A Dynamic Programming Approach to Optimal Lane Merging of Connected and Autonomous Vehicles

Shang-Chien Lin[1], Hsiang Hsu[1], Yi-Ting Lin[1], Chung-Wei Lin[1], Iris Hui-Ru Jiang[1], Changliu Liu[2]
[1]National Taiwan University, [2]Carnegie Mellon University

***Abstract*— Lane merging is one of the major sources causing traffic congestion and delay. With the help of vehicle-to-vehicle or vehicle-to-infrastructure communication and autonomous driving technology, there are opportunities to alleviate congestion and delay resulting from lane merging. In this paper, we first summarize modern features and requirements for lane merging, along with the advance of vehicular technology. We then formulate and propose a dynamic programming algorithm to find the optimal solution for a two-lane merging scenario. It schedules the passing order for vehicles while minimizing the time needed for all vehicles to go through the merging point (equivalent to the time that the last vehicle goes through the merging point). We further extend the problem to a consecutive lane-merging scenario. We show the difficulty to apply the original dynamic programming to the consecutive lane-merging scenario and propose an improved version to solve it. Experimental results show that our dynamic programming algorithm can efficiently minimize the time needed for all vehicles to go through the merging point and reduce the average delay of all vehicles, compared with some greedy methods.**

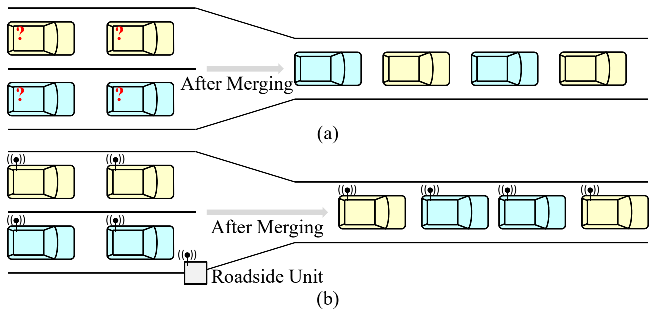

Fig. 1. (a) When vehicles are uncertain about other vehicles' behavior, they need to slow down or even stop for safety and result in potentially avoidable congestion and delay. (b) The communication provides vehicles' information to lower the uncertainty between vehicles, and, after vehicles reach the consensus of a passing order, the autonomous driving technology provides precise control to comply with the passing order and reduce vehicle-following gaps efficiently.

I. Introduction

Lane merging is one of the major sources causing traffic congestion and delay. This is not only because of the reduction of road capacity (number of lanes) but also because of the uncertainty of vehicle behavior. When vehicles are uncertain about other vehicles' behavior, they need to slow down or even stop for safety and result in potentially avoidable congestion and delay, as shown in Figure 1(a). With the help of vehicle-to-vehicle or vehicle-to-infrastructure communication and autonomous driving technology, there are opportunities to alleviate congestion and delay resulting from lane merging, as shown in Figure 1(b). The communication provides vehicles' information (*e.g.*, via the Basic Safety Messages of Dedicated Short-Range Communications [1]) to lower the uncertainty between vehicles, and, after vehicles reach the consensus of a passing order, the autonomous driving technology provides precise control to comply with the passing order and reduce vehicle-following gaps efficiently. To complete the whole lane-merging operation for connected and autonomous vehicles, in this paper, we are targeting how to decide the passing order for vehicles.

Lane merging has been a traditional problem in the domain of traffic management. Uno *et al.* utilized vehicle-to-vehicle communication to facilitate the lane-merging operation in an early work [2]. They considered the problem as a series of sub-problems, and only one vehicle is merging in each sub-problem. Based on vehicle-to-vehicle communication, they created a virtual vehicle to allocate sufficient and safe space, but they just assumed that the passing order is pre-defined.

Utilizing vehicle-to-vehicle or vehicle-to-infrastructure communication, many following works develop controllers for merging from secondary lanes to main lanes. Rios-Torres and Malikopoulos provided detailed survey and summary [3]. For example, Milanes *et al.* calculated the time that a merging vehicle should enter the main lane and designed a fuzzy controller to manage the actuation [4]. Wang *et al.* also created virtual vehicles and calculated the desired speeds of vehicles [5]. They discussed a scenario that two vehicles form a platoon and merge into the main lane together (not just first-arrive-first-go), but only four vehicles (two on each lane) are considered in the scenario. Ntousakis *et al.* gave an analytic solution to minimize the required acceleration of each merging vehicle [6]. Aoki *et al.* presented a merging protocol to ensure safety and high throughput in [7]. Their protocol was designed both for traffic with only autonomous vehicles and for traffic with a mixture of autonomous vehicles as well as human-driven vehicles by using vehicle-to-vehicle communication and the perception systems. However, these previous approaches about merging optimization do not consider the variant vehicle-following gaps, which should depend on whether two consecutive vehicles are from the same lane or not before merging.

Regarding passing-order optimization, Raravi *et al.* proposed a greedy algorithm which iteratively evaluates the first vehicles on both lanes and selects the one with a lower cost to pass first [8]. Awal *et al.* targeted to find an optimal passing order and pointed out that checking all possible passing orders is not necessary so that they proposed an algorithm to prune some passing orders which are infeasible or not prospective [9]. Regarding dynamic programming in related problems of transportation systems, Wu *et al.* used dynamic programming for intersection management [10], and van de Hoef *et al.* used dynamic programming to maximize the probability that two vehicles meet at an intersection so that they can form a platoon [11].

With the advance of connected and autonomous technology, vehicles can collect more information and react more precisely, and thus the passing-order optimization can further consider the interactions between the first vehicles and other vehicles (not only the first vehicles on both lanes [8]) to improve the solution quality. Also, lane merging forms a good structure for dynamic programming so that the passing order can be solved optimally in an efficient way. With these observations, we summarize some modern features and requirements for lane merging as follows:

- **Variant Vehicle-Following Gaps**. Connected and autonomous vehicles on the same lane can utilize cooperative adaptive cruise control (CACC) to form a platoon. *Although the platooning can reduce the vehicle-following gaps and thus improve the traffic throughput on a lane, it makes lane-merging optimization non-trivial.* This is because consecutive vehicles from the same lane can keep the vehicle-following gaps maintained by CACC right after merging (*e.g.*, two vehicles from the bottom lane in Figure 1(b)), while interleaving vehicles from different lanes will increase the vehicle-following gaps (without CACC) for all vehicles (except the first one) right after merging (*e.g.*, Figure 1(a))—even if they utilize CACC to form a new platoon afterward, congestion and delay at the merging point may have occurred, and it introduces higher platoon-establishing overhead.
- **Global View within Communication Range**. Considering the variant vehicle-following gaps mentioned above, the passing-order decision is no longer a problem only for the first vehicles on different lanes. It should have a global view of vehicles within the communication range so that the optimization can be more effective.
- **Real-Time Decision**. The passing order of vehicles should be decided in real time. Considering all vehicles within the communication range may increase the computational overhead of the passing-order decision and thus decrease its performance and scalability. Furthermore, the computational resource of a roadside unit or a vehicle is usually limited. As a result, an algorithm deciding the passing order should be very efficient so that a single roadside unit or a single vehicle can execute it alone.
- **Fundamental of Conflict Resolution**. Lane merging is a fundamental type of conflict resolution—vehicles intend to pass (occupy) a region, but only one vehicle can pass (occupy) it at the same time. The algorithm deciding the passing order can provide insights to more complicated scenarios which need to resolve conflicts of vehicles.

To address the features and requirements, the main contributions of this paper are as follows:

- We formulate a two-lane merging (passing-order decision) problem, and the formulation considers the variant vehicle-following gaps and has a global view for vehicles within the communication range (or a certain range).
- We propose a dynamic programming algorithm to find the optimal passing order for the two-lane merging problem and minimize the time needed for all vehicles to pass the merging point. The algorithm is extremely efficient so that it can be applied in real time.
- We further extend the problem to a consecutive lane-merging scenario, which is fundamental to further generalization, such as a roundabout and neighboring ramps.

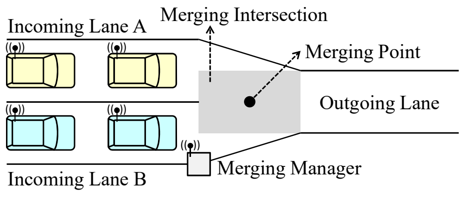

Fig. 2. Two-lane merging scenario.

The rest of this paper is organized as follows: Section II presents the terminology and the problem formulation. Section III describes our dynamic programming algorithm. Section IV generalizes the problem and algorithm to consecutive lane-merging scenarios. Section V provides experimental results, and Section VI concludes this paper.

II. Terminology and Problem Formulation

A. Definitions and Assumptions

Lane Merging. We assume that all vehicles are moving along the same direction. Lane merging is the process that vehicles from different incoming lanes merge into one outgoing lane. For example, in a two-lane merging problem, we have two incoming lanes merging into one outgoing lane (see Figure 2). There is no priority for each lane (*i.e.*, no main or secondary lane)[1], and vehicles are not allowed to overtake other vehicles during the process.

Vehicle. We assume that all vehicles are connected and autonomous. It continuously broadcasts its dynamic information, such as current position and speed (*e.g.*, via the Basic Safety Messages of Dedicated Short-Range Communications [1]).

Merging Intersection and Merging Point. A merging intersection is the junction between incoming lanes and the outgoing lane. Merging point is a representative point of the merging intersection.

Merging Manager. A merging manager is a system at the merging intersection, and it can be a roadside unit or a leader of the vehicles[2]. A merging manager periodically solves the passing-order optimization problem, and only the vehicles inside the communication range will be taken into consideration in a single period. More precisely, the merging manager periodically receives the information broadcast from incoming vehicles in the communication range and calculates

[1]Although the priority is not discussed in this paper, it is an easy extension by checking the last row or column in the dynamic programming.

[2]It can also be a distributed system if all information is shared by all vehicles and the lane-merging algorithm is agreed between vehicles in advance.

TABLE I

THE NOTATION IN TWO-LANE MERGING PROBLEM.

Index	i / j	the index of vehicles on Lane A / B
Given (Input)	Δ_i / δ_j	the i-th / j-th vehicle on Lane A / B
	α / β	the number of vehicles on Lane A / B
	a_i / b_j	the earliest arrival time of Δ_i / δ_j
	$W^=$ / W^+	the waiting time if two consecutive vehicles are from the same / different incoming lane(s)
Output	s_i / t_j	the scheduled entering time of Δ_i / δ_j

the *earliest arrival time* of each vehicle. Then, the merging manager computes the passing order and the *scheduled entering time* of each vehicle. After that, the merging manager broadcasts these results and prepares for the next period. Note that the time needed for every vehicle to pass the merging intersection is identical, so the passing order is actually arranged by the scheduled entering time of each vehicle.

Earliest Arrival Time. Each vehicle Δ_i is associated with its earliest arrival time a_i. It is calculated by the merging manager based on the information provided from Δ_i. Vehicle dynamics can be included in the information provided from Δ_i and considered in the calculation. For the calculation, the merging manager assumes that there is no other vehicle in front of Δ_i before the merging point, so it is the earliest possible time for Δ_i to arrive at the merging point.

Waiting Time. Two vehicles are consecutive if they pass the merging point one after the other without any other vehicle between them. Waiting time is the minimum time (*i.e.*, gap time or time headway) needed for two consecutive vehicles to arrive at the merging point. If two consecutive vehicles arrive at the merging point within their waiting time, it is regarded as "unsafe" since they are too close to each other. We assume that the waiting time for two consecutive vehicles from the same incoming lane is $W^=$, while the waiting time for two consecutive vehicles from different incoming lanes is W^+. Generally, W^+ is greater than $W^=$, as the feature of "variant vehicle-following gaps" described in Section I, and the setting of waiting time is exactly how we address the feature in this paper.

Scheduled Entering Time. Each vehicle Δ_i is associated with its scheduled entering time s_i. It is computed by the merging manager and broadcast to Δ_i. After receiving it, Δ_i dynamically adjusts its speed to arrive at the merging point at the scheduled entering time. We assume that the vehicle dynamics is capable of following the scheduled entering time at the merging point.

B. Problem Formulation

The notation used in the two-lane merging problem is summarized in Table I. In the problem, two incoming lanes, *Lane A* and *Lane B*, merge into one outgoing lane. In the merging process, the merging manager computes the scheduled entering time of each vehicle on the incoming lanes and in the communication range.

Suppose there are α vehicles from *Lane A* and β vehicles from *Lane B* in a single period. The earliest arrival time of each vehicle Δ_i and δ_j is a_i and b_j, respectively. Given the waiting time $W^=$ and W^+, the two-lane merging problem is

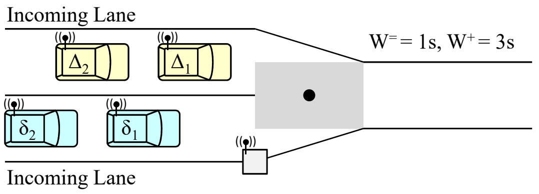

Fig. 3. A two-lane merging instance.

to determine the passing order of these vehicles by assigning the scheduled entering time s_i and t_j to each vehicle Δ_i and δ_j respectively, while minimizing the scheduled entering time of the last vehicle (equivalent to the time needed for all vehicles to pass the merging point)[3]. The formulation is as follows:

$$\begin{aligned} \text{minimize} \quad & \max_{i,j} \{s_i, t_j\} && (1) \\ \text{subject to} \quad & s_i \geq a_i, && (2) \\ & t_j \geq b_j, && (3) \\ & s_{i+1} - s_i \geq W^=, && (4) \\ & t_{j+1} - t_j \geq W^=, && (5) \\ & \left| s_i - t_j \right| \geq W^+, && (6) \\ & i = 1, 2, \ldots, \alpha, \quad j = 1, 2, \ldots, \beta. \end{aligned}$$

The objective function (1) is to minimize the scheduled entering time of the last vehicle. Constraints (2) and (3) state that the scheduled entering time of each vehicle should not be earlier than the earliest arrival time of the vehicle. Constraints (4)–(6) guarantee that every two consecutive vehicles satisfy the requirement of waiting time.

III. ALGORITHM

A. Motivation

Most existing works apply a greedy method which only considers the first vehicles on both lanes as a sub-problem. There is a greedy method to determine the passing order, which is intuitive and easy to compute, *e.g.*, the approach in [8]. The method first sorts the vehicles with their earliest arrival times in ascending order and then computes their scheduled entering times according to this order. Based on sorted arrival times, *First-Arrive-First-Go* algorithm can compute the scheduled entering times in linear time (except sorting). However, the waiting time between two consecutive vehicles is related to whether they are from the same incoming lane or not. We will show that this greedy method cannot find the optimal solution due to the existence and the influence of the waiting time.

Figure 3 illustrates a simple example for the two-lane merging problem. There are two vehicles from *Lane A* and two vehicles from *Lane B*, named $\Delta_1, \Delta_2, \delta_1, \delta_2$, respectively, merging into one outgoing lane. We assume the waiting time is 1 second for $W^=$ and 3 seconds for W^+. Their earliest arrival times are shown in Table II.

[3]We solve the problem from the perspective of the merging manager. An optimal solution to the merging manager (merging system) does not guarantee an optimal solution for each individual vehicle.

TABLE II
THE ARRIVAL TIMES OF THE INSTANCE IN FIGURE 3.

	Earliest Arrival Time	First-Arrive-First-Go	Optimal Solution
		Scheduled Entering Time	
Δ_1	1	1	1
δ_1	2	4	6
Δ_2	3	7	3
δ_2	4	10	7

If we solve their scheduled entering times using the *First-Arrive-First-Go* algorithm, the passing order is $\Delta_1 \rightarrow \delta_1 \rightarrow \Delta_2 \rightarrow \delta_2$. Since any two consecutive vehicles are from different incoming lanes in this order, the waiting time between them is W^+, which is much greater than $W^=$. The resulting scheduled entering time of the last vehicle δ_2 is 10 seconds.

However, we can achieve an optimal solution for this instance with the passing order $\Delta_1 \rightarrow \Delta_2 \rightarrow \delta_1 \rightarrow \delta_2$, which results in the least times of cross lane waiting time W^+. The scheduled entering time of the last vehicle δ_2 can be reduced to 7 seconds, 3 seconds earlier than the result of the *First-Arrive-First-Go* algorithm.

It can be seen that the *First-Arrive-First-Go* algorithm cannot guarantee the optimal solution due to the lack of considering the excess delay caused by the waiting time. We will present a dynamic programming based algorithm that can guarantee the optimal solution in the following section.

B. Dynamic Programming Based Algorithm

Dynamic programming is efficient in solving multistage decision making problems in discrete-time. It decomposes the original problem into a series of sub-problems, and then using forward or backward search algorithm to obtain an optimum policy.

In the two-lane merging problem, the objective is to minimize the scheduled entering time of the last vehicle. Because the vehicles from the same incoming lane cannot arrive the merging point before all vehicles in front of it have passed the merging point, the last vehicle is either the last vehicle from *Lane A* or the last vehicle from *Lane B*. Hence, we can decompose this problem into the following two situations: (1) The last vehicle is from *Lane A*, or (2) the last vehicle is from *Lane B*, and get

$$\max_{i,j} \left\{s_i, t_j\right\} = \max\left\{s_\alpha, t_\beta\right\}. \tag{7}$$

We define $L(i,j,A)$ to represent the minimum last scheduled entering time among the first i vehicles on *Lane A* and the first j vehicles on *Lane B* when the last vehicle is from *Lane A*, while defining $L(i,j,B)$ to represent it when the last vehicle is from *Lane B*. Combining with Equation (7), the objective of this problem

$$\text{minimize} \quad \max_{i,j} \left\{s_i, t_j\right\}$$

is equal to

$$\text{minimize} \quad \min\left\{L(\alpha,\beta,A), L(\alpha,\beta,B)\right\}. \tag{8}$$

Note that both of $L(\alpha,\beta,A)$ and $L(\alpha,\beta,B)$ imply that all vehicles have been scheduled, but they represent different lanes from which the last vehicle.

Algorithm 1: Dynamic programming for two-lane merging

```
Input: $\{a_i\}, \{b_j\}, W^=, W^+$
Output: $\min\left\{L(\alpha,\beta,A), L(\alpha,\beta,B)\right\}$
Initialization;
$L(0,0,A) = L(0,0,B) = 0$ ;
$L(1,0,A) = a_1$;
$L(0,1,B) = b_1$;
for $i \leftarrow 2$ to $\alpha$ do
    $L(i,0,A) = \max\{a_i, L(i-1,0,A) + W^=\}$;
end
for $j \leftarrow 2$ to $\beta$ do
    $L(0,j,B) = \max\{b_j, L(0,j-1,B) + W^=\}$;
end
for $i \leftarrow 1$ to $\alpha$ do
    for $j \leftarrow 1$ to $\beta$ do
        $L(i,j,A) =$
            $\min\{\max\{a_i, L(i-1,j,A) + W^=\},$
                $\max\{a_i, L(i-1,j,B) + W^+\}\}$;
        $L(i,j,B) =$
            $\min\{\max\{b_j, L(i,j-1,A) + W^+\},$
                $\max\{b_j, L(i,j-1,B) + W^=\}$;
        Record the passing order for both cases;
    end
end
Output the results;
```

To solve $L(i,j,A)$, we show that it can be derived from $L(i-1,j,A)$ or $L(i-1,j,B)$ with different waiting times. If the previous vehicle passing the merging point is also from *Lane A*, then we will compare the last vehicle's earliest arrival time, denoted as a_i, with $L(i-1,j,A)$ plus the same lane waiting time, $W^=$. If a_i is greater than $L(i-1,j,A) + W^=$, then $L(i,j,A)$ is equal to a_i, which means that the last vehicle is not be delayed by other vehicles. If a_i is less than $L(i-1,j,A) + W^=$, then $L(i,j,A)$ is equal to $L(i-1,j,A) + W^=$, which means that the last vehicle is delayed. If the previous vehicle passing the merging point is from *Lane B*, which is different from the last vehicle, we can derive $L(i,j,A)$ from $L(i-1,j,B)$ similarly and get

$$\begin{aligned} L(i,j,A) = \min\Big\{&\max\left\{a_i, L(i-1,j,A) + W^=\right\}, \\ &\max\left\{a_i, L(i-1,j,B) + W^+\right\}\Big\}. \end{aligned} \tag{9}$$

We can derive $L(i,j,B)$ from $L(i,j-1,A)$ or $L(i,j-1,B)$ in the same way.

$$\begin{aligned} L(i,j,B) = \min\Big\{&\max\left\{b_j, L(i,j-1,A) + W^+\right\}, \\ &\max\left\{b_j, L(i,j-1,B) + W^=\right\}\Big\}. \end{aligned} \tag{10}$$

Finally, by combining Equations (8)–(10), we can solve the two-lane merging problem by a dynamic programming based algorithm as listed in Algorithm 1. Note that we record the passing order in each step so that we can assign the scheduled entering time to each vehicle after we compute the optimal solution.

IV. Consecutive Lane Merging

A. Problem Formulation

The consecutive lane merging problem is extended from the two-lane merging problem. In a two-lane merging prob-

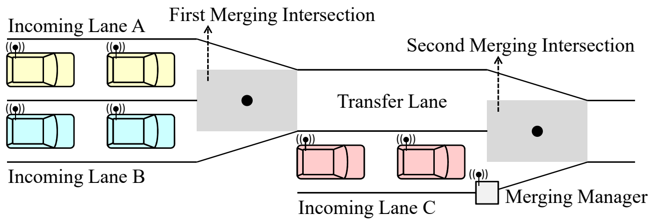

Fig. 4. Consecutive lane merging scenario.

lem, we have only two incoming lanes merged into one outgoing lane through a merging intersection. If there is another incoming lane merged into the outgoing lane at a distance from the first merging intersection and formulate the second merging intersection, then we need to minimize the time for the last incoming vehicle to pass the second merging point. We call this optimization problem is a consecutive lane merging problem (see Figure 4).

For this problem, we follow the definitions in Section II and add new definitions in the following. The additional notation is summarized in Table III.

First Merging Intersection. The first merging intersection is similar to the merging intersection in a two-lane merging problem. Two incoming lanes will merge together through this merging intersection and form a single *Transfer Lane*. We also call the merging point of this merging intersection as the first merging point.

Transfer Lane and Transfer Time. Transfer lane is the lane between the first merging intersection and the *second merging intersection*. After vehicles from the first two incoming lanes are merged and pass the first merging intersection, they will pass through this transfer lane and arrive the second merging intersection. The minimum time needed for each vehicle to pass the transfer lane is called the transfer time, which is identical for every vehicle in our assumption.

Second Merging Intersection and Manager. The second merging intersection is the junction between the transfer lane and the third incoming lane. The merging point of this merging intersection is called the second merging point. In this problem, we place the merging manager in the second merging intersection instead of the first merging intersection[4]. The merging manager should periodically receive the information broadcast from all incoming vehicles in the communication range and minimize the scheduled entering time of the last vehicle to arrive the second merging point.

Earliest Arrival Time. Although there are two merging points in this problem, each vehicle is associated with only one earliest arrival time. For vehicles from the two incoming lanes before the first merging point, their earliest arrival times are their earliest possible times to arrive at the first merging point. For vehicles from the third incoming lane, their earliest arrival times are their earliest possible times to arrive at the second merging point.

Note that in this problem we only need to determine the order of vehicles to pass the second merging point. It is because after vehicles from *Lane A* and *Lane B* are already merged into the transfer lane, they cannot overtake each other since they belong to the same lane. Hence, their order to pass the first merging point will imply their order to pass the second merging point.

[4]It is also possible to put the merging manager in the middle or use a hierarchical structure to coordinate multiple merging managers

TABLE III

THE ADDITIONAL NOTATION IN CONSECUTIVE LANE MERGING.

Index	k	the index of vehicles on Lane C
Given (Input)	ζ_k	the k-th vehicle on Lane C
	γ	the number of vehicles on Lane C
	c_k	the earliest arrival time of ζ_k to the second merging point
	$W_1^=$ / W_1^+	the same / different lane(s) waiting time for the first merging point
	$W_2^=$ / W_2^+	the same / different lane(s) waiting time for the second merging point
	T_f	the transfer time between intersections
Output	s'_i / t'_j	the scheduled entering time of Δ_i / δ_j to the first merging point
	s_i / t_j / u_k	the scheduled entering time of Δ_i / δ_j / ζ_k to the second merging point

We formulate the consecutive lane merging problem as follows. Suppose there are α vehicles from *Lane A*, β vehicles from *Lane B*, and γ vehicles from *Lane C* in a single period. The vehicles from *Lane A* and *Lane B* merge into a transfer lane through the first merging point, and then merge with the vehicles from *Lane C* at the second merging point. Assume the earliest arrival time of each vehicle Δ_i, δ_j, and ζ_k is a_i, b_j, and c_k, respectively. Also given the transfer time T_f and the waiting time $W_1^=$, W_1^+, $W_2^=$, and W_2^+.

The consecutive lane merging problem is to determine the order of these vehicles to pass the second merging point by assigning the scheduled entering time at two merging points s_i, s'_i, t_i, and t'_i to each vehicle Δ_i and δ_j, and assign the scheduled entering time at the second merging point u_k to ζ_k. The formulation is as follows:

$$\begin{aligned}
\text{minimize}\quad & \max_{i,j,k}\{s_i, t_j, u_k\} && (11)\\
\text{subject to}\quad & s'_i \geq a_i, \quad t'_j \geq b_j, \quad u_k \geq c_k, && (12)\\
& s_i - s'_i \geq T_f, \quad t_j - t'_j \geq T_f, && (13)\\
& s'_{i+1} - s'_i \geq W_1^=, \quad t'_{j+1} - t'_j \geq W_1^=, && (14)\\
& \left|s'_i - t'_j\right| \geq W_1^+, && (15)\\
& s_{i+1} - s_i \geq W_2^=, \quad t_{j+1} - t_j \geq W_2^=, && (16)\\
& \left|s_i - t_j\right| \geq W_2^=, \quad u_{k+1} - u_k \geq W_2^=, && (17)\\
& \left|s_i - u_k\right| \geq W_2^+, \quad \left|t_j - u_k\right| \geq W_2^+, && (18)\\
& i = 1, 2, \ldots, \alpha, \quad j = 1, 2, \ldots, \beta, &&\\
& k = 1, 2, \ldots, \gamma. &&
\end{aligned}$$

The objective function (11) is to minimize the scheduled entering time of the last vehicle at the second merging point. Constraint (12) states that the scheduled entering time should not be earlier than the earliest arrival time of each vehicle. Constraint (13) guarantees that there is sufficient transfer time for vehicles from *Lane A* and *Lane B* to pass the transfer lane. Constraints (14)–(18) guarantee that every two consecutive vehicles satisfy the requirement of waiting time at the first and second merging points.

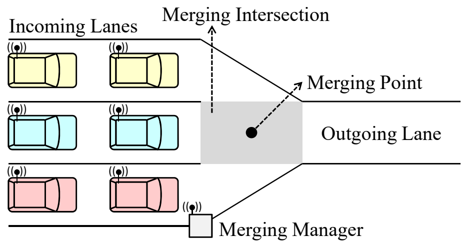

Fig. 5. Three-lane merging scenario.

B. Difficulty of Consecutive Lane Merging

We show the difficulty of solving a consecutive lane merging problem by comparing its difference to a three-lane merging problem. A three-lane merging problem is similar to a two-lane merging problem, except there are three incoming lanes merged into one outgoing lane (See Figure 5). To solve a three-lane merging problem, we can simply extend the algorithm we proposed in Section III-B to consider the last vehicle is from one of the three incoming lanes in each step.

However, the algorithm needs to know the earliest arrival time at the merging point for every vehicle, which is not trivial in a consecutive lane merging problem. When it comes to a consecutive lane merging problem, vehicles from *Lane A* and *Lane B* only have their earliest arrival times to the first merging point. Their earliest arrival times to the second merging point depends on their passing order at the first merging point. Since the objective function of the consecutive lane merging problem is to minimize the scheduled entering time of the last vehicle at the second merging point, we need to modify our dynamic programming algorithm to solve this problem.

C. Modified Dynamic Programming Based Algorithm

In the dynamic programming algorithm proposed in Section III-B, we consider two situations of the last vehicle and compute both cases based on previous results in each step. Because of the difficulty discussed in the previous section, we not only need to know the previous passing order and the scheduled entering time of the previous vehicle at the second merging point, but also have to know the same information at the first merging point.

If the last vehicle passing the second merging point is from *Lane A* or *Lane B*, then the last vehicle passing the first merging point must be from the same incoming lane because the vehicles cannot overtake each other on the transfer lane. If the last vehicle passed the second merging point is from *Lane C*, then we need to record the last vehicle passing the first merging point is from which lane.

Therefore, the problem can be decomposed into four situations: (1) the last vehicle passing both merging points is from *Lane A*, (2) the last vehicle passing both merging points is from *Lane B*, (3) the last vehicle passing the second merging point is from *Lane C*, while the last vehicle passing the first merging point is from *Lane A*, or (4) the last vehicle passing the second merging point is from *Lane C*, while the last vehicle passing the first merging point is from *Lane B*.

For the first situation, we define $L_1(i,j,k,A)$ to represent the last scheduled entering time at the first merging point and $L_2(i,j,k,A)$ to represent the last scheduled entering time at the second merging point. For the second situation, we define them as $L_1(i,j,k,B)$ and $L_2(i,j,k,B)$. For the third and fourth situations, we define them as $L_1(i,j,k,C_A)$, $L_2(i,j,k,C_A)$ and $L_1(i,j,k,C_B)$, $L_2(i,j,k,C_B)$. Then the objective function of this problem

$$\text{minimize} \quad \max_{i,j,k} \left\{ s_i, t_j, u_k \right\},$$

is equal to

$$\begin{aligned} \text{minimize} \quad \min \big\{ & L_2(\alpha,\beta,\gamma,A), L_2(\alpha,\beta,\gamma,B), \\ & L_2(\alpha,\beta,\gamma,C_A), L_2(\alpha,\beta,\gamma,C_B) \big\} . \end{aligned} \tag{19}$$

We can derive $L_2(i,j,k,A)$ from $L_2(i-1,j,k,A)$, $L_2(i-1,j,k,B)$, $L_2(i-1,j,k,C_A)$, and $L_2(i-1,j,k,C_B)$. We detail how to derive $L_2(i,j,k,A)$ from $L_2(i-1,j,k,A)$ as follows. For the derivation, we have to derive $L_1(i,j,k,A)$ from $L_1(i-1,j,k,A)$ first,

$$L_1(i,j,k,A) = \max \left\{ a_i, L_1(i-1,j,k,A) + W_1^{=} \right\} . \tag{20}$$

Then,

$$\begin{aligned} L_2(i,j,k,A) = \max \big\{ & L_1(i,j,k,A) + T_f, \\ & L_2(i-1,j,k,A) + W_2^{=} \big\} . \end{aligned} \tag{21}$$

Equation (20) compares the earliest arrival time at the first merging point a_i of the vehicle Δ_i with $L_1(i-1,j,k,A) + W_1^{=}$, which implies whether Δ_i is delayed at the first merging point or not. Then Equation (21) compares the result from Equation (20) plus the transfer time with $L_2(i-1,j,k,A) + W_2^{=}$, which implies whether Δ_i is delayed at the second merging point or not. By combining Equations (20), (21) and extending them to other situations, we have

$$\begin{aligned} L_2(i,j,k,A) = \min \Big\{ & \\ & \max \Big\{ \max \left\{ a_i, L_1(i-1,j,k,A) + W_1^{=} \right\} + T_f, \\ & \quad L_2(i-1,j,k,A) + W_2^{=} \Big\}, \\ & \max \Big\{ \max \left\{ a_i, L_1(i-1,j,k,B) + W_1^{+} \right\} + T_f, \\ & \quad L_2(i-1,j,k,B) + W_2^{=} \Big\}, \\ & \max \Big\{ \max \left\{ a_i, L_1(i-1,j,k,C_A) + W_1^{=} \right\} + T_f, \\ & \quad L_2(i-1,j,k,C_A) + W_2^{+} \Big\}, \\ & \max \Big\{ \max \left\{ a_i, L_1(i-1,j,k,C_B) + W_1^{+} \right\} + T_f, \\ & \quad L_2(i-1,j,k,C_B) + W_2^{+} \Big\} \Big\}. \end{aligned} \tag{22}$$

$L_2(i,j,k,B)$ can be derived in the same way.

To derive $L_2(i,j,k,C_A)$, we only have to consider $L_2(i,j,k-1,A)$ and $L_2(i,j,k-1,C_A)$ since other cases will change the passing order at the first merging point.

$$\begin{aligned} L_2(i,j,k,C_A) = \min \Big\{ & \max \left\{ c_k, L_2(i,j,k-1,A) + W_2^{+} \right\}, \\ & \max \left\{ c_k, L_2(i,j,k-1,C_A) + W_2^{=} \right\} \Big\}. \end{aligned} \tag{23}$$

Algorithm 2: Dynamic programming for consecutive lane merging

```
Input: $\{a_i\}, \{b_j\}, \{c_k\}, W_1^=, W_1^+, W_2^=, W_2^+, T_f$
Output: $\min\{L_2(\alpha,\beta,\gamma,A), L_2(\alpha,\beta,\gamma,B),$
            $L_2(\alpha,\beta,\gamma,C_A), L_2(\alpha,\beta,\gamma,C_B)\}$
Initialization;
Compute boundary conditions for $i = 0$, $j = 0$, or $k = 0$;
for $i \leftarrow 1$ to $\alpha$ do
    for $j \leftarrow 1$ to $\beta$ do
        for $k \leftarrow 1$ to $\gamma$ do
            Derive $L_2(i,j,k,A)$;
            Derive $L_2(i,j,k,B)$;
            Derive $L_2(i,j,k,C_A)$;
            Derive $L_2(i,j,k,C_B)$;
            Record the passing order and arrival times
            for these four situations;
        end
    end
end
Output the results;
```

Similarly, $L_2(i,j,k,C_B)$ can be derive from $L_2(i,j,k-1,B)$ and $L_2(i,j,k-1,C_B)$.

$$L_2(i,j,k,C_B) = \min\Big\{\max\{c_k, L_2(i,j,k-1,B) + W_2^+\}, \max\{c_k, L_2(i,j,k-1,C_B) + W_2^=\}\Big\}. \tag{24}$$

Finally, we can solve the consecutive lane merging problem by computing and recording the results of these situations in each step. The modified dynamic programming based algorithm is listed in Algorithm 2.

V. Experimental Results

We implemented the proposed algorithm in C++ programming language. Our program was executed on a Linux workstation with Intel Xeon 2.1 GHz CPU and 16 GB memory as well as an embedded platform with Raspberry Pi 2 and Ubuntu 16.04. The traffic is generated at every incoming lane by simulating the earliest arrival time of each vehicle. In practice, the earliest arrival time of each vehicle will be periodically calculated according to its position, kinematics, and dynamics. Since our algorithm focuses on solving the optimal passing order of vehicles, given their earliest arrival times, we assume that the earliest arrival time of each vehicle is fixed and following a Poisson distribution (the inter-arrival time of a Poisson process follows an exponential distribution). As our algorithm is very efficient (all experimental runs were completed in 0.05 and 0.5 second on the workstation and the embedded platform, respectively), if needed, it can also be designed as a periodic task to catch the updates of the earliest arrival times of vehicles.

We use two metrics to evaluate the scheduling results: (1) the scheduled entering time of the last vehicle T_{last} and (2) the average delay time of all vehicles T_{delay}.

The scheduled entering time of the last vehicle T_{last} implies the total time needed for the merging process, and it is the objective function in our dynamic programming based algorithm. For a consecutive lane merging problem, T_{last} is the scheduled entering time at the second merging point.

TABLE IV
Results (in second) for different λ in two-lane merging problem.

λ	First-Arrive-First-Go		Dynamic Programming	
	T_{last}	T_{delay}	T_{last}	T_{delay}
0.1	1057.91	0.73	1057.89	6.58
0.2	525.63	4.44	525.11	2.17
0.3	400.20	36.43	349.89	2.90
0.4	392.90	75.99	260.54	6.13
0.5	387.99	99.77	224.52	14.14

TABLE V
Results (in second) for different numbers of vehicles in two-lane merging problem, where N is the number of vehicles on each incoming lane.

N	First-Arrive-First-Go		Dynamic Programming	
	T_{last}	T_{delay}	T_{last}	T_{delay}
20	78.50	15.83	57.54	3.95
40	156.19	30.25	110.10	4.77
60	233.24	44.53	163.52	5.49
80	316.17	58.89	212.22	5.61
100	392.92	75.74	268.19	5.88

The average delay time T_{delay} is the average difference between each vehicle's scheduled entering time and its possible earliest entering time. The possible earliest entering time is calculated under the assumption that there are no vehicles on other lanes, so it will only be delayed by the vehicles in front of it on the same incoming lane. T_{delay} represents the gap between a solution and a lower bound to this problem. It can reflect the delay time for all vehicles, not only for the last one.

A. Experiments for Two-Lane Merging

In the two-lane merging simulation, we compare the results of our dynamic programming base algorithm with the *First-Arrive-First-Go* greedy method introduced in Section III-A and also in most existing works, *e.g.*, [8]. We show the advantages of our algorithm under the following situations: (1) different λ, (2) different numbers of vehicles, and (3) different vehicle-following gaps.

Different λ. To compare the influence of traffic density, we use different λ values to run the simulations. We set the number of vehicles from both incoming lanes to 100, $W^=$ = 1s, W^+ = 3s. The results are shown in Table IV.

With lower λ values, both algorithms can achieve the minimum T_{last}, which is the objective in our problem formulation. The dynamic programming algorithm reports higher T_{delay} for the case λ = 0.1 because there exist many possible passing orders to achieve minimum T_{last}, and the dynamic programming algorithm may pick the one with higher T_{delay} since it is not in the objective function.

When λ becomes higher, the *First-Arrive-First-Go* algorithm cannot guarantee the minimum T_{last}, and its T_{delay} grows up rapidly. For our dynamic programming algorithm, it can still have the minimum T_{last}, and its T_{delay} grows much slower than the greedy method.

Different Numbers of Vehicles. We use different numbers of vehicles to run the simulations. We set λ = 0.4, $W^=$ = 1s, W^+ = 3s. The results are shown in Table V. The results show that when the number of vehicles increases, our

TABLE VI

RESULTS (IN SECOND) FOR DIFFERENT VEHICLE-FOLLOWING GAPS IN TWO-LANE MERGING PROBLEM.

$W^{=}$	First-Arrive-First-Go		Dynamic Programming	
	T_{last}	T_{delay}	T_{last}	T_{delay}
1	385.49	71.38	266.97	5.31
1.2	411.44	87.13	273.16	13.52
1.4	436.21	96.03	297.11	24.75
1.6	450.79	96.39	330.21	35.37
1.8	473.98	113.08	366.74	57.88
2.0	494.26	115.95	404.02	69.64
2.2	515.26	125.88	442.08	88.20
2.4	535.55	135.93	480.49	106.75
2.6	556.64	137.87	519.96	118.81
2.8	578.28	142.76	559.26	132.71
3.0	598.09	145.27	598.09	145.27

TABLE VII

RESULTS (IN SECOND) FOR DIFFERENT λ IN CONSECUTIVE LANE MERGING PROBLEM.

λ	First-Arrive-First-Go		Dynamic Programming	
	T_{last}	T_{delay}	T_{last}	T_{delay}
0.1	338.73	1.58	338.63	6.50
0.2	186.58	16.74	178.66	3.47
0.3	175.54	45.67	116.56	8.63
0.4	171.82	56.85	103.33	14.78
0.5	161.51	60.20	99.81	19.47

algorithm can generate a more efficient solution for minimum T_{last} and less T_{delay} compared with the greedy method.

Different Vehicle-Following Gaps. To show the advantages of our algorithm under different vehicle-following gaps, we use different $W^{=}$ values to run the simulations. We set the number of vehicles from both incoming lanes to 100, λ = 0.4, W^{+} = 3s. The results are shown in Table VI.

When the vehicle-following gaps are the same whether two consecutive vehicles are from the same lane or different lanes, *i.e.*, $W^{=} = W^{+}$, the optimal solution can be achieved by both algorithms. However, when $W^{=}$ decreases, the *First-Arrive-First-Go* algorithm is farther from the optimal solution which can be achieved by our algorithm. The results show that our algorithm is more effective if the vehicle-following gaps maintained by CACC get improved, which is consistent with the current trend of vehicular technology.

B. Experiments for Consecutive Lane Merging

For consecutive lane merging problem, we also show the results of different λ in comparison with running the *First-Arrive-First-Go* algorithm at both merging points. We set the number of vehicles from all incoming lane to 30, $W_1^{=}$ = 1s, W_1^{+} = 3s, $W_2^{=}$ = 1s, W_2^{+} = 3s and T_f = 3s. The results are shown in Table VII. The results show that for the consecutive lane merging problem, our algorithm can achieve much better T_{last} and T_{delay} when the value of λ exceeds 0.1.

C. Summary

The experimental results demonstrate that our dynamic programming based algorithms can successfully consider the *variant vehicle-following gaps* and achieve optimal solutions due to their *global views*. All experimental runs were completed in 0.05 and 0.5 second on the workstation and the embedded platform, respectively, showing its capability of *real-time decision*. The formulations, algorithms, and experimental results provide *fundamental of conflict resolution* for connected and autonomous vehicles.

VI. CONCLUSION

We formulated the problem and proposed a dynamic programming based algorithm to find the optimal solution for a two-lane merging scenario. The experimental results demonstrated that our algorithm determines an efficient passing order that can optimally minimize the scheduled entering time of the last vehicle and reduce the average delay time. We also formulated the consecutive lane merging problem and proposed a modified dynamic programming based algorithm to solve it. The concepts in solving this problem have great potential for further generalization such as n-lane consecutive merging, roundabouts, neighboring ramps, and combination of high-level and low-level controls.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support from Ministry of Education (MOE) in Taiwan (Grant Numbers: NTU-107V0901 and NTU-108V0901), Ministry of Science and Technology (MOST) in Taiwan (Grant Numbers: MOST-108-2636-E-002-011, MOST-109-2636-E-002-022, MOST-108-2218-E-002-074, MOST-106-2628-E-002-019-MY3, and MOST-108-2221-E-002-099-MY3), MediaTek Inc. (Grant Numbers: MTKC-2018-0167 and MTKC-2019-0070), Maxeda Technology, Global Unichip Corp., Synopsys Inc., and TSMC Ltd as well as the technical assistance from Mr. Ching-Chin Hung.

REFERENCES

[1] J. B. Kenney, "Dedicated short-range communications (DSRC) standards in the united states," *Proceedings of the IEEE*, vol. 99, no. 7, pp. 1162–1182, July 2011.

[2] A. Uno, T. Sakaguchi, and S. Tsugawa, "A merging control algorithm based on inter-vehicle communication," in *IEEE/IEEJ/JSAI International Conference on Intelligent Transportation Systems (ITSC)*, October 1999, pp. 783–787.

[3] J. Rios-Torres and A. A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 18, no. 5, pp. 1066–1077, May 2017.

[4] V. Milanes, J. Godoy, J. Villagra, and J. Perez, "Automated on-ramp merging system for congested traffic situations," *IEEE Transactions on Intelligent Transportation Systems (T-ITS)*, vol. 12, no. 2, pp. 500–508, June 2011.

[5] Y. Wang, W. E, W. Tang, D. Tian, G. Lu, and G. Yu, "Automated on-ramp merging control algorithm based on internet-connected vehicles," *IET Intelligent Transport Systems*, vol. 7, no. 4, pp. 371–379, December 2013.

[6] I. A. Ntousakis, I. K. Nikolos, and M. Papageorgiou, "Optimal vehicle trajectory planning in the context of cooperative merging on highways," *Transportation Research Part C: Emerging Technologies*, vol. 71, pp. 464–488, 2016.

[7] S. Aoki and R. Rajkumar, "A merging protocol for self-driving vehicles," in *ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, April 2017, pp. 219–228.

[8] G. Raravi, V. Shingde, K. Ramamritham, and J. Bharadia, "Merge algorithms for intelligent vehicles," in *Next Generation Design and Verification Methodologies for Distributed Embedded Control Systems*, S. Ramesh and P. Sampath, Eds. Dordrecht: Springer Netherlands, 2007, pp. 51–65.

[9] T. Awal, L. Kulik, and K. Ramamohanrao, "Optimal traffic merging strategy for communication- and sensor-enabled vehicles," in *IEEE International Conference on Intelligent Transportation Systems (ITSC)*, October 2013, pp. 1468–1474.

[10] J. Wu, F. Perronnet, and A. Abbas-Turki, "Cooperative vehicle-actuator system: a sequence-based framework of cooperative intersections management," *IET Intelligent Transport Systems*, vol. 8, no. 4, pp. 352–360, June 2014.

[11] S. van de Hoef, K. H. Johansson, and D. V. Dimarogonas, "Efficient dynamic programming solution to a platoon coordination merge problem with stochastic travel times," *IFAC-PapersOnLine — IFAC World Congress*, vol. 50, no. 1, pp. 4228–4233, 2017.