Passing-Order Decision for Three-to-Two Lane Merging of Connected and Autonomous Vehicles

Cheng-Pei Chien[1], Ben-Hau Chia[1], Ching-Yun Chang[1], Shang-Chien Lin[1]
Iris Hui-Ru Jiang[1], Changliu Liu[2], Chung-Wei Lin[1]
[1]National Taiwan University, Taipei, Taiwan, [2]Carnegie Mellon University, Pittsburgh, PA, USA
{r09922155, r10922060, b10902064, r07943106}@ntu.edu.tw
huirujiang@ntu.edu.tw, cliu6@andrew.cmu.edu, cwlin@csie.ntu.edu.tw

***Abstract*—Lane merging is a common cause of traffic congestion and delays. The advancement of connected and autonomous vehicles can address the problem and improve the traffic efficiency. In this paper, we aim to schedule the passing order of vehicles in a three-to-two lane-merging problem and minimize the time needed for all vehicles to pass the merging intersection. We propose a three-dimensional dynamic-programming-based algorithm to solve the problem. We introduce a grouping-based strategy to reduce the computational overhead. The experimental results and SUMO simulation show that the three-dimensional dynamic-programming-based algorithm achieves a comparable solution quality in much less time than an optimal approach based on mixed integer linear programming. Besides, the grouping-based strategy makes the real-time computation applicable and achieves a balancing trade-off between computation time and solution quality. The dynamic-programming-based algorithm can be further generalized for solving any n-to-(n-1) lane-merging problem.**

***Index Terms*—Connected and Autonomous Vehicles, Lane Merging, Passing-Order Decision, Scheduling.**

I. Introduction

Lane merging is a common cause of traffic congestion and delays, mainly due to the reduction of available lanes and the unpredictable movements of vehicles. When drivers are unsure about how other vehicles will behave, they tend to slow down or stop to avoid accidents, leading to unnecessary disruptions in traffic flow. To ensure safe and efficient merging while minimizing congestion, researchers explore various strategies for avoiding collisions, planning vehicle trajectories, and determining passing order. The recent development of connected and autonomous vehicles (CAVs) integrates communication technologies, such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, and autonomous driving systems into merging strategies. Vehicle communication enables real-time information sharing, reducing uncertainty, and autonomous driving technologies enhance the precision of vehicle movements, improving overall traffic efficiency.

Extensive research has focused on two-to-one lane merging, where two lanes of traffic must be merged into one. While this scenario is conceptually simple, achieving an optimal merging plan that ensures safety and minimizes delays can still be computationally intensive, especially when incorporating real-time constraints and coordination among multiple CAVs. Solutions such as dynamic programming (DP) [1] or mixed-integer linear programming (MILP) have been proposed, but the time complexity of exact algorithms poses a scalability challenge.

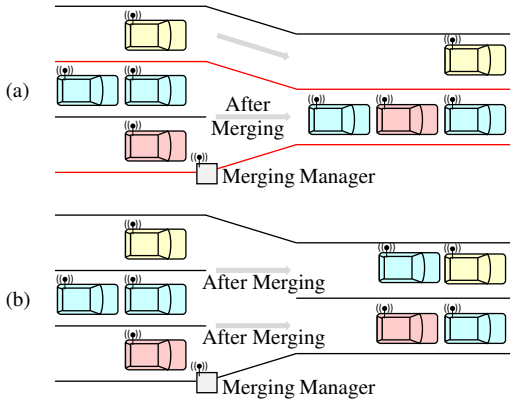

Fig. 1: (a) When we view it as a two-lane merging problem by merging only two of the lanes and simply making a one-to-one mapping for the other lanes, we do not fully utilize the road capacity. (b) When we further merge the other adjacent lanes, then the other lanes can share the traffic loading to improve the traffic efficiency.

In the reality of lane-merging scenarios, the number of lanes can be more than two. For example, a three-lane or four-lane highway has lane reduction or road construction. In a multi-lane merging scenario, though we can just tackle the merging problem for two of the lanes and simply make a one-to-one mapping for the other lanes (as shown in Figure 1(a)), the road capacity is not fully utilized. The other lanes can share the traffic loading to improve traffic efficiency (as shown in Figure 1(b)). To solve the multi-lane merging problem considering the modern features (the global view of incoming vehicles within the communication range, the variant vehicle-following gaps maintained by different autonomous functions,

This work is partially supported by Ministry of Education (MOE) in Taiwan under Grant Number NTU-113V2003-2 and National Science and Technology Council (NSTC) in Taiwan under Grant Number NSTC-112-2221-E-002-168-MY3.

2325-1301/25/$31.00 ©2025 IEEE
DOI 10.1109/RTCSA66114.2025.00012

and the need of real-time decision), the main contributions of this paper are as follows:

- We propose a DP-based algorithm to solve the three-to-two lane-merging problem. Compared with an optimal MILP-based approach, the DP-based algorithm achieves comparable solution quality (traffic efficiency) with much less time.
- We propose a grouping-based strategy to reduce the computational overhead of the DP-based algorithm and make a balancing trade-off between solution quality and computational efficiency.
- We show how to generalize the DP-based algorithm to solve an n-to-(n-1) lane-merging problem.
- We integrate our algorithms with SUMO [2] to further demonstrate the effectiveness and efficiency of our algorithms.

The rest of this paper is organized as follows: Section II reviews the related work. Section III presents the system model and the problem formulation. Section IV outlines the DP-based algorithm for the two-to-one lane-merging problem as a background. Section V introduces our DP-based algorithm for the three-to-two lane-merging problem, the grouping-based strategy to speed up the process, and the generalization from three-to-two to n-to-(n-1) lane-merging scenarios. Section VI provides experimental results. Section VII concludes this paper.

II. Related Work

Numerous studies have investigated merging control for CAVs, focusing on collision avoidance, trajectory optimization, and passing order determination.

Rios-Torres and Malikopoulos [3], as well as Zhu *et al.* [4], provide comprehensive surveys of CAV coordination strategies. Milanes *et al.* adopt the concept of a virtual vehicle and propose a fuzzy controller to modulate the throttle and brake of vehicles, enabling them to reach the desired speed and position in advance for safe merging [5]. Similarly, Wang *et al.* use a virtual platoon approach to ensure collision-free merging operations [6]. Rios-Torres and Malikopoulos take safety and energy optimization into consideration and aim to achieve online coordination of CAVs. Their approach converts the merging problem into an unconstrained optimal control problem and derives a closed-form solution by applying Hamiltonian analysis [7].

Ntousakis *et al.* develop a trajectory planning model with an analytical solution that minimizes engine effort and passenger discomfort using a combined objective function of acceleration and its derivatives [8]. Fukuyama proposes a dynamic game-based approach suitable for decentralized control systems, where each vehicle optimizes its trajectory based on the estimated actions of its competitor [9]. Both methods focus on single-mainline on-ramp merging scenarios. In contrast, Marinescu *et al.* propose a slot-based approach for cooperative merging control in multi-mainline on-ramp scenarios. The approach checks slot availability in the target lane and selects the best one based on acceptable merging acceleration [10].

When it comes to optimizing passing order, Awal *et al.* focus on minimizing delays by using V2V communication to group vehicles, where the group leader determines the optimal sequence by evaluating possible passing orders [11]. Xu *et al.* formulate the problem as a MILP problem and propose a grouping strategy to enhance computational efficiency [12]. Similarly, Kherroubi *et al.* also formulate the problem as an MILP problem but utilize DP to solve it, aiming to maximize traffic throughput [13]. Karbalaieali *et al.* introduce a dynamic adaptive algorithm based on a cost function of travel time for both on-ramp and mainline vehicles, evaluated through simulations under a two-lane mainline scenario with varying traffic demands [14]. Chen and Yang propose a framework that first applies cooperative game theory to mainline vehicles for lane selection based on driving payoff and then the merging sequence containing only vehicles on the inner lane (adjacent to the ramp) and the ramp is also determined using cooperative game theory [15].

Existing studies mainly focus on two-lane merging scenarios, while multi-lane cases are less explored or are typically defined as configurations involving multiple mainlines and a single ramp. In this work, we address the multi-lane merging problem under the assumption that all lanes have equal priority.

III. System Model and Problem Formulation

A. System Model

We consider a three-to-two lane-merging scenario where all vehicles are CAVs moving in the same direction. There is no priority assigned to either lane, and vehicles are not allowed to overtake the others on the same lane. The merging process takes place at a merging intersection, defined as the junction where incoming lanes converge into outgoing lanes. This intersection can be represented by a merging point. A merging manager periodically receives information from incoming vehicles, solves the passing-order optimization problem, and broadcasts the results to the vehicles. The update period can be fixed or dynamically adjusted depending on the system designer's decision. A longer period includes more vehicles within a period, increasing the coordination effectiveness, while a shorter period reduces the computation time. The merging manager can be a roadside unit, a leader of the vehicles, or a distributed system if all vehicles share information and agree on the lane-merging algorithm in advance.

In the process of solving the problem, the merging manager calculates the *earliest arrival time* of each vehicle, considers the *waiting time* between vehicles, and determines the passing order as well as the *scheduled entering time* of each vehicle. The definitions of the earliest arrival time, the waiting time, and the scheduled entering time are provided below, following those in [1].

Earliest Arrival Time. The earliest arrival time of a vehicle is calculated by the merging manager based on the information provided by the vehicle. The merging manager determines the earliest arrival time of a vehicle under the assumption that no other vehicles are present between the

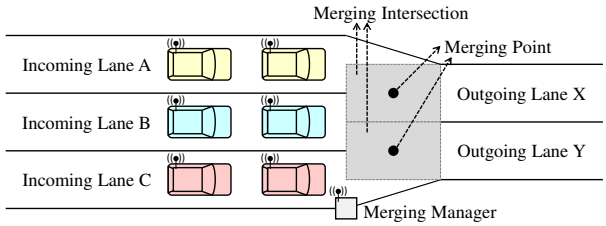

Fig. 2: A three-to-two lane-merging scenario.

vehicle and the merging point, which means there is no congestion or waiting time before the vehicle arrives at the merging point.

Waiting Time. Two vehicles are consecutive if they pass the merging point one after the other without any vehicle in between. The waiting time refers to the minimum time interval between the arrivals of two consecutive vehicles at the merging point. It is unsafe if two consecutive vehicles arrive at the merging point within their waiting time. For consecutive vehicles from the same lane, the waiting time is $W^{=}$, while for vehicles from different lanes, it is W^{+}. Typically, W^{+} is larger than $W^{=}$ since vehicles in the same lane can utilize Cooperative Adaptive Cruise Control (CACC) to maintain shorter following gaps[1].

Scheduled Entering Time. The scheduled entering time of a vehicle is determined by the merging manager. After receiving it from the merging manager, the vehicle adjusts its speed to reach the merging point at the assigned time. We assume that vehicle dynamics enable vehicles to follow their scheduled entering time.

B. Three-to-Two Lane-Merging Problem

As shown in Figure 2, three incoming lanes, *Lane A*, *Lane B*, and *Lane C*, merge into two outgoing lanes, *Lane X* and *Lane Y*. For traffic efficiency and safety, we assume that the possible outgoing lanes for each vehicle must be near its incoming lane. That is, vehicles on *Lane A* can only go to *Lane X*, vehicles on *Lane B* can go to *Lane X* or *Lane Y*, and vehicles on *Lane C* can only go to *Lane Y*.

We define the i-th vehicle on *Lane A*, the j-th vehicle on *Lane B*, and the k-th vehicle on *Lane C* as Δ_i, δ_j, and ϵ_k, respectively. The following parameters are given:

- α / β / γ: the number of vehicles on *Lane A* / *B* / *C*.
- A_i / B_j / C_k: the earliest arrival time of Δ_i / δ_j / ϵ_k.
- $W^{=}$ / W^{+}: the waiting time if two consecutive vehicles are from the same / different incoming lane(s).

Given the parameters above, the three-to-two lane-merging problem is to schedule the passing order of these vehicles and assign the scheduled entering time a_i, b_j, and c_k to each vehicle Δ_i, δ_j, and ϵ_k, respectively, while minimizing the scheduled entering time of the last vehicle with the given waiting time, $W^{=}$ and W^{+}. The formulation is as follows:

$$\min \quad \max_{i,j,k}\{a_i, b_j, c_k\} \tag{1}$$
$$\text{s.t.} \quad a_i \geq A_i, \tag{2}$$
$$b_j \geq B_j, \tag{3}$$
$$c_k \geq C_k, \tag{4}$$
$$a_{i+1} - a_i \geq W^{=}, \quad \text{if } i \neq \alpha, \tag{5}$$
$$b_{j+1} - b_j \geq W^{=}, \quad \text{if } j \neq \beta, \tag{6}$$
$$c_{k+1} - c_k \geq W^{=}, \quad \text{if } k \neq \gamma, \tag{7}$$
$$|a_i - b_j| \geq W^{+}, \quad \text{if } \Delta_i \text{ and } \delta_j \text{ go to } \textit{Lane X}, \tag{8}$$
$$|b_j - c_k| \geq W^{+}, \quad \text{if } \delta_j \text{ and } \epsilon_k \text{ go to } \textit{Lane Y}, \tag{9}$$
$$i = 1, 2, \ldots, \alpha, \quad j = 1, 2, \ldots, \beta, \quad k = 1, 2, \ldots, \gamma.$$

The objective function (1) is to minimize the scheduled entering time of the last vehicle. Constraints (2)–(4) state that the scheduled entering time of each vehicle should not be earlier than the earliest arrival time of the vehicle. Constraints (5)–(7) guarantee that every two consecutive vehicles from the same lane satisfy the requirement of waiting time. Constraints (8)–(9) guarantee that every two consecutive vehicles from different lanes satisfy the requirement of waiting time.

IV. Background: Two-to-One Lane Merging

Most existing methods use a greedy approach, considering only the first vehicles from both lanes as a sub-problem. One such method, the *First-Arrive-First-Go* algorithm, sorts vehicles by their earliest arrival times and computes scheduled entering times accordingly [16]. However, although this method can determine scheduled entering times in linear time, excluding the sorting time, [1] has shown that it does not guarantee the optimal solution due to additional delays caused by waiting times.

A DP-based algorithm is proposed in [1] to solve the two-lane merging problem. The objective is to minimize the scheduled entering time of the last vehicle, which is either the last vehicle from *Lane A* or the last vehicle from *Lane B*. The algorithm defines $L_A(i, j)$ as the minimal scheduled entering time among the first i vehicles from *Lane A* and the first j vehicles from *Lane B* when the last vehicle is from *Lane A*, and $L_B(i, j)$ as the minimal scheduled entering time when the last vehicle is from *Lane B*. Hence, the objective of this problem can be formulated as

$$\text{minimize} \quad \max\left\{L_A(\alpha, \beta), L_B(\alpha, \beta)\right\}, \tag{10}$$

where α and β refer to the number of vehicles from *Lane A* and *Lane B* respectively.

The DP-based algorithm proposed in [1] works as follows. $L_A(i, j)$ can be derived from $L_A(i-1, j)$ or $L_B(i-1, j)$ with corresponding waiting times. If the previous vehicle passing the merging point is also from *Lane A*, the earliest arrival time of the last vehicle, denoted as A_i, is compared with $L_A(i-1, j)+W^{=}$. If A_i is greater than or equal to $L_A(i-1, j)+W^{=}$, then $L_A(i, j) = A_i$, which means that the last vehicle is not

[1]If $W^{=}$ and W^{+} depend on vehicles, they can be given as $W^{=}_{i,i'}$ and $W^{+}_{i,j}$, respectively, where i, i', j are the indices of vehicles. The proposed algorithms can still work without further changes.

delayed. Otherwise, if A_i is smaller than $L_A(i-1,j)+W^=$, then $L_A(i,j) = L_A(i-1,j)+W^=$, which means that the last vehicle is delayed and needs to wait the previous vehicle. Similarly, we can derive $L_A(i,j)$ from $L_B(i-1,j)$ and get

$$L_A(i,j) = \min\Big\{\max\left\{A_i, L_A(i-1,j)+W^=\right\}, \max\left\{A_i, L_B(i-1,j)+W^+\right\}\Big\}. \tag{11}$$

We can derive $L_B(i,j)$ in the same way:

$$L_B(i,j) = \min\Big\{\max\left\{B_j, L_A(i,j-1)+W^+\right\}, \max\left\{B_j, L_B(i,j-1)+W^=\right\}\Big\}. \tag{12}$$

By applying the DP-based algorithm, an optimal passing order can be obtained. Besides, the passing order is recorded in each step and assigned to each vehicle with their scheduled entering time after the solution is computed.

V. Three-to-Two Lane Merging

We modify the DP-based algorithm in Section IV for the three-to-two lane-merging problem (Section V-A). However, the proposed algorithm results in exponential time complexity for optimality, which is the main challenge and difference between the two-to-one and three-to-two lane-merging problems. To address this, we propose a grouping-based strategy to make a good trade-off between solution quality and computational efficiency (Section V-B). We also generalize the approach to the n-to-(n-1) lane-merging problem (Section V-C).

A. Three-Dimensional DP-Based Algorithm

1) Algorithm: Extended from the DP-based algorithm in Section IV, we propose a three-dimensional (3D) DP-based algorithm for the three-to-two lane-merging problem. Considering the last vehicle going to each outgoing lane, this problem can be decomposed into the following four situations:

1) The last vehicle going to *Lane X* is from *Lane A*, and the last vehicle going to *Lane Y* is from *Lane B*.
2) The last vehicle going to *Lane X* is from *Lane A*, and the last vehicle going to *Lane Y* is from *Lane C*.
3) The last vehicle going to *Lane X* is from *Lane B*, and the last vehicle going to *Lane Y* is from *Lane B*.
4) The last vehicle going to *Lane X* is from *Lane B*, and the last vehicle going to *Lane Y* is from *Lane C*.

Based on the four situations above, we decompose this problem into a series of subproblems, and define $L_{AB}(i,j,k)$, $L_{AC}(i,j,k)$, $L_{BB}(i,j,k)$, and $L_{BC}(i,j,k)$ to represent the solutions to the subproblems. For each solution, the first subscript letter indicates which lane the last vehicle going to *Lane X* comes from, and the second subscript letter indicates which lane the last vehicle going to *Lane Y* comes from. In the parentheses are the numbers of vehicles on the three incoming lanes we consider in the subproblem. The solution to each subproblem is a tuple consisting of two elements, t_x and t_y. t_x represents the scheduled entering time of the last vehicle going to *Lane X*, and t_y represents the scheduled entering time of the last vehicle going to *Lane Y*.

To solve $L_{AB}(i,j,k)$, we show that it can be derived from $L_{AB}(i-1,j,k)$, $L_{BB}(i-1,j,k)$, $L_{AB}(i,j-1,k)$, or $L_{AC}(i,j-1,k)$. The last vehicle passing the merging point is possibly the one going from *Lane A* to *Lane X* or the one going from *Lane B* to *Lane Y*. If it is the one that goes from *Lane A* to *Lane X*, then the previous vehicle that also goes to *Lane X* is possibly from *Lane A* or *Lane B*. On the other hand, if the last vehicle passing the merging point is the one that goes from *Lane B* to *Lane Y*, then the previous vehicle that also goes to *Lane Y* is possibly from *Lane B* or *Lane C*. Therefore, considering the last vehicle passing the merging point and the previous vehicle that goes to the same lane, $L_{AB}(i,j,k)$ can be derived from the following four cases:

1) The last vehicle passing the merging point goes from *Lane A* to *Lane X*, and the previous vehicle that also goes to *Lane X* is from *Lane A*.
2) The last vehicle passing the merging point goes from *Lane A* to *Lane X*, and the previous vehicle that also goes to *Lane X* is from *Lane B*.
3) The last vehicle passing the merging point goes from *Lane B* to *Lane Y*, and the previous vehicle that also goes to *Lane Y* is from *Lane B*.
4) The last vehicle passing the merging point goes from *Lane B* to *Lane Y*, and the previous vehicle that also goes to *Lane Y* is from *Lane C*.

When we derive from Case 1, we compute $L_{AB}(i,j,k)$ from $L_{AB}(i-1,j,k)$. Because the last vehicle passing the merging point goes to *Lane X*, we only need to compute the scheduled entering time of the last vehicle going to *Lane X*, denoted as $L_{AB}(i,j,k).t_x$. To compute $L_{AB}(i,j,k).t_x$, we compare the earliest arrival time of the last vehicle, denoted as A_i, with $L_{AB}(i-1,j,k).t_x$ plus the same lane waiting time, $W^=$. If A_i is greater than or equal to $L_{AB}(i-1,j,k).t_x+W^=$, then $L_{AB}(i,j,k).t_x$ is equal to A_i, which means that the last vehicle is not delayed by other vehicles. If A_i is less than $L_{AB}(i-1,j,k).t_x + W^=$, then $L_{AB}(i,j,k).t_x$ is equal to $L_{AB}(i-1,j,k).t_x + W^=$. There is no change on *Lane Y* from $L_{AB}(i-1,j,k)$ to $L_{AB}(i,j,k)$, so $L_{AB}(i,j,k).t_y$ is equal to $L_{AB}(i-1,j,k).t_y$. When we derive from Cases 2, 3, and 4, we can compute $L_{AB}(i,j,k)$ from $L_{BB}(i-1,j,k)$, $L_{AB}(i-1,j,k)$, and $L_{AC}(i-1,j,k)$, respectively, in a similar way and get

$$\begin{aligned}
&L_{AB}(i,j,k)\\
&= (\max\{A_i, L_{AB}(i-1,j,k).t_x + W^=\}, L_{AB}(i-1,j,k).t_y)\\
&= (\max\{A_i, L_{BB}(i-1,j,k).t_x + W^+\}, L_{BB}(i-1,j,k).t_y)\\
&= (L_{AB}(i,j-1,k).t_x, \max\{B_j, L_{AB}(i,j-1,k).t_y + W^=\})\\
&= (L_{AC}(i,j-1,k).t_x, \max\{B_j, L_{AC}(i,j-1,k).t_y + W^+\}).
\end{aligned} \tag{13}$$

We can derive $L_{AC}(i,j,k)$, $L_{BB}(i,j,k)$ and $L_{BC}(i,j,k)$ correspondingly and solve the three-to-two lane-merging problem by a 3D DP-based algorithm as listed in Algorithm 1.

2) Analysis of Time Complexity: In our algorithm, the solution to each subproblem is a tuple containing two elements which are the two scheduled entering times of the last vehicles

Algorithm 1 3D DP-based algorithm for three-to-two lane merging

Input: $\{A_i\}$, $\{B_j\}$, $\{C_k\}$, $W^=$, W^+
Output: $\{a_i\}$, $\{b_j\}$, $\{c_k\}$

1: Create four tables, L_{AB}, L_{AC}, L_{BB}, and L_{BC}, and initialize each cell of them to (∞, ∞)
2: Compute the base cases where any index is 0
3: **for** $i \leftarrow 1$ to α **do**
4: **for** $j \leftarrow 1$ to β **do**
5: **for** $k \leftarrow 1$ to γ **do**
6: Derive $L_{AB}(i,j,k)$, $L_{AC}(i,j,k)$, $L_{BB}(i,j,k)$, $L_{BC}(i,j,k)$
7: Record the outgoing lane of the last vehicle and which source table we derive from for each cell
8: **end for**
9: **end for**
10: **end for**
11: Choose an optimal solution from $L_{AB}(\alpha,\beta,\gamma)$, $L_{AC}(\alpha,\beta,\gamma)$, $L_{BB}(\alpha,\beta,\gamma)$, and $L_{BC}(\alpha,\beta,\gamma)$
12: Backtrack the tables from the optimal solution, and then get its associated passing order and the scheduled entering time of each vehicle
13: Output the results

on *Lane X* and *Lane Y*. However, our objective is to minimize the time needed for all vehicles to pass the merging point, that is, the maximum element of the tuple. Based on the objective, we will show that optimal solutions to the subproblems cannot guarantee an optimal solution to the problem. For example, consider the case where we derive solutions to a problem from its subproblems using the second equation of Equation (13), assuming $W^+ = 3$ and $A_i < L_{BB}(i-1,j,k).t_x + W^+$. Suppose that there are two possible solutions to the subproblem $L_{BB}(i-1,j,k)$, namely $(1,4)$ and $(2,3)$. According to Equation (13), we can obtain two candidate solutions for $L_{AB}(i,j,k)$:

- From $(1,4)$, we get $(\max\{A_i, 1+3\}, 4) = (4,4)$.
- From $(2,3)$, we get $(\max\{A_i, 2+3\}, 3) = (5,3)$.

According to the objective, $(4,4)$ is the optimal solution to the problem. However, it is derived from $(1,4)$, which is not the optimal solution to the subproblem. This shows that choosing only the optimal solution to a subproblem does not guarantee the optimal solution to the full problem. Therefore, if we want to reach an optimal solution at last, we need to store all possible solutions for each subproblem, leading to exponential time complexity.

The exponential time complexity obviously cannot meet the requirement for real-time use. As a result, to reduce the time complexity, we do not store all possible solutions at each step in Algorithm 1. Instead, we store only one solution whose maximum element is the minimum among the four possible solutions, that is, the optimal solution at that step. In this way, although the time complexity is reduced to $O(n^3)$, we cannot guarantee the optimality of the final solution because we lose some possible solutions at the intermediate steps. Furthermore, even though the time complexity is sharply reduced from exponential to polynomial, $O(n^3)$ is still not suitable for practical use because the computation time grows fast as the number of vehicles increases.

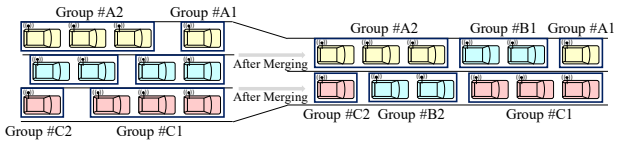

Fig. 3: An example of grouping. The passing order of the vehicles in a group is consecutively fixed.

B. Grouping-Based Strategy

As the waiting time for two consecutive vehicles from the same lane is less than that for two consecutive vehicles from different lanes, when two consecutive vehicles on the same lane can arrive at the merging point at the near time, it is better to put them together in the passing order instead of inserting any vehicle from a different lane between them. For example, assume $W^= = 1$ and $W^+ = 3$, when there are two vehicles Δ_1 and Δ_2 on *Lane A*, and there is one vehicle δ_1 on *Lane B*, and their earliest arrival times are respectively 1, 3 and 2, if the passing order is $\{\Delta_1, \Delta_2, \delta_1\}$, then the scheduled entering time of them can be derived as follows:

- For Δ_1, the scheduled entering time is 1.
- For Δ_2, the scheduled entering time is $\max\{3, 1+1\} = 3$.
- For δ_1, the scheduled entering time is $\max\{2, 3+3\} = 6$.

If the passing order is $\{\Delta_1, \delta_1, \Delta_2\}$, then the scheduled entering time of them can be computed similarly:

- For Δ_1, the scheduled entering time is 1.
- For δ_1, the scheduled entering time is $\max\{2, 1+3\} = 4$.
- For Δ_2, the scheduled entering time is $\max\{3, 4+3\} = 7$.

In this case, the second passing order takes more time for all vehicles to pass the merging point.

With the observation above, we can apply a grouping-based strategy to improve the efficiency of the 3D DP-based algorithm. The main idea of the grouping-based strategy is to view those vehicles which are close enough as a whole when solving the problem. The passing order of the vehicles in a group is consecutively fixed, as shown in Figure 3. In this way, the vehicles in the same group can be viewed as only one vehicle in the DP-based algorithm, thus the number of vehicles in the algorithm decreases, and the computation time is consequently reduced.

To group the vehicles which can arrive at the merging point at the near time, the steps of our approach are as follows:

1) Set a constant value for the maximum number of groups.
2) Set an initial value for the grouping threshold.
3) Calculate the difference between the earliest arrival times of any two consecutive vehicles on the same lane. If the difference is smaller than the grouping threshold, then the two vehicles are grouped into the same group.

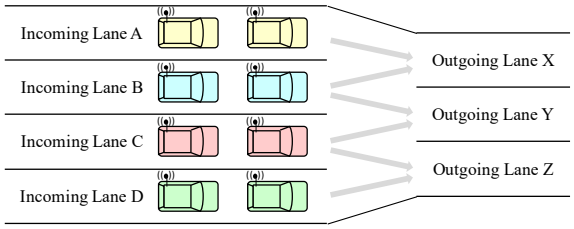

Fig. 4: An example of an n-to-(n-1) lane-merging problem.

4) If the number of groups is greater than the maximum number of groups, then increase the grouping threshold and go to Step 3 to regroup the vehicles. Otherwise, the grouping process is finished.

With the grouping-based strategy, the computation time of the 3D DP-based algorithm can be sharply cut down, but the solution quality may be lowered at the same time. This is predictable because the passing order of the vehicles in the same group is consecutively fixed, which means some possible solutions are omitted. If the grouping threshold is too large, then there will be some good solutions among the omitted solutions, thus the solution quality may be lowered.

C. Generalization

Our main approach, the 3D DP-based algorithm, can be generalized to solve any n-to-(n-1) lane-merging problem. Similar to the three-to-two lane-merging problem, we define the n-to-(n-1) lane-merging problem with the assumption that the possible outgoing lanes for each vehicle must be near its incoming lane. As shown in Figure 4, for each outgoing lane, vehicles may come from 2 possible incoming lanes. Therefore, considering the last vehicle on each outgoing lane, there are 2^{n-1} combinations of them. Based on the DP-based algorithm in Section V-A, we first create 2^{n-1} DP tables as Line 1 in Algorithm 1. The number of the dimensions of each table is equal to the number of incoming lanes, so each table has n dimensions. The length of a dimension should be the number of vehicles on an incoming lane. Hence, the size of each table is the product of the number of vehicles on each incoming lane.

After creating the DP tables, we compute each cell of the tables with nested loops as Lines 3–10 in Algorithm 1. For each cell, an optimal solution can be derived from one of $2 \cdot (n-1)$ possible former solutions because the last vehicle at this step may go to one of the $(n-1)$ outgoing lanes, and the previous vehicle going to that lane may come from 2 possible incoming lanes. We have to calculate the solutions derived from the $2 \cdot (n-1)$ possible former solutions, and then choose one as an optimal solution for the cell. Each solution is a tuple composed of $(n-1)$ numbers which are the scheduled entering times of the last vehicles on the $(n-1)$ outgoing lanes. The solution whose maximum element is minimal among all possible solutions is seen as the optimal one. After computing all the cells of the 2^{n-1} tables, we choose an optimal solution from the last cell of each table. Then, we backtrack the tables from the optimal solution, and finally get the passing order and the scheduled entering time of each vehicle.

To sum up, the steps of solving an n-to-(n-1) lane-merging problem are as follows:

1) Create 2^{n-1} tables. Each table is n-dimensional, and the length of each dimension is the number of vehicles on each incoming lane.
2) Compute all the cells of the 2^{n-1} tables. For each cell, derive possible solutions from $2 \cdot (n-1)$ former solutions and choose the optimal one.
3) Compare the last cell of each table and choose the optimal one.
4) Backtrack the tables from that cell to get the passing order and the scheduled entering time of each vehicle.

For the convenience of complexity analysis, we assume that the number of vehicles on each incoming lane is equal to m. For an n-to-(n-1) lane-merging problem, we need to create 2^{n-1} tables, and each table contains m^n cells. When solving each cell, we need to derive possible solutions from $2 \cdot (n-1)$ former solutions and pick one. Thus, the complexity is $2^{n-1} \cdot m^n \cdot 2 \cdot (n-1)$, where n is the number of incoming lanes and m is the maximum number of vehicles on each incoming lane. Though the complexity is too high for real-time usage, we can apply the grouping-based strategy to reduce computational overhead. Moreover, the number of incoming lanes is seldom more than four in the real world.

VI. Experimental Results

We implement the proposed algorithms in C++ programming language. All the experiments are run on a macOS Monterey notebook with 4 Core 1.4-GHz Intel i5-CPU and 8 GB memory. For the input of the algorithms, we randomly generate the earliest arrival time of each vehicle following a Poisson distribution. Since our algorithm focuses on finding an optimal passing order of vehicles, we assume the earliest arrival time of each vehicle is given and fixed in our experiments. However, in practice, the earliest arrival time of each vehicle will be periodically computed according to its position, kinematics, and dynamics. To compare the performance of different algorithms, we report three metrics in the results: T_{last}, T_{delay}, and T_{exec}. All the metrics are in seconds, and their meanings are as follows:

- T_{last}: The scheduled entering time of the last vehicle. It implies the total time needed for all vehicles to pass the merging point.
- T_{delay}: The average difference between each vehicle's scheduled entering time and its earliest arrival time. It reflects the delay time for all vehicles, not only for the last one.
- T_{exec}: The execution time of the program, representing the computational efficiency of the algorithms.

A. Main Experiments

In the main experiments, we analyze the results of five approaches: (1) First-Arrive-First-Go (FAFG), (2) MILP, (3) 3D DP, (4) Grouping MILP, and (5) Grouping 3D DP. For

TABLE I: Results (in second) for different traffic densities. An asterisk means the MILP solver cannot complete the process of finding an optimal solution within the time limit for some test cases.

λ	FAFG			MILP			3D DP			Grouping MILP			Grouping 3D DP		
	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}
0.4	302.40	29.46	8e-5	270.10	5.94	40.95	270.10	1.54	9.80	270.40	31.55	2.13	270.40	3.28	0.29
0.5	306.20	53.22	8e-5	209.60	6.41	174.48	209.70	2.66	9.79	210.70	14.01	1.99	210.90	3.47	0.16
0.6	306.20	72.56	8e-5	180.80*	11.34*	949.12*	179.00	5.31	9.80	179.20*	13.31*	181.44*	180.50	6.67	0.10
0.7	313.90	88.53	7e-5	166.50*	15.64*	1,800.00*	160.80	11.79	9.80	162.20*	12.88*	1,234.91*	162.50	12.43	0.22
0.8	322.30	101.74	7e-5	159.60*	20.27*	1,800.00*	156.90	18.14	9.78	158.10*	19.00*	1,008.80*	159.50	18.82	0.08

TABLE II: Results (in second) for different number of vehicles. An asterisk means the MILP solver cannot complete the process of finding an optimal solution within the time limit for some test cases.

N	FAFG			MILP			3D DP			Grouping MILP			Grouping 3D DP		
	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}
20	60.90	14.48	4e-5	39.20	4.98	4.88	39.20	3.12	0.09	39.40	4.77	0.10	39.50	3.22	0.01
40	120.50	27.55	5e-5	74.20*	5.22*	741.08*	73.80	3.90	0.67	74.10	5.56	1.30	74.30	4.03	0.05
60	181.90	42.09	5e-5	107.40*	5.98*	1,287.13*	106.80	4.43	2.14	107.00	5.17	6.58	108.10	4.80	0.14
80	242.50	53.68	6e-5	146.50*	7.60*	768.37*	145.20	4.39	5.09	146.20	9.01	1.93	146.70	5.50	0.11
100	306.20	72.56	8e-5	180.80*	11.34*	949.12*	179.00	5.31	9.80	179.20*	13.31*	181.44*	180.50	6.67	0.10

TABLE III: Results (in second) for different vehicle-following gaps. An asterisk means the MILP solver cannot complete the process of finding an optimal solution within the time limit for some test cases.

$W^=$ (s)	FAFG			MILP			3D DP			Grouping MILP			Grouping 3D DP		
	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}
1.0	306.20	72.56	8e-5	180.80*	11.34*	949.12*	179.00	5.31	9.80	179.20*	13.31*	181.44*	180.50	6.67	0.10
1.5	342.80	89.37	8e-5	234.50*	36.81*	1,800.00*	229.2	32.86	9.84	230.95*	33.99*	1,780.87*	232.65	33.53	0.10
2.0	381.00	106.25	8e-5	301.20*	68.38*	1,800.00*	300.60	67.87	9.86	302.40*	68.96*	1,800.00*	302.70	68.08	0.10
2.5	422.85	125.64	9e-5	374.25*	104.76*	1,800.00*	374.05	104.60	9.79	375.15*	105.16*	1,800.00*	378.50	105.11	0.10
3.0	466.00	145.53	8e-5	448.10*	141.63*	1,800.00*	448.00	141.58	9.94	448.00*	141.58*	1,800.00*	450.40	141.59	0.10

the MILP-based approaches, we formulate the three-to-two lane-merging problem as an MILP problem and use Gurobi Optimizer (9.1.2) as the solver for the optimization problem. For the 3D DP approaches, we adopt the sub-optimal one with $O\left(n^3\right)$ time complexity. For the grouping-based approaches, we set the maximum number of groups to 35. The experiments are conducted under the following scenarios: (1) different densities (λ), (2) different numbers of vehicles (N), and (3) different vehicle-following gaps ($W^=$). We adjust the density of the vehicles by assigning different values to the parameter λ in the Poisson distribution, and we use the Poisson distribution to generate the earliest arrival time for each vehicle. Note that N is the number of vehicles on *each* lane. The results are shown in Tables I–III, respectively. Each value in the tables is the average value calculated from the results of 10 test cases. Note that we limit the execution time to 1,800 seconds to prevent the MILP solver from taking too much time to solve the optimization problem. In Tables I–III, an asterisk means the MILP solver cannot complete the process of finding an optimal solution within the time limit for some test cases, thus the output solution may not be optimal.

1) Different Traffic Densities: In this scenario, we set N = 100, $W^=$ = 1s, and W^+ = 3s. Table I shows that FAFG takes the least T_{exec}, but the other four approaches outperform FAFG in T_{last} and T_{delay} in almost all cases. When λ becomes higher, the other four approaches have a much greater advantage over T_{last} and T_{delay}. Comparing MILP and 3D DP, 3D DP can achieve comparable T_{last} to that of MILP while it takes much less execution time. When we apply the grouping-based strategy in the MILP-based algorithm, though the execution time can be reduced a lot, it still can not be comparable with 3D DP's execution time. Even though 3D DP is much faster than MILP, it is not sufficient for the real-time requirement. However, Grouping 3D DP can be done in 0.3 seconds and the solution quality is close to that of 3D DP. Though the difference between the solutions of Grouping 3D DP and 3D DP grows larger when λ becomes higher, the solution of Grouping 3D DP is still much better than that of FAFG. Comparing T_{delay} of the DP-based approaches and the MILP-based approaches, we can observe that T_{delay} of the DP-based approaches are always smaller, even if T_{last} of them are larger.

2) Different Number of Vehicles: In this scenario, we set λ = 0.6, $W^=$ = 1s, and W^+ = 3s. As shown in Table II, compared with FAFG, the other four approaches have a much greater advantage over T_{last} and T_{delay} when the number of vehicles increases. T_{exec} of MILP, 3D DP, and Grouping MILP grow quickly when the number of vehicles increases while Grouping 3D DP always solves the problem within 0.2 seconds.

3) Different Vehicle-Following Gaps: In this scenario, we set λ = 0.6, N = 100, and W^+ = 3s. As shown in Table III, when $W^=$ decreases, the solution quality of FAFG is farther from those of the other four approaches. This implies our approaches get more advantageous when the vehicle-following gaps maintained by CACC get improved.

4) Summary: During the main experiments, we show that Grouping 3D DP is the best approach because it not only can solve the problem in real time but also can achieve a comparable solution with the optimal one. MILP and Grouping MILP cannot solve the problem in real time. Though 3D DP can solve the problem in real time when the number of vehicles is small, the execution time grows quickly when the number

TABLE IV: Results (in second) for different traffic densities.

λ	Pruned 3D DP			2-D DP			3D DP with window size = 5			3D DP with window size = 10			3D DP with window size = 20		
	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}
0.4	270.10	1.59	3.30	270.20	5.73	2e-3	270.20	6.60	0.03	270.10	5.04	0.10	270.10	3.54	0.40
0.5	209.70	2.68	2.80	211.90	14.69	2e-3	212.90	7.06	0.03	210.20	4.43	0.10	209.70	3.51	0.40
0.6	179.00	5.27	9.68	183.60	24.49	2e-3	193.20	14.93	0.03	182.40	9.01	0.10	179.90	6.92	0.40
0.7	160.80	11.79	10.25	167.60	28.23	2e-3	191.20	25.44	0.03	174.10	17.46	0.10	165.10	13.72	0.40
0.8	156.90	18.14	10.23	158.90	29.94	2e-3	190.30	33.83	0.03	173.40	25.64	0.10	162.70	20.56	0.40

TABLE V: Results (in second) for different number of vehicles.

N	Pruned 3D DP			2-D DP			3D DP with window size = 5			3D DP with window size = 10			3D DP with window size = 20		
	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}
20	39.20	3.16	0.08	40.30	6.70	1e-4	41.10	4.61	6e-3	40.00	3.73	0.02	39.20	3.13	0.09
40	73.80	3.90	0.59	76.30	11.30	4e-4	79.30	6.85	0.01	75.30	4.94	0.04	74.20	4.20	0.16
60	106.80	4.37	2.02	110.60	15.63	7e-4	116.90	9.30	0.02	109.80	6.16	0.06	107.90	5.13	0.24
80	145.20	4.41	4.62	149.60	19.46	1e-3	157.10	11.36	0.02	147.80	7.13	0.08	145.90	5.46	0.32
100	179.00	5.27	9.68	183.60	24.49	2e-3	193.20	14.93	0.03	182.40	9.01	0.10	179.90	6.92	0.40

TABLE VI: Results (in second) for different vehicle-following gaps.

$W^{=}$ (s)	Pruned 3D DP			2-D DP			3D DP with window size = 5			3D DP with window size = 10			3D DP with window size = 20		
	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}	T_{last}	T_{delay}	T_{exec}
1.0	179.00	5.27	9.68	183.60	24.49	2e-3	193.20	14.93	0.03	182.40	9.01	0.10	179.90	6.92	0.40
1.5	229.20	32.81	10.26	230.80	47.91	2e-3	255.45	45.52	0.03	242.80	39.19	0.10	234.15	35.21	0.40
2.0	300.60	67.82	10.26	300.80	81.98	2e-3	323.10	79.16	0.03	309.40	72.44	0.10	305.40	70.33	0.40
2.5	374.05	104.59	10.23	374.40	118.73	2e-3	384.30	109.69	0.03	378.55	106.87	0.10	376.50	105.80	0.40
3.0	448.00	141.58	10.46	448.00	155.62	2e-3	448.00	141.58	0.03	448.00	141.58	0.10	448.00	141.58	0.41

of vehicles increases. FAFG can always solve the problem in real time, but the solution quality is much worse than that of Grouping 3D DP. Besides, the advantage of Grouping 3D DP is greater when the traffic density becomes higher, the number of vehicles increases, or the vehicle-following gaps maintained by CACC get improved.

B. Additional Experiments

In the additional experiments, we implement three additional approaches which can reduce the computational overhead of the DP-based algorithm, and then compare the results with 3D DP and Grouping 3D DP approaches. The three additional approaches are as follows:

- Pruned 3D DP: When A_{i+1}, B_{j+1} and C_{k+1} are all greater than $L_{AB}(i,j,k)$, $L_{AC}(i,j,k)$, $L_{BB}(i,j,k)$ or $L_{BC}(i,j,k)$, we can be certain that Δ_{i+1}, δ_{j+1} and ϵ_{k+1} will not affect the solution for the vehicles in front of them. Therefore, we can prune some possible passing orders, skip solving those subproblems during the process of computing the DP tables, and thus reduce the computational overhead.
- 2-D DP: This approach views the three-to-two lane-merging problem as a combination of two two-to-one lane-merging problems: one is to merge *Lane A* and *Lane B* to *Lane X*, and the other is to merge *Lane C* and *Lane B* to *Lane Y*. This approach decides the outgoing lane for the vehicles on *Lane B* one by one based on the DP-based algorithm in Section IV. By reducing the dimension of the problem, this approach achieves lower computational complexity at the cost of potentially losing the optimality of DP subproblems.
- Window-based 3D DP: This approach divides the vehicles into fixed-size windows based on the order of their earliest arrival times, and then solves the passing order for vehicles in one window at a time. For example, when the window size is set to 5, then Δ_1–Δ_5, δ_1–δ_5 and ϵ_1–ϵ_5 are contained in the first window. This approach reduces the problem size per DP computation, enabling faster overall computation.

The experiments are also conducted under different densities (λ), different numbers of vehicles (N), and different vehicle-following gaps ($W^{=}$ and W^{+}). The results are shown in Tables IV–VI. Each value in the tables is the average value calculated from the results of 10 test cases.

1) Different Traffic Densities: In this scenario, we set N = 100, $W^{=}$ = 1s, and W^{+} = 3s. Comparing Pruned 3D DP in Table IV with 3D DP in Table I, Pruned 3D DP is more efficient than 3D DP only when λ is low. When λ is higher than 0.6, Pruned 3D DP even takes more time than the original 3D DP does because it needs extra time at each step to decide whether the following steps can be skipped, but no step can be skipped when the traffic is dense. Comparing 2-D DP and the window-based 3D DP in Table IV, 2-D DP is more effective when λ is high, and the window-based 3D DP is more effective when λ is low. However, 2-D DP always tends to result in larger T_{delay}. For T_{exec}, 2-D DP is the second most efficient approach, which only loses to FAFG. Comparing the 3D DP with different window sizes, when λ is getting higher, the difference between their T_{last} is getting larger. Furthermore, comparing 3D DP with window size = 20 in Table IV with Grouping 3D DP in Table I, when λ is

TABLE VII: Results (in second) for different traffic densities in SUMO. An asterisk means the MILP solver cannot complete the process of finding an optimal solution within the time limit for some test cases.

λ	FAFG		MILP		3D DP		Grouping MILP		Grouping 3D DP	
	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}
0.4	847.65	57.88	815.75*	39.15*	807.85	36.27	800.00	33.26	811.05	37.50
0.5	835.80	66.77	802.50*	48.34*	810.70	47.39	791.35	42.37	800.75	43.78
0.6	848.25	75.43	820.90*	55.26*	817.75	50.40	804.60	44.39	800.10	46.03
0.7	833.15	69.20	809.35*	52.36*	795.55	48.34	793.70	45.89	802.75	48.66
0.8	830.60	71.98	796.65*	48.53*	797.00	47.38	788.85	43.97	814.95	52.37

TABLE VIII: Results (in second) for different number of vehicles in SUMO. An asterisk means the MILP solver cannot complete the process of finding an optimal solution within the time limit for some test cases.

N	FAFG		MILP		3D DP		Grouping MILP		Grouping 3D DP	
	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}
20	465.65	9.55	465.35	7.66	465.10	7.41	464.10	7.05	464.90	7.66
40	566.30	27.39	548.65*	20.21*	551.10	19.72	548.35	18.89	552.10	19.97
60	636.20	31.42	629.45*	29.33*	626.25	28.48	622.90	26.84	627.90	28.94
80	740.00	58.07	712.65*	42.45*	708.95	42.96	717.90	43.09	716.95	41.28
100	848.25	75.43	820.90*	55.26*	817.75	50.40	804.60	44.39	800.10	46.03

TABLE IX: Results (in second) for different vehicle-following gaps in SUMO. An asterisk means the MILP solver cannot complete the process of finding an optimal solution within the time limit for some test cases.

$W^=$ (s)	FAFG		MILP		3D DP		Grouping MILP		Grouping 3D DP	
	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}	T_{last}	T_{delay}
1.0	848.25	75.43	820.90*	55.26*	817.75	50.40	804.60	44.39	800.10	46.03
1.5	827.85	44.25	799.05*	28.94*	797.20	26.73	793.20	28.20	817.25	33.13
2.0	892.70	97.91	846.20*	77.11*	834.30	69.52	850.50	71.75	852.50	76.15
2.5	925.70	114.31	884.70*	97.17*	872.00	92.42	886.30	96.69	885.60	96.96
3.0	889.15	71.82	874.65*	62.83*	829.65	48.20	889.95	57.15	893.00	61.47

higher than 0.6, Grouping 3D DP can reach better solutions within less execution time.

2) Different Numbers of Vehicles: In this scenario, we set λ = 0.6, $W^=$ = 1s, and W^+ = 3s. Comparing Pruned 3D DP in Table V with 3D DP in Table II, T_{exec} of Pruned 3D DP is always less than that of 3D DP, and the difference between their T_{exec} is almost not affected by the number of vehicles. Comparing 2-D DP with the window-based 3D DP in Table V, 2-D DP wins on T_{last} only when the window size is 5. Comparing the window-based 3D DP with different window sizes, when the number of vehicles is getting higher, the difference between their T_{last} is getting larger. Comparing 3D DP with window size = 20 in Table V with Grouping 3D DP in Table II, Grouping 3D DP can reach comparable solutions within less execution time, and its execution time does not increase a lot when the number of vehicles gets higher.

3) Different Vehicle-Following Gaps: In this scenario, we set λ = 0.6, N = 100, and W^+ = 3s. Comparing 2-D DP with 3D DP in Table III, when $W^=$ is large, 2-D DP can win 3D DP a little on T_{last}, but results in much greater T_{delay}. Comparing the window-based 3D DP with different window sizes, when the $W^=$ is equal to W^+, there is no difference between their solution. When $W^=$ decreases, 3D DP with a larger window size results in a much better solution than 3D DP with a smaller window size does. Comparing 3D DP with window size = 20 in Table VI with Grouping 3D DP in Table III, it seems that the difference between $W^=$ and W^+ does not affect the difference between their solutions.

4) Summary: With the results of additional experiments, we can draw conclusions about the properties of each additional approach. Pruned 3D DP can reduce the computational overhead of 3D DP only when λ is less than 0.6. 2-D DP is very efficient, but it tends to result in large T_{delay}. 2-D DP has an advantage over T_{last} when the traffic density is high. Window-based 3D DP is suitable for use when the traffic density is low. A larger window size results in better solution quality, and the advantage gets larger when the traffic density becomes higher, the number of vehicles increases, or the difference between $W^=$ and W^+ becomes larger. Comparing the window-based 3D DP and Grouping 3D DP, Grouping 3D can reach better solutions within less execution time. As a result, Grouping 3D makes a balancing trade-off between computation time and solution quality.

C. Simulation with SUMO

The experiments above focus on solving an optimal passing order (high-level control) of vehicles. To further test the effectiveness of our approaches in practical scenarios with low-level control, we simulate the three-to-two lane-merging scenario and implement our algorithms in SUMO 1.9.2. For the MILP-based approach, we integrate SUMO with Gurobi Optimizer to achieve the implementation. To retrieve the information about vehicles and change vehicles' states to follow the scheduled

passing order in the simulation scenarios, we utilize TraCI [17] as an interface to communicate with SUMO.

The departure time is the time at which the vehicle departs from the start point of the incoming lane. Unlike the above experiments that the earliest arrival times are given, in the simulations we estimate the earliest arrival time of each vehicle based on its velocity, acceleration, and distance from the merging intersection. Then we use the estimated earliest arrival times as the input of our algorithms to solve the passing order. To make the vehicles follow the passing order and simulate the following-gap, we set each vehicle's desired time headway and its reference vehicle based on the passing order, $W^{=}$, and W^{+}. In this way, each vehicle changes its headway smoothly to the given value with respect to the given reference vehicle.

We perform simulations to test the effectiveness of the main approaches in Section VI-A. Table VII–IX show the results of simulation. For the traffic in the simulations, we randomly generate the departure time of each vehicle following a Poisson distribution. The conclusions we get from the simulation results are almost the same. The only difference is that applying the grouping strategy can sometimes get better results. This demonstrates that low-level control somehow affects the effectiveness of the proposed approaches. We can observe two main differences between the simulations and the experiments. One is that we estimate the earliest arrival time of each vehicle in the simulation, but the estimation may not be accurate. The other difference is that we make the vehicles follow the passing order by using the function provided by TraCI, but vehicles may not accurately pass the merging intersection at the scheduled entering time. Though there are some deviations in the simulations, the results show that when we consider the scenarios with low-level control, Grouping 3D DP is still a good approach that can solve the passing order in real time and outperforms FAFG.

VII. Conclusion

We formulated a three-to-two lane-merging problem and proposed a 3D DP-based algorithm. To reduce the computational overhead of the 3D DP-based algorithm, we introduced a grouping-based strategy. The experimental results showed that the 3D DP-based algorithm can achieve a comparable solution in much less time than the optimal MILP-based approach. When the number of vehicles is large, the grouping-based strategy enabled the 3D DP-based algorithm to solve the problem in real time and made a balancing trade-off between computation time and solution quality. Moreover, we conducted simulations with SUMO to further test the effectiveness of our algorithm in practical scenarios with low-level control. The simulation results showed that the DP-based algorithm and the grouping strategy can perform well with low-level control. We also explained how to generalize the DP-based algorithm for solving any n-to-(n-1) lane-merging problem.

Future work includes co-design of high-level control and low-level control, guaranteeing the effectiveness of the algorithm in anomalous cases where some vehicles provide erroneous information or some vehicles do not follow the scheduled passing order, and extending scenarios to cover the mixed traffic of autonomous vehicles and human-driven vehicles.

References

[1] S.-C. Lin, H. Hsu, Y.-T. Lin, C.-W. Lin, I. H.-R. Jiang, and C. Liu, "A dynamic programming approach to optimal lane merging of connected and autonomous vehicles," in *IEEE Intelligent Vehicles Symposium*, 2020, pp. 349–356.

[2] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wiessner, "Microscopic traffic simulation using SUMO," in *IEEE International Conference on Intelligent Transportation Systems*, 2018, pp. 2575–2582.

[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*, vol. 18, no. 5, pp. 1066–1077, 2017.

[4] J. Zhu, S. Easa, and K. Gao, "Merging control strategies of connected and autonomous vehicles at freeway on-ramps: A comprehensive review," *Journal of Intelligent and Connected Vehicles*, vol. 5, no. 2, pp. 99–111, 2022.

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

[6] 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, 2013.

[7] J. Rios-Torres and A. A. Malikopoulos, "Automated and cooperative vehicle merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 4, pp. 780–789, 2017.

[8] 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.

[9] S. Fukuyama, "Dynamic game-based approach for optimizing merging vehicle trajectories using time-expanded decision diagram," *Transportation Research Part C: Emerging Technologies*, vol. 120, p. 102766, 2020.

[10] D. Marinescu, J. Čurn, M. Bouroche, and V. Cahill, "On-ramp traffic merging using cooperative intelligent vehicles: A slot-based approach," in *IEEE International Conference on Intelligent Transportation Systems*, 2012, pp. 900–906.

[11] 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*, 2013, pp. 1468–1474.

[12] H. Xu, S. Feng, Y. Zhang, and L. Li, "A grouping-based cooperative driving strategy for CAVs merging problems," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 6, pp. 6125–6136, 2019.

[13] Z. E. A. Kherroubi, S. Aknine, and R. Bacha, "Novel decision-making strategy for connected and autonomous vehicles in highway on-ramp merging," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 12 490–12 502, 2022.

[14] S. Karbalaieali, O. A. Osman, and S. Ishak, "A dynamic adaptive algorithm for merging into platoons in connected automated environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 10, pp. 4111–4122, 2020.

[15] R. Chen and Z. Yang, "Cooperative ramp merging strategy at multi-lane area for automated vehicles," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 10, pp. 14 326–14 340, 2024.

[16] 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. Springer Netherlands, 2007, pp. 51–65.

[17] A. Wegener, M. Piorkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: An interface for coupling road traffic and network simulators," *Communications and Networking Simulation Symposium*, pp. 155–163, 2008.