i,\max}-\mu_{i,1}+\mu_{i,2}$. Then (1) gives

$$x_i(t)=x_i(\gamma_i[k_i-1])+u_i^*(t)\,(t-\gamma_i[k_i-1])\,,\ \ t\in[\gamma_i[k_i-1],\alpha_i[k_i]]. \tag{28}$$

Using the boundary condition $x_i(\alpha_i[k_i]) = x_i(\gamma_i[k_i-1])+\mathrm{D}-\mathrm{L}$ from Eq. 8 and setting $t=\alpha_i[k_i]$ in Eq. 28, we then have the equality

$$x_i(\gamma_i[k_i-1])+\mathrm{D}-\mathrm{L}=x_i(\gamma_i[k_i-1])+u_i^*(t)\,(\alpha_i[k_i]-\gamma_i[k_i-1])$$

which produces the first formula for $u_i^*(t)=\frac{\mathrm{D}-\mathrm{L}}{\alpha_i[k_i]-\gamma_i[k_i-1]}$ where $k_i \geq 1$. Due to Eqs. 3 and 6, we can derive $\alpha_i[k_i]=\tilde{\alpha}_i[k_i]+\delta_i[k_i]=\alpha_i[k_i-1]+C_i+T_i+\delta_i[k_i]=\gamma_i[k_i-1]-C_i+C_i+T_i+\delta_i[k_i]=\gamma_i[k_i-1]+T_i+\delta_i[k_i]$. Therefore, we have $\alpha_i[k_i]-\gamma_i[k_i-1]=T_i+\delta_i[k_i]$, which gives us the second formula in Eq. 26. □

Theorem 1 computes the analytical solution of the optimal speed for vehicle i, given the speed of vehicle j and the significant moments from the dynamic timing model. With this solution, we can directly compute the branch cost. The pseudo code to compute each branch cost is presented by Algorithm 1. The algorithm solves the optimal control design iteratively in the order of priorities.

Algorithm 1 Branch cost.

Data: t_{l-1}^c, t_l^c, M, $\mathbf{P}_m$
Result: $\mathbf{u}^*(t)$, $w_{l,j}$
$u_i^*(t)=u_{i,\max}$, $i=1,\ldots,N$ and $t\in[t_{l-1}^c, t_l^c]$;
for $k=2:M$ **do**
    i is the index of vehicle such that $p_i=k$;
    j is the index of vehicle such that $p_j=k-1$;
    Compute $u_i^*(t)$ based on (26);
$w_{l,j}=\sum_{i=1}^{N}\sum_{k_i=\underline{k}_i}^{\bar{k}_i}\min_{u_i(t)}\int_{\gamma_i[k_i-1]}^{\alpha_i[k_i]}\frac{1}{2}\left[u_{i,\max}-u_i^*(t)\right]^2\mathrm{d}t$

4.4 Costs for searching algorithm

Based on the decision tree, the MIP problem in Section 2.2 can now be converted to the problem of finding a path from t_0 to t_f such that the whole cost along the path is lowest. Among all the paths in decision tree, the lowest cost path can be found by path planning algorithms and the priority assignments and control commands along the lowest cost path will be the solution for the MIP problem (7). However, constructing the entire decision tree would be exhaustive and unrealistic when considering a relatively large number of control systems or a long time window. we propose a search algorithm that only needs to construct a subtree of the decision tree while we are searching for the optimal path. This method is

inspired by the A-star algorithm (Hart et al. 1968) that has been widely used for online path planning in robotics, which has been found to significantly reduce computation time.

The A-star algorithm will iteratively generate and search the leaves starting from the root and terminate when it reaches a terminal leaf. To use the A-star algorithm, we define leaves in two categories: i) If a leaf has been generated and all its child leaves have been generated by the search algorithm, then we call such a leaf *closed*. ii) If a leaf has been generated and at least one of its child leaves has not been generated by the search algorithm, then we call such a leaf *open*. If a leaf is open and its parent leaf is closed, then the leaf is called a *frontier leaf*. All frontier leaves are added to a set called the *frontier list*, which keeps track of the leaves that can be expanded by the A-star algorithm. The *frontier list* is a sorted list where all the leaves in it are sorted according to a function

$$C_f(v_l) = C_g(v_l) + C_h(v_l) \tag{29}$$

from the smallest to largest value where l is the index of a leaf. The function $C_g(v_l)$ is called the stage cost, which is the sum of branch costs along the path starting from the root to the current leaf v_l and $C_h(v_l)$ is the minimal future cost from the current leaf v_l to a terminal leaf where the minimization is over all priority assignments and allowable controls.

Since the path from the root v_0 to a leaf v_l is unique, the stage cost can be computed using $C_g(v_l) = C_g(v_p) + w_{p,l}$ where p is the index of the parent leaf of v_l. For the A-star algorithm to work, an estimation $\hat{C}_h(v_l)$ of future cost (also called the heuristic cost) is needed for which $\hat{C}_h(v_l) \leq C_h(v_l)$ for all v_l, so the estimated cost $\hat{C}_f(v_l) = C_g(v_l) + \hat{C}_h(v_l)$ equals to the actual cost $C_f(v_l)$ when v_l is a terminal leaf. Since if no contention occurs, all the vehicles can travel with their maximal speed limit. Therefore, we can estimate the future cost $\hat{C}_h(v_l)$ as

$$\hat{C}_h(v_l) = 0, \tag{30}$$

During the search by the A-star algorithm, all leaves v in the *frontier list* are sorted according to the value of $\hat{C}_f(v)$, from the smallest to the largest. At each iteration, the A-star algorithm expands the leaf with the smallest $\hat{C}_f$ by generating all its child leaves and then removes the expanded leaf from the *frontier list*. All its child leaves are added to the *frontier list*. The heuristic cost $\hat{C}_h(v_l)$ will make it possible to search the most promising paths first, and the optimal solution can be found without examining all possible paths. Therefore, the search algorithm leveraging A-star does not generate the entire decision tree.

In addition to the *frontier list*, we also have a *generated set* which consists of all leaves which have been generated by the A-star algorithm. Each leaf v_l in the *generated set* is also assigned a pointer $PT(v_l)$ which equals the index of its parent leaf so that the A-star algorithm can backtrack from it to its parent leaf.

4.5 Contention-resolving MPC algorithm

Algorithms 2–4 present the pseudocode for our proposed algorithm based on the A-star algorithm to solve the optimization problem (7). Algorithm 2 presents the search algorithm. The optimal path search starts from the root v_0. The search algorithm keeps updating two sets, which are the *frontier list* and the *generated set*.

Algorithm 2 Main program.

Data: t_0, t_f, λ_i for $1 \le i \le N$, $\mathbf{x}(t_0)$, $\mathbf{u}(t_0)$, $Z(t_0)$
Result: $\mathbf{P}^*(t)$, $\mathbf{u}^*(t)$
Let *frontier list*=*generated set*= $\{v_0\}$;
$\hat{C}_f(v_0) = \hat{C}_h(v_0)$, $t = t_0$;
while $t_l^c \le t_f$ **do**
    v_l is the leaf in *frontier list* with minimal $\hat{C}_f$ cost;
    t_l^c is the contention time instant corresponding to v_l;
    Let $p = PT(v_l)$; ▷ v_p is the parent leaf of leaf v_l.
    if $t_l^c = t_f$ **then**
        return Reconstruct(v_l); Break;
    else
        j is the number of elements in *generated set*;
        for m-th permutation $\mathbf{P}_m \in \mathcal{P}(\{1, ..., M\})$ **do**
            $(v_{j+m}, t_{j+m}^c, w_{l,j+m})$=**Expand**$(v_l, \mathbf{P}_m, t_l^c)$;
            Add v_{j+m} into *frontier list* and *generated set*;
            $C_g(v_{j+m}) = C_g(v_l) + w_{l,j+m}$;
            $\hat{C}_h(v_{j+m}) = 0$;
            $\hat{C}_f(v_{j+m}) = C_g(v_{j+m}) + \hat{C}_h(v_{j+m})$;
            $PT(v_{j+m}) = l$;
        Remove v_l from *frontier list*;

Algorithm 3 backtracks the path from the selected terminal leaf to v_0 when case (1) is satisfied in the search algorithm. The backtracking starts from the terminal leaf v_l and utilizes the pointer $PT(v_l)$ to obtain the parent leaf v_p. The optimal priority assignment $\mathbf{P}^*(t)$ for the time interval between the contention time instants of v_p and v_l equals the priority assignment along the branch connecting v_p and v_l. Then we repeat this process with v_l and v_p replaced by v_p and the parent leaf of v_p, respectively. We repeat the backtracking process to obtain the optimal priority assignment $\mathbf{P}^*(t)$ until the contention time instant equals t_0. Algorithm 3 returns the optimal priority assignment $\mathbf{P}^*(t)$ for all $t \in [t_0, t_f]$ to the main program in Algorithm 2.

Algorithm 3 Reconstruct.

Data: v_l
Let $t = t_f$ and $p = PT(v_l)$;
while $t > t_0$ **do**
    Let $\mathbf{P}^*(t)$ be the priority assigned to the branch that connects v_p and v_l, from the contention time t_p^c of leaf v_p to the contention time t_l^c of v_l;
    Let $l = p$ and $p = PT(v_p)$;
    Let t be the corresponding contention time of v_l;
return $\mathbf{P}^*(t)$;

Algorithm 4 expands the selected leaf from the *frontier list* when case (2) is satisfied in the search algorithm. It utilizes Proposition 1 to determine the next contention time after a contention time t. Then it calls Algorithm 1 to compute the optimal control $u_i^*(t)$ and compute the branch cost $w_{l,j+m}$. Algorithm 4 returns the child leaf v_{j+m}, the next contention time t_{j+m}^c and the branch cost $w_{l,j+m}$ to the main program in Algorithm 2.

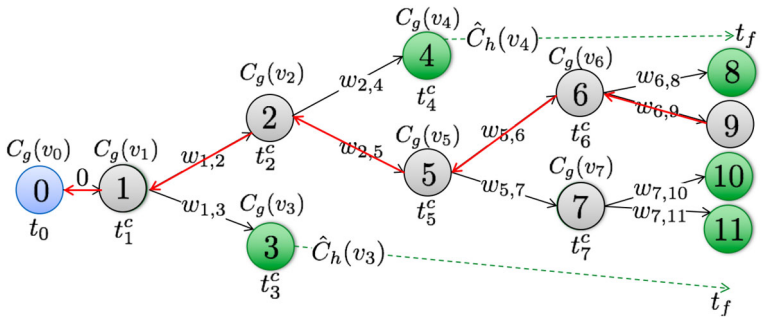


Fig. 5 Illustration of the subtree constructed by the proposed search algorithm. The blue circle represents the root v_0. Green circles represent leaves in the *frontier list*. Solid black arrows represent branches generated by the algorithm and dashed green arrows represent the estimate cost $\hat{C}_h(v_l)$. The red arrows represent the path with lowest cost

Algorithm 4 Expand.

1: **Data**: v_l, $\mathbf{P}_m$, t
2: Find the next contention time under priority $\mathbf{P}_m$ based on (21), and denote this contention time as t^c_{j+m};
3: Compute $u_i^*(t)$ and $w_{l,j+m}$ using Algorithm 1;
4: **return** v_{j+m}, t^c_{j+m}, $w_{l,j+m}$;

The searching algorithm does not generate the whole decision tree. Instead, it efficiently generates a subtree without losing optimality. At each iteration of the searching algorithm, it determines which leaf to expand further by selecting the leaf v_l with minimal $\hat{C}_f$ cost. Once the selected leaf is a terminal leaf, the lowest cost path is found. Figure 5 is an illustration of the subtree constructed by our algorithm described above, using the same example as Fig. 3. Compared with the entire decision tree in Fig. 3, some internal leaves in the subtree are open because our algorithm does not expand every leaf but intelligently expands a subset of leaves without losing optimality. Once the construction of the subtree reaches the terminal leaf, our algorithm backtracks the path along the red arrows. The total number of branches generated by the algorithm is 11, which reduces the computational workload for generating the entire tree.

4.6 Optimality of Contention-resolving MPC

In this subsection, we prove that our algorithm finds the optimal solutions $\mathbf{P}^*(t)$ and $\mathbf{u}^*(t)$ which minimize (7) given the CIA assumption.

Proposition 2 *The CIA assumption is a necessary condition for contention-resolving MPC algorithm to find the global optimal solution.*

Proof We prove the contrapositive of this proposition: if the CIA assumption is relaxed, then there are situations where a better solution exists compared to the solution computed by contention-resolving MPC.

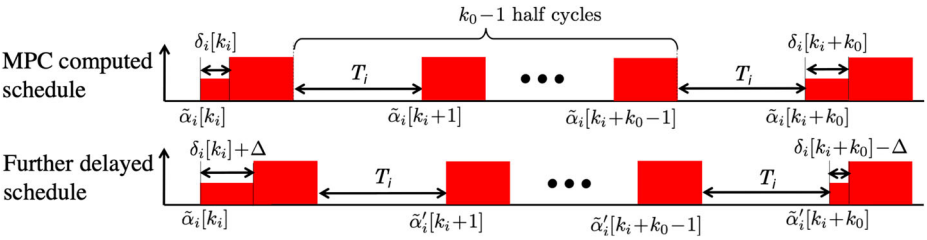


Fig. 6 Illustration of further delaying the arrival of a vehicle

Assume that a contention occurs among vehicle i and some other vehicles at time $\tilde{\alpha}_i[k_i]$. Then after the computation of contention-solving MPC, the priority assignment is determined and the time delay of vehicle i, $\delta_i[k_i]$, can be computed by Eq. 20 given the priorities computed by MPC. And we assume after k_0-1 half cycles, the second contention occurs to vehicle i at time $\tilde{\alpha}_i[k_i+k_0]$ and the time delay of vehicle i is $\delta_i[k_i+k_0]$. For the k_0-1 half cycles between the time interval $[\tilde{\alpha}_i[k_i]+\delta_i[k_i]+C_i, \tilde{\alpha}_i[k_i+k_0]]$, vehicle i can travel with its maximal speed. Figure 6 shows an illustration of the considered case. According to Eq. 26, the optimal speed for vehicle i within $[\gamma_i[k_i-1], \alpha_i[k_i])$ is $u_i^*(t) = \frac{\mathrm{D-L}}{T_i+\delta_i[k_i]}$ and within $[\gamma_i[k_i+k_0-1], \alpha_i[k_i+k_0])$, the optimal speed is $u_i^*(t) = \frac{\mathrm{D-L}}{T_i+\delta_i[k_i+k_0]}$. The cost under such schedule can be computed as

$$J_{\mathrm{MPC}} = \left(u_{i,\max} - \frac{\mathrm{D-L}}{T_i+\delta_i[k_i]}\right)^2 (T_i+\delta_i[k_i]) + \left(u_{i,\max} - \frac{\mathrm{D-L}}{T_i+\delta_i[k_i+k_0]}\right)^2 (T_i+\delta_i[k_i+k_0]).$$

Now we consider the case where the actual arrival time of vehicle i is further delayed by Δ at the first contention time, i.e., vehicle i arrives at the intersection at $\tilde{\alpha}_i[k_i]+\delta_i[k_i]+\Delta$. All the earliest arrival times after time $\tilde{\alpha}_i[k_i]+\delta_i[k_i]+\Delta$ are also delayed by Δ. And we assume the delay Δ does not affect the schedule of other vehicles and introduce more contentions. The cost under such schedule can be computed as

$$J(\Delta) = \left(u_{i,\max} - \frac{\mathrm{D-L}}{T_i+\delta_i[k_i]+\Delta}\right)^2 (T_i+\delta_i[k_i]+\Delta) + \left(u_{i,\max} - \frac{\mathrm{D-L}}{T_i+\delta_i[k_i+k_0]-\Delta}\right)^2 (T_i+\delta_i[k_i+k_0]-\Delta)$$

where $\Delta > 0$. Then, if we take the difference of these two costs, we have

$$J_{\mathrm{MPC}} - J(\Delta) = \frac{(\mathrm{D-L})^2(2T_i+\delta_i[k_i]+\delta_i[k_i+k_0])(\delta_i[k_i+k_0]-\delta_i[k_i]-\Delta)\Delta}{(T_i+\delta_i[k_i])(T_i+\delta_i[k_i+k_0])(T_i+\delta_i[k_i]+\delta_i[k_i+k_0])T_i} \tag{31}$$

From Eq. 31 we can see, if $0 < \Delta < \delta_i[k_i+k_0]$ and $\delta_i[k_i+k_0]-\delta_i[k_i]-\Delta > 0$, we have $J_{\mathrm{MPC}} > J(\Delta)$. In other words, if the time delay $\delta_i[k_i+k_0]$ is greater than $\delta_i[k_i]$, then we can always find a Δ satisfying $0 < \Delta < \delta_i[k_i+k_0]-\delta_i[k_i]$ and the cost $J(\Delta)$ will be less than J_{MPC}. In the case where $\delta_i[k_i+k_0] > \delta_i[k_i]$, contention-resolving MPC can only find a sub-optimal solution if Assumption 3 is relaxed. □

A numerical exmaple where the conditions $0 < \Delta < \delta_i[k_i+k_0]$ and $\delta_i[k_i+k_0]-\delta_i[k_i]-\Delta > 0$ are satisfied will be presented in Section 5.3 to further justify the proof above.

Theorem 2 *Based on Assumptions 1, 2 and 3, the Algorithm 2 finds the optimal solution* $\mathbf{P}^*(t)$ *and* $\mathbf{u}^*(t)$ *for the optimization problem formulated by Eq. 7.*

Proof Since all branch cost is non-negative, $\hat{C}_h(v_n) \leq C_h(v_n)$ for all v_n is satisfied. From Hart et al. (1968), if $\hat{C}_h(v_l) \leq C_h(v_l)$ for all v_l, the A-star algorithm finds the minimal total cost from v_0 to a terminal leaf. □

Remark 3 After contention-resolving MPC algorithm computes the optimal solution $\mathbf{P}^*(t)$ and $\mathbf{u}^*(t)$ for all $t \in [t_0, t_f]$, only the $\mathbf{P}^*(t)$ and $\mathbf{u}^*(t)$ for the first sub-interval $t \in [t_0, t_1]$, where t_1 is the significant moment computed by Eq. 11. Then at time instant t_1, the contention resolving MPC will be triggered again. The decision tree will be reconstructed with t_1 replaced by t_0 and t_f replaced by $t_f + t_1 - t_0$. Then the $\mathbf{P}^*(t)$ and $\mathbf{u}^*(t)$ will be applied for the sub-interval $t \in [t_1, t_2]$. And the process repeats.

Remark 4 Since the set $\mathcal{P}(\{1, ..., M\})$ contains all possible priority assignments, it also includes the priorities following the FCFS and the HSF rules. Therefore, the priorities assigned by the FCFS and HSF strategies are represented by paths in the decision tree, but not necessarily the path with the minimal cost. Therefore, our method can guarantee to find a better or same solution as FCFS and HSF strategies.

Remark 5 And since all the scheduling strategies and the optimal solution are based on Assumption 3, we will also provide a insight discussion about the cases where Assumption 3 is relaxed in Section 5.3.

5 Case studies

This section presents the simulation results obtained by the proposed method implemented in Matlab. We compare our proposed method with first come first serve scheduling (or FCFS) strategy and highest speed first (or HSF) strategy and demonstrate that our optimal scheduling method can provide a better solution than FCFS and HSF.

In the simulation, we consider 5 vehicles traveling on the figure eight track. The vehicle length L is 15 feet. We choose S and D such that $S + L = 0.75$ miles, $D - L = 6$ miles. Let $t_f = 25$ minutes and the speed limit of the first vehicle be $u_{1,\max} = 1.25$ miles per minute (75 mph). Let the speed limit of the other four vehicles be the same, $u_{2,\max} = u_{3,\max} = u_{4,\max} = u_{5,\max} = 1$ mile per minute (60 mph). The intersection speed $u_{i,\text{int}} = 0.75$ mile per minute (45 mph) for all i.

5.1 Contention-resolving MPC VS FCFS

For the first studied case, we set the initial positions to be $x_1^0 = 4.25$, $x_2^0 = 4.85$, $x_3^0 = 3.25$, $x_4^0 = 0.75$ and $x_5^0 = 0$ miles. In this setup, vehicle 5 arrives at the intersection at time 0. The earliest arrival times of vehicles 1,2,3 and 4 are 2, 1.9, 3.5 and 6 minutes, respectively. The algorithm only takes 0.12 seconds to find the solution for this example. The total cost is 0.3758.

The intersection occupation result is shown in Fig. 7. The vehicle speed design is shown in Fig. 8. Four contentions occur in the time interval [0, 25]. The first contention occurs at

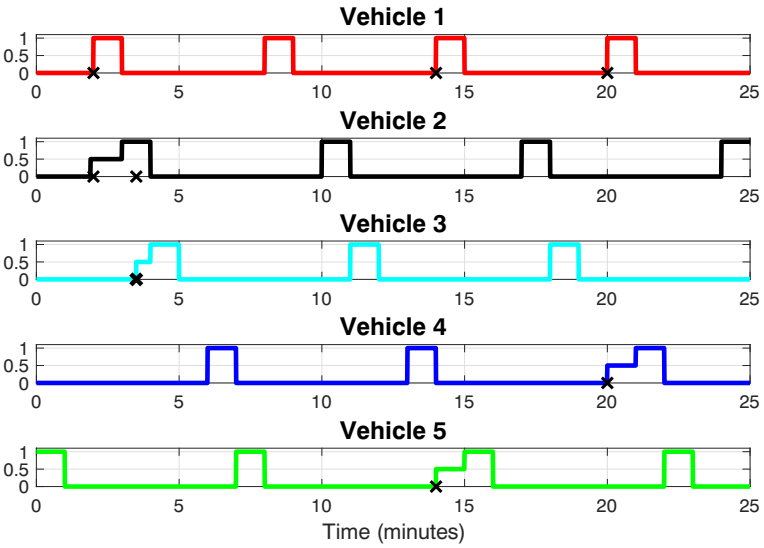


Fig. 7 Intersection occupation for scheduling five vehicles. The y axis value 1 means that the vehicle is occupying the intersection, 0 means that the vehicle has not arrived at the intersection, and 0.5 means that the earliest arrival of a vehicle is delayed by a contention. The black crosses mark the time instant when a contention occurs

time 2. Although the earliest arrival time of vehicle 2 is smaller than vehicle 1, the priority assignment computed by our method gives vehicle 1 higher priority (which is different from the priority assigned by the FCFS strategy) because it has a higher speed limit than vehicle 2. At time 3.5, the second contention occurs between vehicles 2 and 3, which creates the possibility that vehicle 2 can be delayed twice. Our method can solve this problem and determines that the optimal priority assignment is that vehicle 2 crosses the intersection

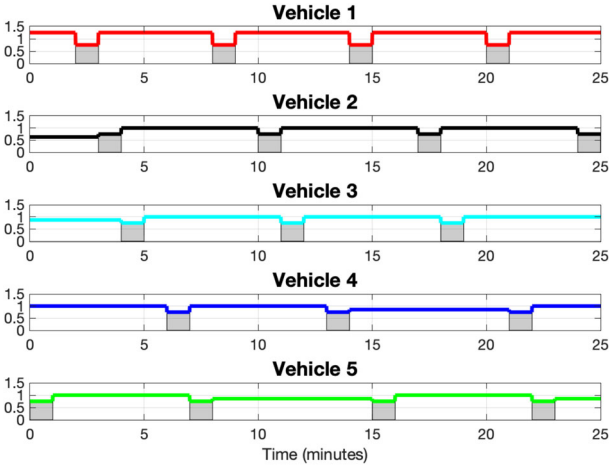


Fig. 8 Optimal vehicle speed of scheduling five vehicles. The shaded areas mark the time interval when a vehicle is crossing the intersection

before vehicle 3. Two more contentions occur at 14 and 20 minutes, and vehicle 1 is assigned a higher priority to resolve these two contentions.

For comparison, if we always assign higher priority to the vehicle which arrives at the intersection first, i.e. following the FCFS strategy, and use regular MPC to design the vehicle speeds, then the cost would be 0.5114, which is 36% higher than our solution. While the example is simple, the simulation results show that our method performs better than the FCFS. Notice that in this specific simulation case, vehicle 1 always has highest priority, which agrees with the HSF scheduling strategy because vehicle 1 has the highest traveling speed. However, HSF is not always the optimal solution, which will be shown in the next subsection.

5.2 Contention-resolving MPC VS HSF

In the previous simulated case, the optimal priority assignment is the same as the priority assignment under HFS. However, if we change the initial condition, the HFS will not be optimal and we will show that our optimal priority assignment can perform significantly better than HSF strategy.

Let the initial condition to be $x_1^0 = 1$, $x_2^0 = 3$, $x_3^0 = 4$, $x_4^0 = 5$ and $x_5^0 = 0$ miles. In this setup, vehicle 5 arrives at the intersection at time 0. The earliest arrival times of vehicles 1,2,3 and 4 are 4, 3, 2 and 1 minutes, respectively. The contention-resolving MPC algorithm only takes 0.09 seconds to find the solution for this example. The total cost is 0.4662.

The decision tree constructed by contention-resolving MPC is shown in Fig. 9. Six contentions occur in the time interval [0, 25]. Therefore, the total number of leaves in the fully constructed decision tree is $2^6 = 64$. And we can see that using the A* inspired searching

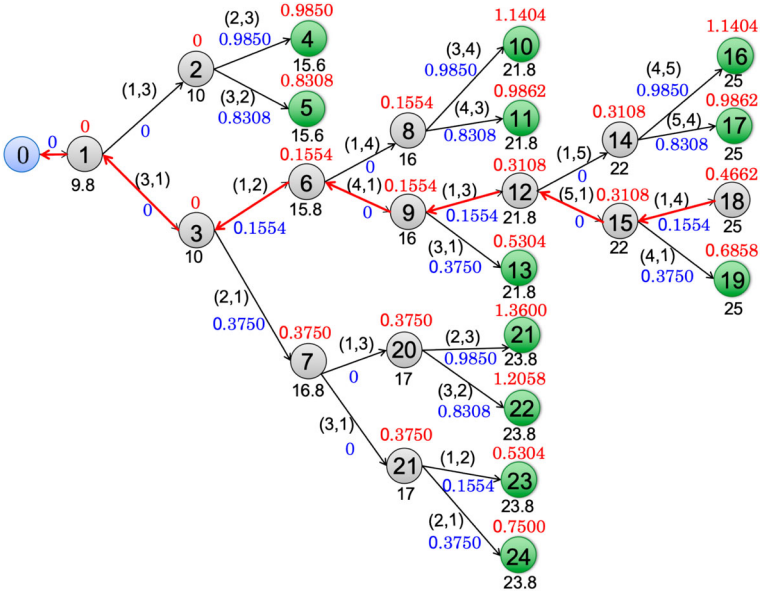


Fig. 9 Decision tree constructed by contention-resolving MPC. Blue numbers represent branch costs $w_{l,j}$. The black numbers under leaves represent contention time instant t_l^c. The red numbers above leaves represent estimated total costs $\hat{C}_f(v)$. The red arrows represent the path with lowest cost

algorithm, contention-resolving MPC only needs to generate 24 leaves to find the optimal solution.

The intersection occupation result where the vehicles are scheduled under the optimal priority assignment is shown in Fig. 10 and the vehicle speed design is shown in Fig. 11. The first contention occurs between vehicles 1 and 3 at time 9.8, marked by the black cross. The second contention occurs between vehicles 1 and 2 right after the first contention at time 10, marked by the purple cross. As we can see from Fig. 10, although the maximal speed limit of vehicle 1 is larger than vehicle 3, the priority assignment computed by our method assigns vehicle 3 with higher priority (represented by the branch between leaf 1 and 3 in Fig. 9). The intuitive reason for such solution is that vehicle 3 only needs an extra 0.2 minutes to pass the intersection when the first contention occurs, while vehicle 1 needs 1 minute to pass the intersection. Therefore, assigning vehicle 3 with higher priority leads to a smaller cost. Under such priority assignment, the optimal speed $u_1^*(t)$ of vehicle 1 is reduced to 1.2 miles per minute, which is shown within the time interval [5, 10] in Fig. 11. Vehicle 1 only needs to slightly sacrifice its maximal speed to resolve this contention while vehicle 3 travels with its maximal speed. Then at the next contention time 10, the second contention occurs between vehicles 1 and 2, which creates the possibility that vehicle 1 can be delayed twice. Our method can resolve this issue and determine that the optimal priority assignment is that vehicle 1 has higher priority than vehicle 2. Seen from Fig. 11, under this priority assignment, the optimal speed of vehicle 2 is $\frac{6}{7}$ miles per minute within time interval [4, 11] minutes. Similar situations occur twice at 15.8 and 21.8 minutes, where vehicle 1 is assigned with a lower priority than the first contended vehicle and a higher priority than the second contended vehicle.

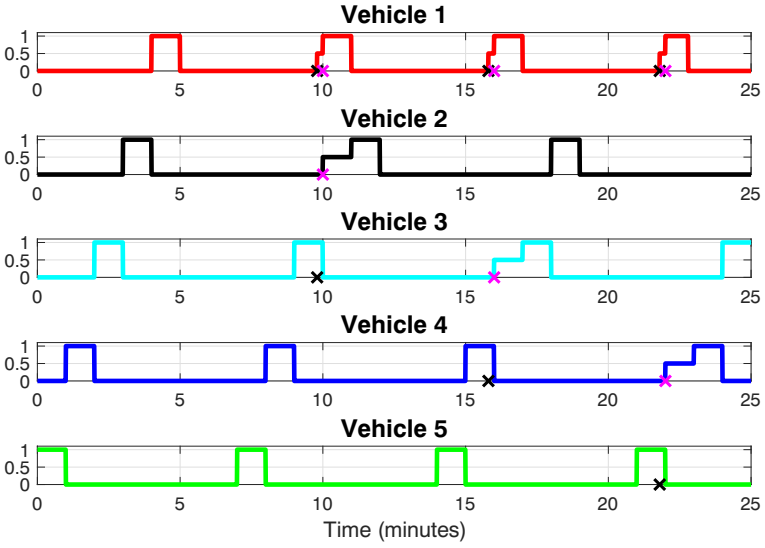


Fig. 10 Intersection occupation for scheduling five vehicles. The y axis value 1 means that the vehicle is occupying the intersection, 0 means that the vehicle has not arrived at the intersection, and 0.5 means that the earliest arrival of a vehicle is delayed by a contention. The black crosses mark the time instant when a contention occurs

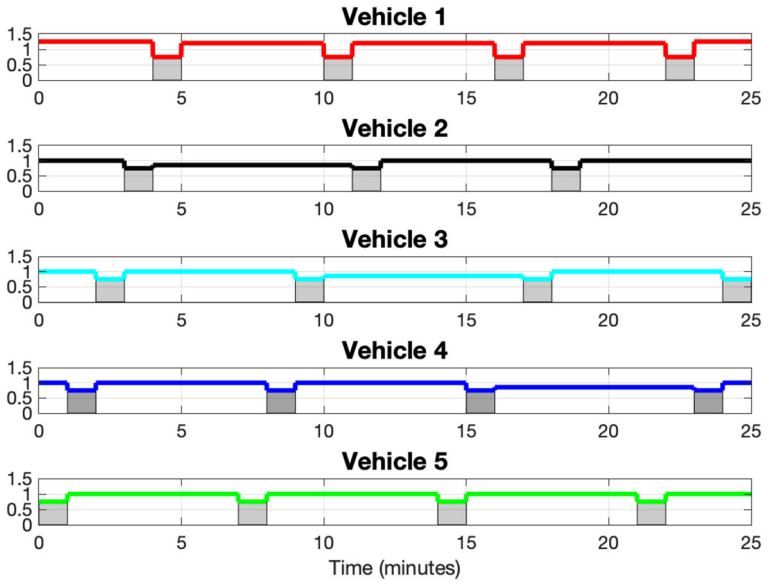


Fig. 11 Optimal vehicle speed of scheduling five vehicles. The shaded areas mark the time interval when a vehicle is crossing the intersection

For comparison, if we always assign higher priority to the vehicle with the highest speed limit, i.e. vehicle 1 always has highest priority following the HSF strategy, and use regular MPC to design the vehicle speeds, then the cost would be 1.4235, which is 205% higher than our solution.

5.3 Numerical results without the CIA assumption

In this subsection, we will discuss cases where the CIA assumption is relaxed and show a numerical example where the contention-resolving MPC can only find a sub-optimal solution.

Consider 2 vehicles traveling on the figure eight track. Let the speed limit of the first vehicle be $u_{1,\max} = 1.5$ miles per minute (90 mph). Let the speed limit of the second vehicle be $u_{2,\max} = 1$ mile per minute (60 mph). The intersection speed is $u_{i,\text{int}} = 0.75$ mile per minute (45 mph) for all i. The initial positions are $x_1^0 = 4.8$ and $x_2^0 = 0$ miles. All the other parameters are the same as previous simulations.

Figure 12 shows the intersection occupation results of vehicle 1 and vehicle 2 with and without CIA assumption. With the CIA assumption, we can see two contentions occur between vehicle 1 and vehicle 2 within the time horizon [0, 25]. The first contention occurs at 7 minutes. To resolve this contention, vehicle 1 is assigned with higher priority and it leaves the intersection at 7.2 minutes. Therefore, vehicle 2 can only enter the intersection at or after time 7.2 minutes. Under Assumption 3, the speed of vehicle 2 is decreased to be $\frac{6}{6.2}$ miles per minute for the time window [1, 7.2] such that vehicle 2 arrives at and enter the intersection at time 7.2 minutes. And it leaves the intersection at 8.2 minutes. Then vehicle 2 travels with its maximal speed and arrives at the intersection at time 14.2 without any contention with vehicle 1. The second contention between vehicles 1 and 2 occurs at time 21.2. Vehicle 1 is also assigned with higher priority than vehicle 2 to resolve this contention.

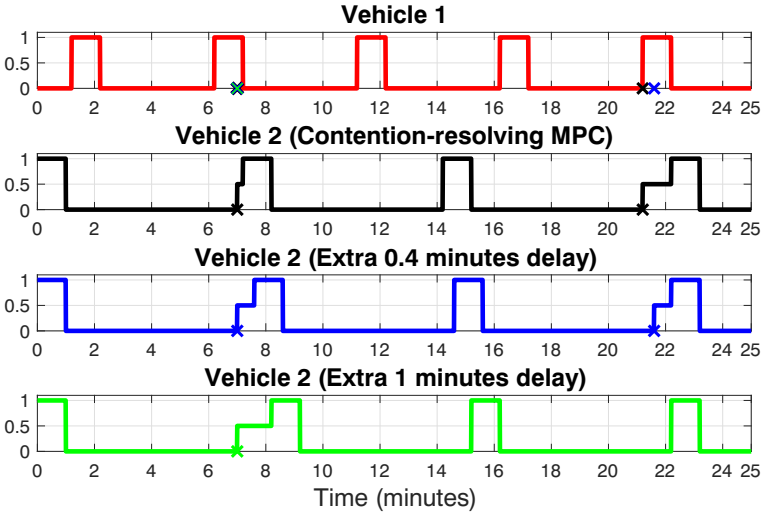


Fig. 12 Intersection occupation for scheduling two vehicles. The First two sub-figures show the intersection occupation results of vehicle 1 and vehicle 2 computed by contention-resolving MPC under Assumption 3. The third sub-figure (from top to bottom) shows the intersection occupation time of vehicle 2 with an extra 0.4 minutes time delay. The forth sub-figure shows the intersection occupation time of vehicle 2 with an extra 1 minutes time delay

Therefore, vehicle 1 enters the intersection at at time 21.2 and leaves the intersection at time 22.2. Vehicle 2 travels with a reduced speed $\frac{6}{7}$ miles per minute and arrives at the intersection at time 22.2 and leaves the intersection at time 23.2. The corresponding speed design for vehicle 2 with CIA is shown by the second sub-figure in Fig. 13. The total cost under such scheduling and control co-design solution is $J_1 = \left(1 - \frac{6}{6.2}\right)^2 \cdot 6.2 + \left(1 - \frac{6}{7}\right)^2 \cdot 7 = 0.1493$.

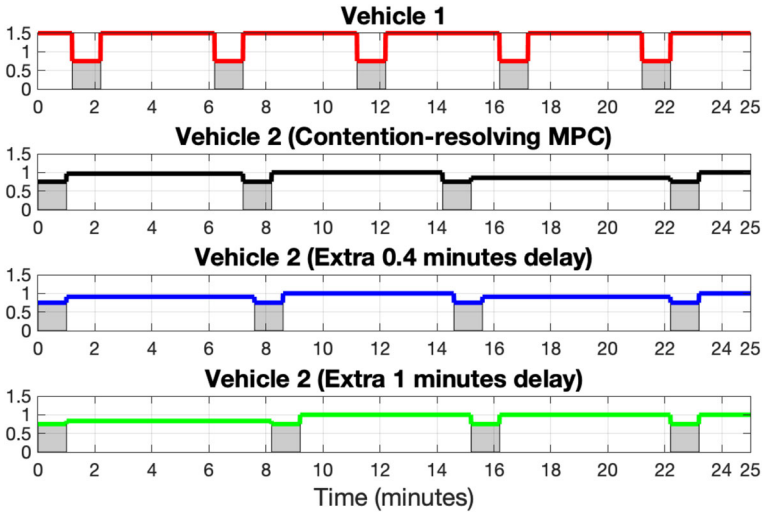


Fig. 13 vehicle speed design of scheduling two vehicles

Case 1: if we relax the CIA assumption and let vehicle 2 be delayed for 0.6 minutes at the first contention (as shown by the third sub-figure in Fig. 12), then vehicle 2 travels with a reduced speed $\frac{6}{6.6}$ miles per minute and arrives at the intersection at time 7.6, which is 0.4 minute after the time instant when vehicle 1 leaves the intersection. Since vehicle 2 is delayed for an extra 0.4 minutes compared to the case with the CIA assumption, the next earliest possible arrival time of vehicle 2 after time 7.6 is 14.6, which is also delayed for an extra 0.4 minutes compared to the case under the CIA assumption, shown by the sub-figure in the middle of Fig. 12. At the earliest arrival time 14.6, vehicle 2 does not contend with vehicle 2 so it can travel with its maximal speed within time [8.6, 14.6] and enter the intersection at time 14.6. The next earliest arrival time of vehicle 2 after time 14.6 is 22.6 where second contention occurs. Vehicle 1 is also assigned with higher priority than vehicle 2 to resolve this contention. Therefore, vehicle 1 enters the intersection at at time 21.2 and leaves the intersection at time 22.2. Vehicle 2 travels with a reduced speed $\frac{6}{6.6}$ miles per minute and arrives at the intersection at time 22.2 and leaves the intersection at time 23.2. The corresponding vehicle speed design is shown by the third sub-figure in Fig. 13. The total cost under this schedule and vehicle speed control is $J_2 = \left(1 - \frac{6}{6.6}\right)^2 \cdot 6.6 + \left(1 - \frac{6}{6.6}\right)^2 \cdot 6.6 = 0.1091$, which is less than J_1. This numerical result shows that if the CIA assumption is relaxed, contention-resolving MPC cannot find the global optimal solution. A solution which leads to smaller cost than the solution computed by contention-resolving MPC may exist.

Case 2: if we relax the CIA assumption and further delay the arrival of vehicle 2 at the first contention, as shown by the bottom sub-figure in Fig. 12, then vehicle 2 travels with a reduced speed $\frac{6}{7.2}$ miles per minute and arrives at the intersection at time 8.2, which is 1 minute after the time instant when vehicle 1 leaves the intersection. Then the next two earliest possible arrival times of vehicle 2 after time 7.6 are 15.2 and 23.2 minutes. At both arrival times, vehicle 2 does not contend with vehicle 1 which reduces the number of contentions. It can travel with its maximal speed within time [9.2, 25]. The corresponding vehicle speed design is shown by the bottom sub-figure in Fig. 13. The total cost under this schedule is $J_3 = \left(1 - \frac{6}{7.2}\right)^2 \cdot 7.2 = 0.2$, which is greater than J_1 and J_2. This example shows that further delaying the arrival of a vehicle not only affect the cost of the co-design optimization solution, it can also change the number of contentions in the future, which changes the structure of the decision tree. From Eq. 31, if $\Delta \geq \delta_i[k_i + k_0]$, i.e., the extra time delay is large enough such that the second contention will not occur, then we have $\delta_i[k_i + k_0] - \delta_i[k_i] - \Delta < 0$, which leads to $J_{\text{MPC}} < J(\Delta)$. Therefore, further delaying the arrival of a vehicle to avoid the second contention cannot reduce the cost computed by contention-resolving MPC.

6 Conclusions and future work

Coordinating automated vehicles at traffic intersections is a challenging problem that is of compelling ongoing research interest, owing to the limitations of traditional traffic light, FCFS and HSF scheduling approaches. We presented a novel method to co-design

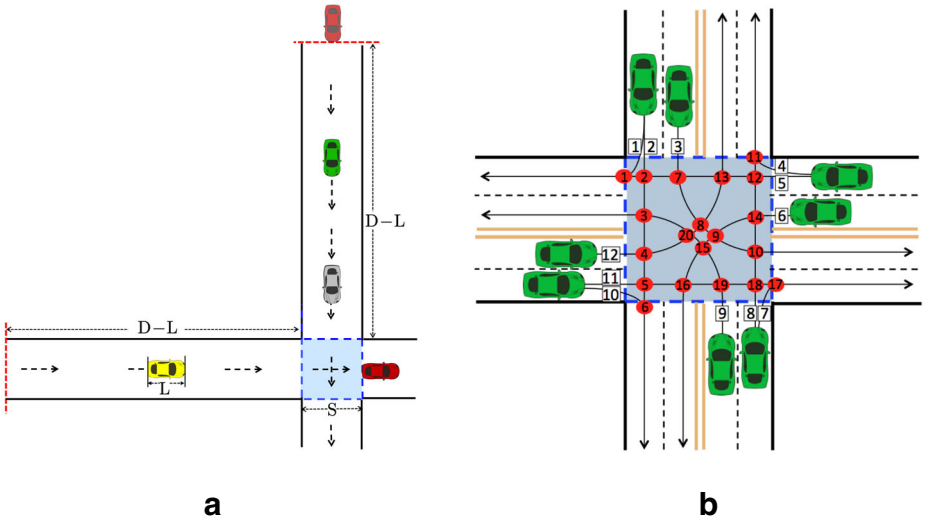


Fig. 14 Examples of realistic intersections setup that "figure eight" shaped road is related to. **a** The equivalent two-lane intersection as the "figure eight" set up. **b** A complicated four-lane intersection

priority assignments and control the travel speeds of vehicles at an idealized intelligent intersection. Our simulation results demonstrated promising improvements using the contention-resolving MPC method, compared with the FCFS and HSF scheduling methods.

The idealized traffic model is quite similar to a two-lane intersection as Fig. 14a, where arrival times of a new vehicle (arriving at the red dashed line) are predictable. For future work, we may change the distribution of the vehicle arrival time to obey Poisson distributions. This "two-lane" intersection is the basic problem because it is an element of a complicated intersection where a vehicle can turn right, go straight, or turn left, as shown in Fig. 14b. Each red node is one contented area that can be viewed as the intersection area in the idealized model. For future work, we will develop contention-resolving MPC for traffic control and scheduling at such multi-lane intersections.

Acknowledgements This work was partially supported by ONR grants N00014-19-1-2556, N00014-19-1-2266 and N00014-16-1-2667; NSF grants OCE-1559475, CNS-1828678, and S&AS-1849228; NRL grants N00173-17-1-G001 and N00173-19-P-1412; and NOAA grant NA16NOS0120028.

References

Ahn H, Colombo A, Del Vecchio D (2014) Supervisory control for intersection collision avoidance in the presence of uncontrolled vehicles. In: 2014 American control conference. IEEE, pp 867–873

Conway RW, Maxwell WL, Miller LW (2003) Theory of scheduling. Courier Corporation

Dallal E, Colombo A, Del Vecchio D, Lafortune S (2017) Supervisory control for collision avoidance in vehicular networks using discrete event abstractions. Discret Event Dyn Syst 27(1):1–44

Dimitrakopoulos G, Demestichas P (2010) Intelligent transportation systems. IEEE Veh Technol Mag 5(1):77–84. https://doi.org/10.1109/MVT.2009.935537

Fayazi SA, Vahidi A (2018) Mixed-integer linear programming for optimal scheduling of autonomous vehicle intersection crossing. IEEE Trans Intell Veh 3(3):287–299

Frejo J, Camacho E (2012) Global versus local MPC algorithms in freeway traffic control with ramp metering and variable speed limits. IEEE Trans Intell Transp Syst 13(4):1556–1565. https://doi.org/10.1109/TITS.2012.2195493

Fukushima H, Kon K, Matsuno F (2013) Model predictive formation control using branch-and-bound compatible with collision avoidance problems. IEEE Trans Robot 29(5):1308–1317

Hart P, Nilsson N, Raphael B (1968) A formal basis for the heuristic determination of minimum cost paths. IEEE Trans Syst Sci Cybern 4(2):100–107. https://doi.org/10.1109/TSSC.1968.300136

Hult R, Zanon M, Gras S, Falcone P (2018) An miqp-based heuristic for optimal coordination of vehicles at intersections. In: 2018 IEEE Conference on decision and control (CDC). IEEE, pp 2783–2790

Jiao P, Wang H, Sun T (2014) Real-time arterial coordination control based on dynamic intersection turning fractions estimation using genetic algorithm. Math Probl Eng, 2014

Kim KD, Kumar PR (2014) An mpc-based approach to provable system-wide safety and liveness of autonomous ground traffic. IEEE Trans Autom Control 59(12):3341–3356

Lee J, Park B (2012) Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. IEEE Trans Intell Transp Syst 13(1):81–90. https://doi.org/10.1109/TITS.2011.2178836

Lin S, De Schutter B, Xi Y, Hellendoorn H (2011) Fast model predictive control for urban road networks via milp. IEEE Trans Intell Transp Syst 12(3):846–856. https://doi.org/10.1109/TITS.2011.2114652

Lu Q, Kim KD (2017) A genetic algorithm approach for expedited crossing of emergency vehicles in connected and autonomous intersection traffic. J Adv Transp, 2017

Malikopoulos AA, Cassandras CG, Zhang YJ (2018) A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections. Automatica 93:244–256

Meng Y, Li L, Wang FY, Li K, Li Z (2018) Analysis of cooperative driving strategies for nonsignalized intersections. IEEE Trans Veh Technol 67(4):2900–2911

Qian X, Gregoire J, De La Fortelle A, Moutarde F (2015) Decentralized model predictive control for smooth coordination of automated vehicles at intersection. In: 2015 European control conference (ECC). IEEE, pp 3452–3458

Rios-Torres J, Malikopoulos AA (2017) A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps. IEEE Trans Intell Transp Syst 18(5):1066–1077

Sethi SP, Thompson GL (2000) Optimal control theory applications to economics and management science, 2nd edn. Kluwer Academic Publishers, Norwell

Shi Z, Zhang F (2015) Model predictive control under timing constraints induced by controller area networks. Real-Time Syst, 1–32

Shi Z, Yao N, Zhang F (2017) Scheduling feasibility of energy management in micro-grids based on significant moment analysis. In: Song H, Rawat DB, Jeschke S, Brecher C (eds) Cyber-physical systems: foundations, principles and applications. Elsevier, pp 431–450

Tachet R, Santi P, Sobolevsky S, Reyes-Castro L, Frazzoli E, Helbing D, Ratti C (2016) Revisiting street intersections using slot-based systems. PloS One 11(3):e0149607. https://doi.org/10.1371/journal.pone.0149607

Yan F, Dridi M, El Moudni A (2013) An autonomous vehicle sequencing problem at intersections: a genetic algorithm approach. Int J Appl Math Comput Sci 23(1):183–200. https://doi.org/10.2478/amcs-2013-0015

Yao N, Zhang F (2018) Resolving contentions for intelligent traffic intersections using optimal priority assignment and model predictive control. In: 2018 IEEE Conference on control technology and applications (CCTA). IEEE, pp 632–637

Yao N, Malisoff M, Zhang F (2017) Contention resolving optimal priority assignment for event-triggered model predictive controllers. In: American control conference (ACC), 2017. IEEE, pp 2357–2362

Yao N, Malisoff M, Zhang F (2019) Contention-resolving model predictive control for coordinating automated vehicles at a traffic intersection. In: 2019 IEEE Conference on decision and control (CDC). IEEE, p Accepted

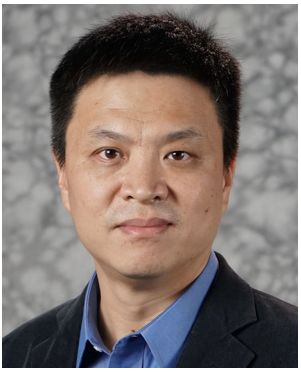
Yao N, Malisoff M, Zhang F (2020) Contention-resolving model predictive control for coupled control systems with a shared resource. Automatica 122:109219

Zhang K, Zhang D, de La Fortelle A, Wu X, Gregoire J (2015) State-driven priority scheduling mechanisms for driverless vehicles approaching intersections. IEEE Trans Intell Transp Syst 16(5):2487–2500

Zhang K, Yang A, Su H, de La Fortelle A, Miao K, Yao Y (2017) Service-oriented cooperation models and mechanisms for heterogeneous driverless vehicles at continuous static critical sections. IEEE Trans Intell Transp Syst 18(7):1867–1881

Zhang YJ, Malikopoulos AA, Cassandras CG (2016) Optimal control and coordination of connected and automated vehicles at urban traffic intersections. In: American control conference (ACC), 2016. IEEE, pp 6227–6232

Ningshi Yao received her Ph.D. degree from the School of Electrical and Computer Engineering at Georgia Institute of Technology, Atlanta, USA, in 2020. She received the B.S. degree from Zhejiang University, Hangzhou, China, in 2010. Her research interests include scheduling and control co-design, cyber physical systems, and human robot interaction.



Fumin Zhang is a professor in the School of Electrical and Computer Engineering at the Georgia Institute of Technology. He received a PhD degree in 2004 from the University of Maryland (College Park) in Electrical Engineering, and held a postdoctoral position in Princeton University from 2004 to 2007. His research interests include mobile sensor networks, maritime robotics, control systems, and theoretical foundations for cyber-physical systems. He received the NSF CAREER Award in 2009 and the ONR Young Investigator Program Award in 2010.