GeoInformatica (2023) 27:491–523
https://doi.org/10.1007/s10707-022-00473-2

Real-time road safety optimization through network-level data management

Lakmal Muthugama[1] · **Hairuo Xie**[1] · **Egemen Tanin**[1] · **Shanika Karunasekera**[1]

Received: 22 November 2021 / Revised: 11 June 2022 / Accepted: 4 August 2022 /
Published online: 22 August 2022

Abstract
With the increasing connectedness of vehicles, real-time spatio-temporal data can be collected from citywide road networks. Innovative data management solutions can process the collected data for the purpose of reducing travel time. However, a majority of the existing solutions have missed the opportunity to better manage the collected data for improving road safety at the network level. We propose an efficient data management framework that uses network-level data to improve road safety for citywide applications. Our framework uses a graph-based data structure to maintain real-time network-level traffic data. Based on the graph, the framework uses a novel technique to generate driving instructions for individual vehicles. By following the instructions, inter-vehicular spacing can be increased, leading to an improvement of road safety. Experimental results show that our framework improves road safety, measured based on the time to collision between vehicles, from the state-of-the-art traffic data management solutions by a large margin while achieving lower travel times compared with the solutions. The framework is also readily deployable for large-scale real-time applications due to its low computation costs.

Keywords Spatio-temporal data management · Connected autonomous vehicles · Real-time optimization · Intelligent transportation systems

✉ Hairuo Xie
xieh@unimelb.edu.au

Lakmal Muthugama
tmuthugama@student.unimelb.edu.au

Egemen Tanin
etanin@unimelb.edu.au

Shanika Karunasekera
karus@unimelb.edu.au

[1] School of Computing & Information Systems, The University of Melbourne, Parkville, Victoria, Australia

1 Introduction

With the increasing connectedness of vehicles, a vast amount of spatio-temporal data can be collected from large road networks at real time. An example of the data is the routes that vehicles will follow to reach their destinations. Using the collected data, it is possible to enhance traditional traffic engineering systems through integrating them with information systems [1]. A majority of the existing data-driven solutions are focused on reducing travel time [1–5] and have missed the opportunity to better manage the spatio-temporal data for improving road safety at the network level. To maximize road safety at the network level, there needs a global coordination of safety arrangements for all the individual vehicles. Although the current development of connected autonomous vehicle (CAV) technology can help improve road safety, the safety improvement is generally achieved within a limited scope such as a vehicle platoon [6] or an intersection [7], which is different to the coordinated safety arrangements at the network level. Our work is focused on network-level data management for improving road safety.

We aim to develop a data management solution that will give drivers detailed instructions for improving road safety based on the drivers' routes for the future. An ideal solution needs to address two major challenges. The first challenge is to strike a balance between road safety and travel time. Without a proper management scheme, an improved traffic efficiency can lead to a lower safety level [8]. On the other hand, it would be unwise to implement a naive data management solution that leads to the slowdown of all the vehicles. Such a naive solution would make road safer, but it would lead to long travel times for individuals. Therefore, the ideal solution needs to consider road safety improvement and its impact on travel time at the same time. The second challenge is to maintain a high level of computation efficiency for real-time applications. Due to the complexity and the amount of spatio-temporal data, the computation time spent on processing the data can be prohibitive for real-time applications. A number of emerging solutions use sophisticated techniques that are only computationally feasible for optimization in limited scopes such as a single road segment or a single intersection [9, 10]. There is a lack of computationally efficient solutions for improving network-level road safety.

We formalize our research problem based on the balance between road safety and travel time. Our research problem is named **Network-Level Safety Optimization (NSO)**. To achieve the balance between the two factors, our preliminary work develops Safest Platooning Graph (SPG), which is a prototype solution to give driving directions for safety improvements [11]. To build SPG, we model traffic conflicts in a graph and then resolve those conflicts on First-Come-First-Serve (FCFS) basis to decide the vehicle passing order at the conflicting locations, including lanes and intersections inside the road network. Then we space out vehicles iteratively while obeying the vehicle passing order and build the SPG. Though SPG helps space out vehicles while limiting the travel time increase, it has some drawbacks and limitations. First, it does not maintain comprehensive information about traffic conflicts in the graph structure. Second, it does not consider a non-FCFS passing order that can help achieve a better balance between travel time and safety. Third, it uses a computationally expensive algorithm for spacing out vehicles, which makes it less suitable for citywide real-time traffic optimization.

In this paper, we present a data management framework that is comprehensively improved from SPG by addressing its limitations. Our framework is named **Conflict Zone Graph based Motion Planning (CZMP)**. The proposed framework has three major contributions over SPG. First, it uses an enhanced data structure to better represent

network-level traffic information. Compared to the original data structure, the enhanced version contains more types of information related to traffic conflicts, enabling a more comprehensive view of the road network. Second, CZMP uses a delay planning algorithm to plan delays for vehicles based on a safety-oriented but efficient non-FCFS passing order at conflicting locations. The delay planning algorithm helps distribute the available spaces in the road network and the waiting times at vehicle queues for spacing out vehicles more effectively. Third, CZMP uses a significantly faster motion planning algorithm to generate vehicle trajectories based on the planned delays of vehicles. Different to SPG's motion planning algorithm, CZMP's motion planning algorithm decides the motion in a direct manner without incremental adjustments to the trajectories, achieving significant improvements in computation time, as we show in the results. In contrast, SPG iteratively plans the motions of vehicles and checks the amount of delay achieved by each vehicle. SPG needs to adjust the plan in more iterations if there exist vehicles that can get a further delay. By separating delay planning from trajectory computation, CZMP avoids the iterative process, thus CZMP improves the quality of the results and reduces the computation complexity significantly.

Our framework has three components. The first component is a data structure called **Conflict Zone Graph (CZG)**, which is a time-dependent graph that keeps comprehensive information about *conflict zones* where vehicles would come close to each other for a particular time horizon under consideration. As CZG is time-dependent, it is constructed using real-time network-level spatio-temporal information. The second component is a heuristic algorithm called **Incremental Delay Planning Algorithm**. Using the information in CZG, the algorithm plans delays to modify the time that individual vehicles arrive at conflict zones, which helps space out vehicles. The third component is an algorithm called **Safer Network Motion Plan Generation Algorithm**. Based on the planned delays, the algorithm performs motion planning for the vehicles. The motion plans contain detailed instructions for vehicles to realize the planned delays. Our framework modularizes each important step in the workflow for better usability and extendability. Modularization increases the applicability of the framework for similar applications. The components shown in this paper can be replaced with other application-specific algorithm versions in the future.

We compare the CZMP framework against many existing spatio-temporal data management solutions for traffic optimization in a variety of experimental scenarios created with both synthetic and real data. Our experiments show that CZMP outperforms the existing solutions in achieving a good balance between road safety and travel time. For example, in synthetic data based experiments, CZMP improves the safety level, measured based on the time to collision between vehicles, by up to 68% from a state-of-the-art traffic-efficiency-oriented solution, Intersection Conflict Resolution (ICR) [12]. Interestingly, CZMP achieves a better traffic efficiency than ICR even though CZMP is safety-oriented and ICR is efficiency-oriented. CZMP reduces travel time by up to 60% from ICR. The comprehensive advantages of CZMP over ICR come from two key differences between the two solutions. First, ICR only imposes a minimal safety constraint based on fixed-size inter-vehicular spaces, whereas CZMP can improve safety by enlarging inter-vehicular spaces whenever possible. Second, ICR only considers traffic information at individual intersections when optimizing traffic. Differently, CZMP coordinates traffic at a larger scope as it considers network-level traffic information. This makes CZMP more effective in managing safety while having a positive side effect on traffic efficiency. Our experiments show that CZMP not only outperforms ICR, but also outperforms the prototype safety optimization solution SPG which is developed in our previous work. The advantage of CZMP over SPG is mostly noticeable in computation efficiency. CZMP consumes a significantly shorter computation time compared to

SPG. For example, CZMP can generate travel plan for 1500 vehicles within 1 second while SPG spends 34 seconds to compute for the same number of vehicles. Even in real data based experiments, we achieve similar improvements in terms of traffic safety, traffic efficiency and computational efficiency. The main contributions of our work are summarized as follows.

- We formalize a research problem that is important to the management of traffic data. The problem aims to improve road safety while minimizing the impact on travel time at the network level. The problem is named **Network-Level Safety Optimization (NSO)**.
- We develop a spatio-temporal data management framework, **Conflict Zone Graph based Motion Planning (CZMP)**, which incorporates a novel time-dependent conflict graph and two algorithms to develop motion plans that are safety focused.
- We compare road safety level, travel time and computation costs of CZMP against existing spatio-temporal data management solutions such as ICR and variants of the prototype solution developed in our previous work. Our results show that CZMP achieves a better balance between road safety and travel time compared to all other solutions, in both synthetic data and real data based experiment setups. With increasing levels of connectedness and autonomy in transport, our problem and solutions will be fundamental in optimizing traffic efficiency while considering road safety.

2 Related work

Information Systems for Traffic Management The rapid advancement of information systems has accelerated the development of various Intelligent Transportation Systems (ITSs) [1, 13–15], where information systems are integrated with traffic engineering. With the spatio-temporal data collected from vehicles and traffic infrastructures, ITSs can perform advanced real-time traffic optimization. There are different objectives of optimization such as minimizing the total travel delays of vehicles [5, 16] or maximizing the throughput of the network [17, 18]. The optimizations can be performed at a small scale, such as a particular highway segment [9], an intersection [19] or a traffic corridor [20]. The optimizations can also be performed at larger scales [4, 14, 21, 22]. Large-scale traffic optimizations can give globally better results but they need to apply highly efficient spatio-temporal data management techniques for real-time applications.

In respect to road safety, most of the existing data management solutions for ITSs only impose minimum inter-vehicular spaces to prevent collisions in normal situations, which makes the solutions less robust for preventing accidents in many unpredictable events [23, 24]. There exist solutions that improve mobility while having a positive side effect on road safety [5, 25, 26]. However, those solutions are not designed for optimizing inter-vehicular spacing. In contrast, our proposed framework is focused on improving road safety at the network level by optimizing inter-vehicular spacing. Our framework is also different to routing-based solutions that aim to provide personalized safer routes by navigating vehicles away from risky areas [27–29].

The connectivity and autonomy of CAVs have created more opportunities for ITSs to perform finer traffic optimization operations such as controlling vehicle trajectories and manipulating inter-vehicular spacing [30, 31]. However, many of the existing solutions are not suitable for globally coordinated traffic optimizations due to their high computation complexity. Our proposed spatio-temporal data management framework uses efficient algorithms, which enables real-time network-level road safety optimization in an ITS.

Algorithms for Collective Trajectory Optimization There exists a large body of work that uses well-known techniques such as linear programming, genetic algorithms, dynamic programming or gradient descent algorithms to optimize the trajectories of a group of vehicles collectively for improving mobility, safety, or fuel-efficiency [20, 30–35]. To the best of our knowledge, all the existing solutions in this area can only work well within limited scopes such as individual intersections [33], individual highway segments [31], or individual traffic corridors [20] using a limited set of spatio-temporal data. For example, Wei et al. [35] use a dynamic programming approach to adjust the trajectories of a group of vehicles between two intersections for improving safety and mobility performance. In addition to the algorithms that are dedicated to trajectory optimization, there exist other data management techniques that help improve the quality of trajectories indirectly. For example, Khondaker et al. [9] develop an algorithm to optimize variable speed limit signals for a freeway segment, which can result in better trajectories. Another recent study [36] develops a reinforcement learning-based approach to optimize traffic signals, which can result in safer trajectories. The approach performs optimization using a variety of spatio-temporal data such as traffic volume, queue length, shock wave area, and platoon ratio etc. Different to our proposed framework, none of the aforementioned solutions optimizes trajectories based on the globally optimized inter-vehicular spacing. Our approach produces trajectories that help improve road safety at the network level. In addition, our framework is more suitable for real-time applications compared with many existing solutions, which is due to the fact that our framework can use a novel graph-based data structure and heuristic algorithms to reduce computation costs.

Data Structures for Traffic Optimization Due to the complexity of traffic optimization problems, innovative data structures are used for efficient spatio-temporal data management. For example, Giridhar et al. [2] develop an approach that uses a directed graph to represent spatio-temporal traffic data for scheduling automated traffic. The graph is based on a discrete-space model of a road network, where each lane is divided into one or more cells and each cell corresponds to a vertex in the graph. A similar cell-based representation of road network is used for deadlock prevention of self-driving vehicles [37].

Graph-based data structures have been used for modelling conflicts between vehicles, which is important to our work on road safety. Liu et al. [12] develop a method, which we call Intersection Conflict Resolution (ICR), to optimize the order that vehicles pass intersections. For a specific road intersection, the method models the sequence that vehicles arrive at the intersection using a type of conflict graph. Each vertex in the graph represents a vehicle and each edge connects to a pair of vehicles that will enter the intersection from conflicting approaches. Based on the graph, the method determines the order that conflicting vehicles arrive at the intersection. Lin et al. [38] develop a different conflict graph that uses three types of edges to represent different ways that traffic conflicts can be formed. The graph is used by a scheduling algorithm that resolves the conflicts while preventing deadlocks on the road. The aforementioned graphs only model traffic conflicts at individual road segments or individual intersections. Different to these conflict graphs, our previous work [11] develops a data structure called platooning graph, which is based on a conflict graph that models traffic conflicts in a continuous space. The conflict graph can model conflict zones that span multiple road segments and multiple intersections, which enables a significantly more effective representation of spatio-temporal data that suits network-level traffic management. In this work, we develop an enhanced version of the data structure for more efficient and more effective traffic safety management.

3 Problem definition

In this section, we present the formal problem formulation considering a set of vehicles moving in a road network. We assume that the traffic progresses in discrete time steps with a time step size Δt. In the rest of the section, we introduce the key concepts then define our research problem.

Road Graph The road graph is a directed graph RG=(VR, ER), where VR is a set of vertices and ER is a set of edges. An edge is connected to a pair of vertices. Each vertex $vr \in$ VR represents the end of a traffic lane or a point where two or more traffic lanes connect. Each edge $er \in$ ER represents a traffic lane.

Vehicles Given a set of vehicles that are present at a specific time t, $A(t)$, the state of a vehicle $a_i \subseteq A(t)$ has three elements, $e_i(t)$, $x_i(t)$, and $vl_i(t)$. The element $e_i(t)$ is the edge where a_i is located at the time, $x_i(t)$ is the distance between the start of edge $e_i(t)$ and the position of a_i, and $vl_i(t)$ is the velocity of the vehicle at the time. Between two successive time steps t and $t + \Delta t$, a vehicle a_i can make an **acceleration** action denoted by $ac_i(t)$. The state of a_i at time $t + \Delta t$ depends on its state at time t and the acceleration $ac_i(t)$.

Time To Collision Assuming two vehicles a_i and a_j keep moving without changing their speed, the period between a given time t and the future time that the two vehicles would collide is the **Time To Collision (TTC)** between the two vehicles at time t, denoted as $ttc_{i,j}(t)$.

TTC has been used as a surrogate safety measure for studying the safety impact of autonomous vehicles [39]. We assume that the TTC between any pair of vehicles must be equal to or larger than a **minimum allowed TTC** (TTC_{mn}) to guarantee a minimum safety level. There is also an upper limit of TTC called **TTC threshold** (TTC_{thr}), which is a constant where $TTC_{thr} > TTC_{mn}$. If the TTC between two vehicles is higher than TTC_{thr}, we assume that the risk of a collision would be negligible, hence not a safety concern. The TTC of a specific vehicle a_i at a specific time t is denoted as $ttc_i(t)$. As shown in Eq. (1), $ttc_i(t)$ is either the lowest TTC between vehicle a_i and any other vehicle at that time or the TTC threshold, whichever is lower.

$$ttc_i(t) = min\left(\min_{a_j \in A(t), \forall i \neq j}(ttc_{i,j}(t)), TTC_{thr}\right) \tag{1}$$

Network Motion Plan A motion plan π_i of a vehicle a_i contains a sequence of accelerations at regular time intervals. A Network Motion Plan (NMP) π consists of the motion plans for all the vehicles under the assumption that the TTC between any pair of vehicles would be equal to or larger than the minimum allowed TTC (TTC_{mn}) if the vehicles follow their motion plans.

Distance to TTC Threshold Given a network motion plan π and a TTC threshold TTC_{thr}, we define a road safety metric, **Distance to TTC Threshold** ($DTTC$), using Eq. (2)[1], where $A(t)$

[1] We chose to use a standard L2 loss function-based formulation rather than a simple sum for the equation. As shown later, our solution optimizes traffic based on the minimization of DTTC. During the optimization process, the sum of squares can help even out inter-vehicular spaces, which is useful for achieving universal safety improvements across the whole network. The relationship between the sum of squares and the TTCs of individual vehicles can be shown with the following example. Let a_i and a_j be two vehicles and the TTC threshold be $10s$. In the first case, $ttc_i(t) = 2s$ and $ttc_j(t) = 8s$, which results in $DTTC(\pi) = 68s^2$. In the second case, $ttc_i(t) = 5s$ and $ttc_j(t) = 5s$, which results in $DTTC(\pi) = 50s^2$. As the example shows, a lower DTTC indicates that the TTCs are more uniform, which implies that the road safety is optimized more universally.

is the set of vehicles at a particular time t and $ttc_i(t)$ is the TTC of vehicle a_i defined with Eq. (1). Our previous study also measures road safety based on DTTC [11]. DTTC is derived from Time Integrated TTC [40] that has been widely adopted by the research community. Based on our definition of TTC Eq. (1), if the minimal TTC between vehicle a_i and any other vehicle is equal to or higher than the TTC threshold, the value of $(TTC_{thr} - ttc_i(t))$ is 0 and does not contribute to DTTC. In other words, DTTC is aggregated based on the TTC values that are below the TTC threshold. When these TTC values are higher, the gap between the values and the TTC threshold is lower. Therefore, a lower DTTC corresponds to a better road safety level.

$$DTTC(\pi) = \sum_{t \in T} \sum_{a_i \in A(t)} (TTC_{thr} - ttc_i(t))^2 \tag{2}$$

Fastest Travel Time The fastest travel time of vehicle a_i is the travel time achieved when the vehicle moved at the speed limit at all times. It is denoted as ftt_i.

Desired Maximum Travel Time Given a **desired maximum travel time factor** $\alpha > 1$ and the fastest travel time ftt_i of a vehicle a_i, the desired maximum travel time of the vehicle is $\alpha \times ftt_i$.

Network-Level Safety Optimization (NSO) Problem[2] Given a road graph (RG), a set of vehicles (A), a minimum allowed TTC (TTC_{mn}), a TTC threshold (TTC_{thr}), and a desired maximum travel time factor (α), find the network motion plan (NMP) π, which minimizes the distance to TTC threshold (DTTC) while the travel time tt_i of each vehicle $a_i \in A$ is within the desired maximum travel time, i.e., $tt_i \leq \alpha \times ftt_i, \forall a_i \in A$. The definition can be formulated with Eq. (3).

$$\pi^* = \underset{\pi \in \Pi}{\operatorname{argmin}} \{DTTC(\pi)\} \tag{3}$$

where Π is the set of all the possible NMPs.

Due to the complexity of traffic systems, there is no efficient way to compute the theoretical optimum of DTTC. For this reason, we do not aim to find the exact NMP that achieves the theoretical optimum. Instead, we aim to find a NMP that approximates the hypothetical best case of DTTC, which is 0.

4 Conflict zone graph based motion planning framework

In this section, we detail the Conflict Zone Graph based Motion Planning (CZMP) framework, which we use to compute approximate solutions of the NSO problem defined in Section 3. CZMP can be applied to a transportation system where vehicles are connected to a traffic management system. Our strategy for improving road safety includes two steps. First, we find the common lane segments and intersections shared by the vehicles. These lane segments and intersections are the areas with a risk of vehicle collision. Second, we delay the time that certain vehicles arrive at these areas.

The framework consists of three main components (Fig. 1), which includes a Conflict Zone Graph (CZG), an Incremental Delay Planning Algorithm, and a Safer Network Motion Plan Generation Algorithm. Due to the dynamic nature of traffic, CZMP

[2] NSO problem is a variation of the well-known Multi Agent Path Finding (MAPF) problem [41].

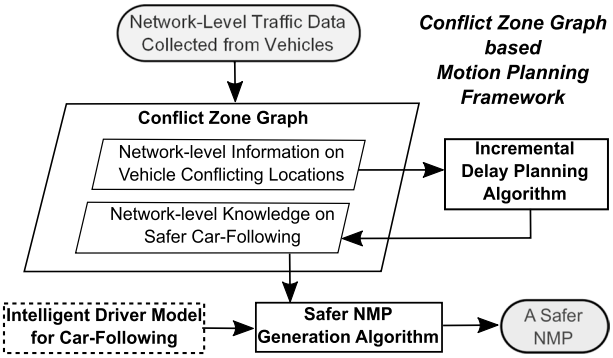


Fig. 1 CZMP framework

re-optimizes traffic at a certain **update interval** because the effectiveness of an optimized motion plan may deteriorate when traffic condition changes in time. An update interval may last one or more time steps. During each update interval, CZMP performs optimization by constructing a new CZG and running the two algorithms.

When constructing a CZG, CZMP finds all the safety-critical areas based on the routes of vehicles moving in the road network. The safety-critical areas are called **conflict zones** in our work. A conflict zone is a part of the road network where collisions may happen in the future [12, 38]. The information of conflict zones is maintained in the CZG. Based on this data structure, the framework uses the Incremental Delay Planning Algorithm to compute the delayed times that some of the vehicles should arrive at the conflict zones. The delays are planned for reducing the risk of collisions while considering various constraints such as the desired maximum travel time of vehicles. The delays change the order that vehicles arrive at conflict zones and enable a better distribution of available spaces and queue waiting times among all the vehicles. As shown in Fig. 1, the CZG is then updated with the delay information. In order to arrive at the conflict zones with the planned delays, vehicles may need to adjust their motion plan, which includes the acceleration of individual vehicles at future time steps. The framework runs the Safer Network Motion Plan (NMP) Generation Algorithm to compute the desired motion that is needed to achieve the delays. The algorithm uses a well-known car-following model, Intelligent Driver Model (IDM) [42], to generate a safe motion plan for each vehicle based on the information from the CZG. The output motion plan is an approximate solution to the optimization problem defined in Section 3.

As the distance that vehicles can travel within an update interval is limited by the speed limit of roads, CZMP only considers the conflict zones within a **look ahead distance (LA)** for each vehicle. We conducted an experiment to evaluate the impact of LA on safety, travel time and computation cost (Section 5.3.3). Our result shows that considering conflict zones beyond certain distance would not improve safety and travel time significantly but would increase computation cost.

The framework modularizes each important step of the workflow to improve usability in future applications. For example, it is possible to replace the incremental delay planning algorithm easily with a different algorithm version without changing other components. Similarly, we could use a different NMP generation algorithm based on other car-following or platooning strategies without changing other components.

In the rest of the section, we describe the three main components of the CZMP framework and how they work together to optimize traffic during an update interval. Table 1 highlights the key notations which are frequently used throughout the paper.

Table 1 Description of key notations

Context	Notation	Description
Problem Definition	Δt	Time step size
	t	A particular time
	RG	A road graph
	$vr \in VR$	An end point of a traffic lane (vr) and the set (VR)
	$er \in ER$	A traffic lane (er) and the set of traffic lanes (ER)
	A	The set of vehicles
	a_i	A vehicle with the index i
	$A(t)$	The set of vehicles on the road graph at time t
	$e_i(t)$	The edge where a_i is located at time t
	$x_i(t)$	The distance travelled by a_i on edge $e_i(t)$ at time t
	$vl_i(t)$	The velocity of vehicle a_i at time t
	$ac_i(t)$	The acceleration of vehicle a_i between time t and $t + \Delta t$
	$ttc_{i,j}(t)$	The TTC between vehicles a_i and a_j at time t
	TTC_{mn}	The minimum allowed TTC (the lower limit of TTC values)
	TTC_{thr}	The TTC threshold (the upper limit of TTC values)
	$ttc_i(t)$	The TTC of vehicle a_i at time t (Eq. 1)
	T	The time taken to complete trips of all the vehicles
	tt_i	The travel time of vehicle a_i from its source to destination
	ftt_i	The fastest travel time of vehicle a_i
	α	The desired maximum travel time factor
	π_i	A motion plan of vehicle a_i
	π	A network motion plan (NMP) satisfying minimum allowed TTC constraint between all vehicles
		TTC constraint between all vehicles
	Π	Set of all possible NMPs
	π^*	The NMP with minimum DTTC
CZMP Framework	U	The update interval
	LA	The look-ahead distance
	CZG	A conflict zone graph
	t_u	A particular update time
	$A(t_u)$	The set of vehicle on the road graph at the update time t_u
	C	The set of conflicts between the vehicles in the set $A(t_u)$
	d_i	The planned delay of vehicle a_i
	$c_{i,j}$	A conflict zone between the two vehicles a_i and a_j
	pat_i	The projected arrival time of vehicle a_i to the conflict zone $c_{i,j}$
	D	A set of delay increments
	Δd_k	A delay increment in the set D
	Δs	The time headway increment
IDM Model	AC_{mn}	The desired deceleration of vehicle a_i
	AC_{mx}	The maximum acceleration of vehicle a_i
	vl_0	The desired velocity of vehicle a_i
	V_{mx}	The maximum velocity allowed for vehicle a_i
	δ	The free acceleration exponent
	$\Delta v(t)$	The relative velocity between a_i and the front vehicle a_j at time t
	$s(t)$	The distance between a_i and the front vehicle a_j at time t

Table 1 (continued)

Context	Notation	Description
	s_0	The minimum distance between a_i and the front vehicle a_j
	TH	The time headway to keep between a_i and the front vehicle a_j

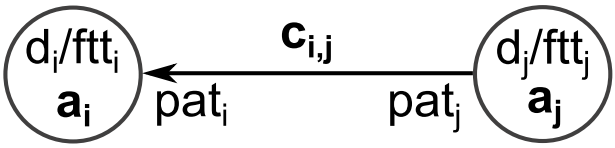


Fig. 2 Representation of a conflict zone in CZG

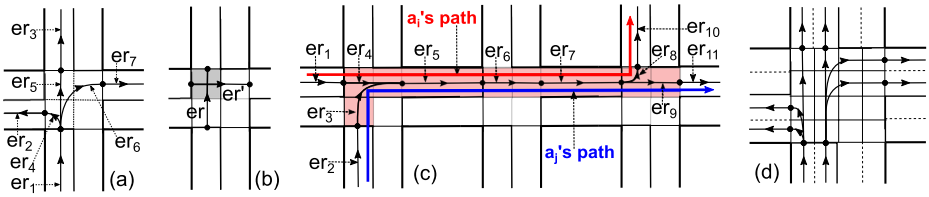


Fig. 3 Examples and details of conflict zones

4.1 Conflict zone graph (CZG)

A $CZG = (A(t_u), C)$ is a time-dependent graph where $A(t_u)$ is a set of vertices and C is a set of edges. CZMP constructs a new CZG at the start of an update interval t_u based on the traffic data available at the time. A vertex in CZG represents a vehicle $a_i \in A(t_u)$ that is present when the graph is constructed. A vertex has two attributes. The first attribute is the **fastest travel time** (ftt_i), which is the shortest time that the vehicle a_i can travel the look ahead distance or the remaining part of its route, whichever is shorter, assuming the vehicle can travel at the speed limit. The second attribute is the **planned delay** (d_i), which is the delay that should be applied to the vehicle. The d_i value is set to zero when constructing the CZG. An edge represents a **conflict zone** ($c_{i,j}$), where i and j represent two vehicles, a_i and a_j, respectively (Fig. 2). A conflict zone is a common/shared part of the routes of the two vehicles. The size of conflict zones can vary in different scenarios. For example, a conflict zone maybe a single intersection where the routes intersect. It may also start at an intersection, where the routes of the vehicles overlap (meet, followed by a set of continuous traffic lanes shared by the vehicles thereafter).

We identify conflict zones based on the following two rules,

- **Rule 1:** If two vehicles a_i, a_j use the same edge $er \in ER$ in their path then er is a part of a conflict zone.
- **Rule 2:** If two vehicles a_i, a_j use two crossing edges $er, er' \in ER$ at an intersection in their paths then er and er' are also a part of a conflict zone. Figure 3a shows a four-legged intersection connecting four two-way single-lane road segments. It shows the edges and vertices of the road graph and three different paths that start from edge er_1. We consider a virtual grid inside the intersecting region based on the number of lanes

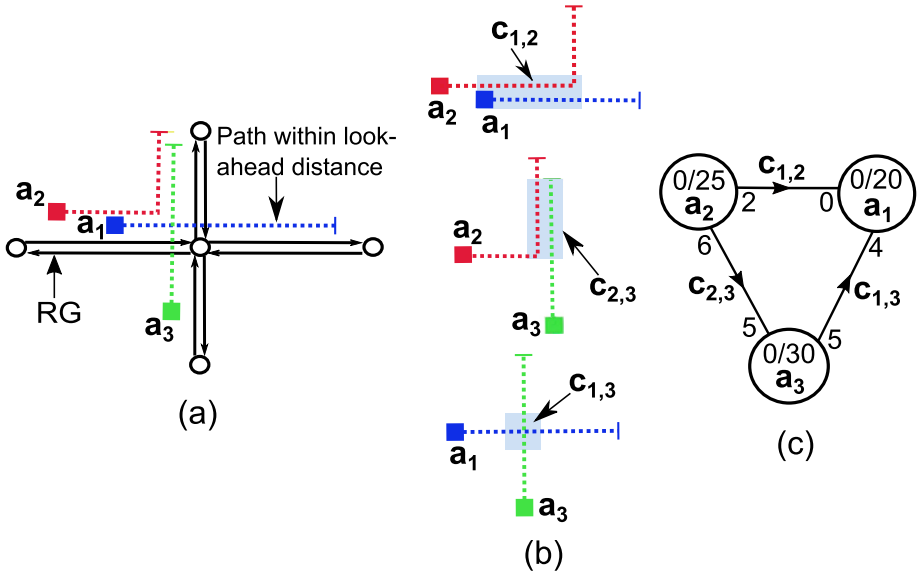


Fig. 4 A traffic scenario, the corresponding conflict zones and the corresponding CZG

from intersecting road segments to determine the conflicting locations inside the intersection. For example, if the two intersecting roads have one lane in each direction, there is a 2×2 virtual grid at the intersection as shown in Fig. 3a. Each edge within the virtual grid occupies a certain sequence of cells. If two different edges $er, er' \in ER$ at an intersection has a shared cell as shaded in Fig. 3b, they are regarded as 'crossing edges'.

Figure 3c shows a road network with three consecutive intersections, where vehicle a_i's path goes through the edges $[er_1, er_4, er_5, er_6, er_7, er_8, er_{10}]$, and vehicle a_j's path goes through the edges $[er_2, er_3, er_5, er_6, er_7, er_9, er_{11}]$. For simplicity, we only show the road edges used by those two paths. The conflict zone $c_{i,j}$ between the two vehicles a_i, a_j is shaded in red and it includes the edges $\{er_3, er_4, er_5, er_6, er_7, er_8, er_9\}$. The er_5, er_6, er_7 are common to both vehicle paths (Rule 1) and edge pairs $(er_3, er_4), (er_8, er_9)$ are crossing edges (Rule 2). Figure 3d shows how we extend the road graph and virtual grid for a double lane intersection scenario. Similarly, we could extend this formulation into other types of intersections appropriately.

When constructing the CZG, we only consider the conflict zones within the look-ahead distance from the current position of vehicles. The direction of an edge in CZG shows the car-following order, i.e., the edge points to the predecessor vehicle that enters the conflict zone first. An edge also maintains the **projected arrival time** (*pat*) for each of the vehicles, pat_i and pat_j. Initially, the value of *pat* is the fastest travel time that a vehicle can arrive at the conflict zone, assuming the vehicle always travels at the speed limit. The value will be increased during optimization when a delay is added to the vehicle.

A simple example of CZG is shown in Fig. 4. This example shows a scenario where three vehicles a_1, a_2 and a_3 have conflicts between their paths (Fig. 4a). For simplicity of the example, we assume that each road segment has one traffic lane in each direction. There are three conflict zones, $c_{1,2}$, $c_{2,3}$ and $c_{1,3}$ (Fig. 4b). Figure 4c shows the CZG based on the three conflict zones. The graph contains three vertices (vehicles) and three edges (conflict zones). Each edge points to the vehicle that arrives at the zone first. For

example, for conflict zone $c_{1,2}$, pat_1 is 0 and pat_2 is 2, which means vehicle a_1 arrives the conflict zone first. Therefore, the corresponding edge points to a_1.

Computing CZG Given a set of vehicles at a particular update time t_u ($A(t_u)$), a look-ahead distance (LA) and a road network (RG), the CZMP framework uses Algorithm 1 to compute CZG.

Input: $A(t_u), LA, RG$
Output: CZG
1 **for** $a_i \in A(t_u)$ **do**
2 $a_i \leftarrow initialize()$
3 **for** $er \in$ *edges on* a_i*'s path within* LA *from* a_i **do**
4 $A_{er} \leftarrow A_{er} \cup \{a_i\}$
  $//A_{er}$ are the vehicles whose future path within LA covers er
5 $C \leftarrow \phi$
6 $Q \leftarrow \{A(t_u)\}$
7 **while** $Q \neq \phi$ **do**
8 $a_i \leftarrow pop(Q)$
9 **for** $er_i \in$ *edges on* a_i*'s path within* LA *from* a_i **do**
10 **for** $er_j \in$ *edges conflicting to* er_i *due to Rule 1 and Rule 2* **do**
11 **for** $a_j \in A_{er_j}$ **do**
12 **if** $a_i \neq a_j$ **then**
13 $C \leftarrow update(C, a_i, er_i, a_j, er_j)$
14 $A_{er_i} \leftarrow A_{er_i} - \{a_i\}$
15 **return** $CZG = (A(t_u), C)$

For each vehicle in the input set of vehicles, the algorithm builds a corresponding vertex in the CZG (Line 1-2). For each road lane within the look ahead distance on the path of a vehicle a_i, the algorithm finds all the conflicting road lanes and the vehicles that use those lanes. An edge is built between a_i and each of the found vehicles (Line 13).

Time complexity The time complexity of Algorithm 1 is $\mathcal{O}(|A(t_u)|^2 |p_{mx}||c_{mx}|)$, where $|A(t_u)|$ is the number of available vehicles at update time t_u, $|p_{mx}|$ is the maximum number of edges on a path, $|c_{mx}|$ is the maximum number of crossing edges for an edge in the road graph.

4.2 Incremental delay planning algorithm

After building the CZG data structure, our framework uses Algorithm 2 to plan the delays of the vehicles' arrival at conflict zones. Adding delay to the arrival time can change the order of vehicle arrivals and help enlarge the inter-vehicular space between two vehicles, resulting in an improvement of road safety with a reduced conflict between the vehicles. The algorithm runs in iterations, where each iteration increases the delay by a value from a predefined sequence of delay increments. There are certain constraints that need to be obeyed when delaying vehicles. For example, the increased travel time caused by the delay

cannot exceed the desired maximum travel time. If any of the constraints could not be satisfied for a vehicle, the delay will not be applied to the vehicle. Otherwise, the algorithm updates the delay value at the vehicle's corresponding vertex in CZG. The delay information in the updated CZG will be used for motion planning as detailed in Section 4.3.

Input: $CZG = (A(t_u), C)$ - Conflict Zone Graph
Input: TTC_{thr} - TTC threshold
Input: $D = [\Delta d_1, \Delta d_2, .., \Delta d_k, ..]$ - A set of delay increments
Output: The updated CZG with planned delays

```
1  for Δd_k ∈ D do
2      while (true) do
3          A_C ← φ
4          for a_i ∈ A(t_u) do
5              if Check desired maximum travel time constraint then
6                  risk_change_i ←
                     Compute change of safety risk for a_i with Δd_k
7                  dep_i ← Find the dependencies of a_i
8                  valid ← Check FIFO constraint
9                  if ((valid)&(risk_change_i < 0)) then
10                     A_C ← A_C ∪ {a_i}
11         if (A_C == φ) then
12             break
13         while (A_C ≠ φ) do
14             a_i ←
                 Remove vehicle with the most negative risk_change_i from A_C
15             d_i ← d_i + Δd_k // Update corresponding vertex in CZG
16             for any edge connected to a_i do
17                 pat_i ← pat_i + Δd_k // Update projected arrival time
18             A_C ← A_C − dep_i
19 for any edge in CZG do
20     if the corresponding conflict zone starts at an intersection then
21         update direction of the edge
22 return CZG
```

Algorithm 2 checks four constraints when planning delays and filters out unsuitable vehicles for adding delays (Line 5-10). This procedure is detailed as follows.

1. **Desired maximum travel time:** As shown in the research problem definition (Section 3), the travel time of a vehicle must be within an acceptable range when additional delay is applied. Specifically, for any vehicle a_i, given the fastest travel time ftt_i in CZG, the current planned delay d_i, the additional delay Δd_k and the desired maximum

travel time factor α, if $ftt_i + d_i + \Delta d_k > \alpha \times ftt_i$, the increased travel time is beyond the desired maximum travel time and thus the algorithm would not apply the delay to a_i. The algorithm checks this constraint in Line 5.

2. **Change of safety risk:** The algorithm checks whether a delay can help reduce road safety risk. If not, the delay is not beneficial and should not be applied. The algorithm computes the change of safety risk for a vehicle a_i (Line 6). The computation is based on Eq. (4). $E(a_i)$ is the edges (conflict zones) of vehicle a_i.

$$risk\_change_i = \sum_{c_{i,j} \in E(a_i)} (h'_{c_{i,j}} - h_{c_{i,j}}) \tag{4}$$

In Eq. (4), $h'_{c_{i,j}}$ is the safety risk for the conflict zone between vehicle a_i and vehicle a_j when the additional delay (Δd_k) is added to a_i, and $h_{c_{i,j}}$ is the safety risk without considering the additional delay. If $risk\_change_i$ is negative, the delay is deemed as beneficial as the safety risk is reduced. The safety risk $h_{c_{i,j}}$ can be computed using (Eq. 5), where TTC_{thr} is the TTC threshold, pat_i is the projected time that a_i arrives at the conflict zone, and pat_j is the projected time that a_j arrives at the conflict zone. We should note that pat_i should be replaced with $pat_i + \Delta d_k$ when computing the risk with the added delay ($h'_{c_{i,j}}$).

$$h_{c_{i,j}} = \begin{cases} (TTC_{thr} - |pat_i - pat_j|)^2 & \text{if } |pat_i - pat_j| \leq TTC_{thr} \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

3. **Dependencies of vehicle:** In the event that both vehicles involved in a conflict zone get the same amount of delay, there would be no change to the safety risk associated with the conflict zone, resulting in unnecessary delays to both vehicles. The algorithm avoids this situation by imposing a dependency constraint. A vehicle a_j has a dependency on a vehicle a_i if a delay of a_i leads to a lower safety risk at the conflict zone $c_{i,j}$ that involves the two vehicles. Under this constraint, if both vehicles have a dependency on each other, only one of the vehicles can get a delay. The algorithm finds the set of dependencies of a given vehicle in Line 7.
4. **First-In-First-Out (FIFO) constraint:** If a conflict zone starts with a lane, a vehicle that enters the conflict zone first must also be the vehicle that leaves the zone first. The constraint does not apply if the conflict zone starts with a road intersection because a vehicle may need to give way to another vehicle from a conflicting direction according to road rules, even when the former vehicle arrives at the intersection first. The FIFO constraint can be checked as follows. Given a conflict zone $c_{i,j}$ between vehicles a_i and a_j, assuming the conflict zone starts with a lane, if a_j enters the zone first, the projected arrival time of a_i (pat_i) must be greater than the projected arrival time of a_j (pat_j). If a delay of Δd_k is added to the predecessor a_j and $pat_i \leq pat_j + \Delta d_k$, the FIFO constraint is violated. The algorithm checks the constraint in Line 8.

After checking the constraints, the vehicles that can get an additional delay of Δd_k are included in a set A_C. Then the algorithm updates CZG using the additional delay Δd_k. For updating CZG, the algorithm starts from the vehicle that would get the most significant benefit by adding a delay (Line 14). Once the delay is added to the vehicle's corresponding vertex and edges, the dependencies of the vehicle are removed from the set A_C (Line 18). This is because delaying a vehicle and its dependencies at the same time would result in no benefit for any of the vehicles.

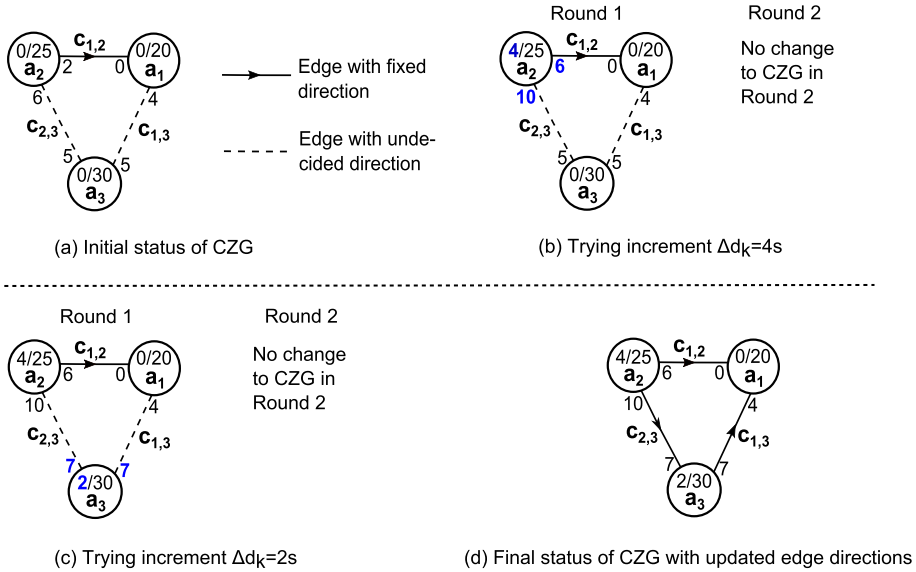


Fig. 5 Example of incremental delay planning algorithm

After planning delays for all vehicles, the algorithm updates the direction of certain edges in CZG (Line 19-21). The direction of an edge shows the order that a pair of vehicles arrive at the corresponding conflict zone. The order of arrival is important for making motion plans of vehicles, which will be detailed in Section 4.3. When delays are applied to vehicles, the order of arrival can change if the conflict zone starts at a road intersection because the vehicles have the opportunity to slow down and give way to other vehicles at intersections. Therefore, the algorithm updates the order of arrival for the conflict zones that start at road intersections. For such a conflict zone, the vehicle with the lower projected arrival time will be set as the first vehicle to reach the intersection. In the rare case that two vehicles have the same arrival time, the vehicle with the lower index in the database will be set as the first vehicle to reach the zone.

Algorithm 2 Example Based on the scenario in Figs. 4 and 5a shows the initial status of a CZG when the algorithm starts. In this example, we assume that the desired maximum travel times of vehicle a_1, a_2 and a_3 are 24s, 30s and 36s, respectively. The TTC threshold TTC_{thr} is 10s and the set of candidate delay increments is $D = [4s, 2s]$. The edge direction for conflict zone $c_{1,2}$ is fixed from the beginning because the zone starts on a road segment, which means the order that the two vehicles arrive at the zone is deterministic. Differently, the direction of the other two conflict zones will be decided as the algorithm runs. We show the direction of the remaining conflict zones in Fig. 5d. Figure 5b and c show the state at each iteration of the algorithm. The algorithm may need to run multiple rounds for a particular delay increment. We start by taking the first delay increment, $4s$, from D (line 1) and look for vehicles that are benefited from a $4s$ delay.

1. $\Delta d_k = 4s$ **Round 1:** We check the potential impact of a $4s$ delay on each of the three vehicles. When adding the delay to a_1, we see that the delay increment violates the

FIFO constraint at the conflict zone $c_{1,2}$ (line 8). Therefore, a_1 should not get the delay in this round. We find that both a_2 and a_3 would be benefited from the delay as they would get negative risk change values (line 6). As adding the delay to a_2 and a_3 would not violate other constraints, we consider them as the candidate vehicles that may actually get the delay (line 9, 10). Then, we apply the delay to a_2 and update the CZG because adding the delay to a_2 would result in a larger reduction of safety risk compared to adding a delay to a_3 (line 14-17). We update the relevant numbers for a_2 in the CZG. The updated numbers are coloured in blue in Fig. 5b-Round 1. We then skip applying the delay to a_3 because a_3 is a dependent of a_2, which means we should not apply the delay to both vehicles at the same time (line 18). Since there are vehicles that are benefited from a delay in this round, we start a second round to see whether a further $4s$ delay would help.

2. $\Delta d_k = 4s$ **Round 2:** In the second round, adding a further delay to vehicle a_2 violates the desired maximum travel time constraint (line 5). Adding the further delay to both a_1 and a_3 cannot lower their safety risk as their risk changes would be non-negative with the delay. Therefore, we cannot find any candidate vehicle that would be benefited from the further $4s$ delay. At this point, we are done with the $4s$ delay increment (line 11, 12). There is no change to the CZG in this round. After that, we start to try a new delay increment, which is $2s$ in D (line 1).
3. $\Delta d_k = 2s$ **Round 1:** In this round, adding a $2s$ delay to a_2 violates the desired maximum travel time constraint. Adding the delay to a_1 gives a non-negative risk change value, which means the vehicle cannot be benefited from the delay. We find that vehicle a_3's risk change value would be negative with the delay. We then apply the $2s$ delay to a_3 and update relevant numbers in the CZG (Fig. 5c-Round 1). Since there is a vehicle that is benefited from the $2s$ delay in this round, we start a new round to see whether adding a further $2s$ delay would help.
4. $\Delta d_k = 2s$ **Round 2:** In this round, adding a $2s$ delay to vehicle a_2 violates the desired maximum travel time constraint. Vehicle a_1 and a_3 are not benefited from the further delay as they would give non-negative risk change values. Therefore, there is no candidate vehicle for applying the delay (line 11, 12). The CZG is not changed in this round. The algorithm terminates as we have finished all the delay increments in D.

Finally, we decide the direction of each edge based on the final projected arrival times. Figure 5d shows the final direction of each edge.

Time complexity The time complexity of Algorithm 2 is $\mathcal{O}(q|D|(|A(t_u)| + |C|))$, where q is the maximum number of delay increments for a vehicle with a particular delay increment Δd_k, $|D|$ is the size of increment sequence D, $|A(t_u)|$ is the number of available vehicles at current update time t_u and $|C|$ is the number of conflict zones between the vehicles.

4.3 Safer network motion plan generation algorithm

To achieve the planned delays, vehicles need to adjust their motion in real time so that they can arrive at conflict zones with the planned delays. Our CZMP framework generates a network motion plan that includes the acceleration of all individual vehicles for each time step during an update interval. The motion plans can be distributed to

connected vehicles, which can adjust their motion according to the plan. The network motion plan is generated with Algorithm 3.

The algorithm is based on a car-following model, Intelligent Driver Model (IDM) [42]. Given the state of a front vehicle a_j and a back vehicle a_i, the IDM model computes the desired acceleration of the back vehicle $ac_i(t)$ at a particular time t using Eqs. (6) and (7).

$$ac_i(t) = AC_{mx}\left[1 - \left(\frac{vl_i(t)}{vl_0}\right)^{\delta} - \left(\frac{s^*}{s(t)}\right)^2\right] \tag{6}$$

$$s^* = s_0 + vl_i(t)TH + \frac{vl_i(t)\Delta v(t)}{2\sqrt{AC_{mx}AC_{mn}}} \tag{7}$$

In Eqs. (6) and (7), AC_{mx} is the maximum acceleration of vehicle a_i, $vl_i(t)$ is the velocity of vehicle a_i at time t, vl_0 is the desired speed (e.g. the maximum velocity allowed V_{mx}), δ is the free acceleration exponent, $\Delta v(t)$ is the relative velocity between a_i and a_j at time t, $s(t)$ is the bumper to bumper distance between a_i and a_j at time t, s^* is the effective minimum gap, s_0 is the minimum bumper to bumper distance between vehicles, TH is the desired time headway for safety, and AC_{mn} is the desired deceleration (i.e. the desired minimum acceleration). A key parameter of the model is **time headway (TH)**, which is the time that the back vehicle takes to reach the current position of the front vehicle. When TH is increased, the space between the two vehicles increases, leading to a delay of the back vehicle and a reduction of traffic conflicts.

Algorithm 3 runs in two parts for each time step. In the first part, it finds the closest front vehicle (predecessor) of a_i using CZG (Line 3-6). As mentioned earlier, a CZG edge represents a conflict zone that involves two vehicles. The direction of the edge points to the vehicle that arrives the conflict zone first. Therefore, an outward edge that starts from a_i points to a vehicle in front of a_i. While iterating over all the outward edges of a_i, the algorithm finds the front vehicle that causes the lowest acceleration of a_i based on the IDM model. This front vehicle has the most significant impact on a_i's safety among all the front vehicles. For example, in the CZG in Fig. 5d, vehicle a_1 does not have any vehicle in front and it could take the maximum acceleration AC_{mx}. However, a_2 has vehicles a_1 and a_3 in front as shown in the CZG. As the CZG in Fig. 5 is based on the traffic scenario shown in Fig. 4, we consider the following two cases when determining the predecessor of a_2.

- **Regarding front vehicle** a_1: Vehicle a_2 should follow vehicle a_1, and let a_1 cross the intersection before start turning at the intersection.
- **Regarding front vehicle** a_3: Vehicle a_2 should stop before the intersection and let a_3 cross the intersection before start turning at the intersection.

We compute a_2's ideal acceleration in each case using the IDM model. Based on the two acceleration values, we select the front vehicle which leads to the lowest acceleration as the predecessor of a_2. The predecessor has the most significant impact on a_2's safety. Similarly, we can find predecessors in other cases as well.

Input: $U = [t_u, t_{u+1}]$, $CZG = (A(t_u), C)$ with planned delays
Output: A safer network motion plan for vehicles
1 **while** $t \in U$ **do**
2 **for** $a_i \in A(t_u)$ **do**
3 $prd(a_i) \leftarrow \phi$
4 **for** $c_{i,j} \in$ *Outward Edges of* a_i **do**
5 **if** a_j *slows down* a_i *more than* $prd(a_i)$ *at time* t **then**
6 $prd(a_i) \leftarrow a_j$
7 $THs \leftarrow GetTHs(TTC_{mn}, TTC_{thr}, \Delta s)$
8 $ac_i(t) \leftarrow GetBestAcc(a_i, prd(a_i), d_i, THs)$
9 update motion plan with $ac_i(t)$
10 **return** The safer network motion plan obtained

In the second part, the algorithm computes the best acceleration value for a_i, based on the state of a_i and the front vehicle found in the first part, using the IDM model (Line 7-9). To do this, the algorithm computes a set of candidate acceleration values while increasing the TH value in the IDM model. Increasing TH leads to an increase of inter-vehicular space, thus an increase of the delay for vehicle a_i. The TH values are varied between TTC_{mn} and TTC_{thr} with an increment of Δs between adjacent values. With each of candidate acceleration values, the algorithm predicts the next state of a_i, i.e., the position and velocity of a_i at the next time step. Different acceleration values would lead to different delays, which may or may not be close to the planned delay for a_i, i.e., the value d_i stored in the CZG. Among all the candidate acceleration values, the value that will be assigned to vehicle a_i is the one that would result in a delay closest to d_i.

Time complexity The time complexity of Algorithm 3 is $\mathcal{O}(|U||A(t_u)|(|C| + \frac{TTC_{thr} - TTC_{mn}}{\Delta s}))$, where $|U|$ is the number of time steps in the update interval, $|A(t_u)|$ is the number of available vehicles at time t_u, $|C|$ is the maximum number of outward edges connected to a vertex in CZG, and $\frac{TTC_{thr} - TTC_{mn}}{\Delta s}$ is the number of TH values to be explored.

5 Experimental methodology

To evaluate the performance of the proposed CZMP framework, we consider two experiment setups, Synthetic Data based Experiment Setup (SDE-Setup) and Real-world Data based Experiment Setup (RDE-Setup). First we perform comparative tests using both setups to evaluate the impact of several traffic parameters on CZMP and a set of baselines. Then, we perform parameter sensitivity tests using SDE-Setup to evaluate the impact of two parameters that are used by CZMP and a variant of CZMP.

Our experiments are conducted with SMARTS[3] [43], a microscopic traffic simulator. We implement the CZMP framework and all the baseline approaches in Java 8. All the

[3] https://projects.eng.unimelb.edu.au/smarts/.

experiments are conducted on a computer equipped with an Intel(R) Core(TM) i7-8550U 1.80 GHz processor, 16 GB RAM and 64 bit Windows 10 Education operating system.

5.1 Synthetic data based experiment setup (SDE-Setup)

5.1.1 Baselines

ICR Intersection Conflict Resolution (ICR) is a state-of-the-art traffic-efficiency-oriented intersection scheduling method [12]. ICR uses an intersection-level conflict graph to model traffic condition around intersections such as the waiting time of vehicles. Based on the graph, ICR determines the order that vehicles pass through the intersections for improving traffic efficiency. In addition, ICR performs motion planning but it uses a vastly different approach compared to CZMP. To make a fair comparison between ICR and our framework, we replace the motion planning component of ICR with the motion planning model of CZMP.

ATSC Actuated Traffic Signal Control (ATSC) is a classical traffic management solution that extends the active color phase if more traffic is observed in that phase [44].

FPG The Fastest Platooning Graph (FPG) algorithm is proposed in our previous work [11]. FPG is more focused on improving travel time rather than achieving a balance between travel time and road safety. FPG and CZMP use the same parameter settings where possible. We derive the following three baselines from FPG.

- **FPG-1:** This baseline uses the TTC_{mn} as the spacing value in FPG (the time headway TH in the IDM model) such that all vehicles maintain the minimum space from other vehicles.
- **FPG-2:** This baseline uses $2 \times TTC_{mn}$ as the spacing value in FPG such that all vehicles naively improve safety by doubling inter-vehicular spaces.
- **FPG-L:** This baseline uses a variable spacing value based on the vehicles' position in a lane for more flexible management of road safety. At the start point of a lane, the spacing value is TTC_{mn}. The value gradually increases up to TTC_{thr} at the middle of the lane and then gradually decreases to TTC_{mn} at the end of the lane. Inside intersections, the spacing value is set to TTC_{mn}.

SPG The Safest Platooning Graph (SPG) is also proposed in our previous work [11]. SPG is similar to CZMP as it aims to achieve a balance between road safety and travel time. However, SPG is computationally expensive. SPG and CZMP use the same parameter settings where possible.

CZMP-U CZMP-U is similar to CZMP except that it only uses a fixed delay increment value, $2s$. That is, CZMP-U does not allow different delay increment values as CZMP does.

5.1.2 Evaluation measures

Average Travel Time Ratio (*ATTR*) As shown in our problem definition (Section 3), an ideal safety management solution would balance safety and travel time. We use the *ATTR*

Table 2 Parameter settings

Parameter	Default value	Value range
Traffic volume	1000	{250, 500, 750, 1000, 1250, 1500}
Road graph	RG-3	{RG-1, RG-2, RG-3, RG-4}
Traffic distribution	UTD	{UTD, GTD}
Look-ahead distance (LA)	900m	{300m, 600m, 900m, 1200m, 1500m, 1800m, 2100m}
Desired maximum travel time factor (α)	1.5	{ 1.0, 1.25, 1.5, 1.75, 2.0, 2.25 , 2.5}

to evaluate the solutions from the traffic efficiency perspective. For a given vehicle a_i, $\frac{tt_i}{ftt_i}$ is the ratio between the actual travel time (tt_i) and the hypothetical shortest travel time (ftt_i), which can be achieved when the vehicle travels at the road speed limit during its entire trip. $ATTR$ is computed as $\frac{1}{n}\sum_{a_i \in A}\frac{tt_i}{ftt_i}$, where n is the number of vehicles. We evaluate traffic efficiency using ATTR rather than a simple summation of travel times because the later form is less suitable to reflect the travel time delays experienced by individual vehicles. The hypothetical optimum value of ATTR is 1, which is also the minimum value that can be achieved. A lower ATTR indicates that the actual travel times are closer to the hypothetical shortest travel times, which represents a better traffic efficiency.

Average Distance to TTC Threshold (*ADTTC*) We use $ADTTC$ to evaluate the solutions from the safety perspective. The value is calculated as $\frac{1}{n}DTTC$, where $DTTC$ is the road safety metric in our problem definition (Section 3) and n is the number of vehicles.

Update Interval Utilization (*UIU*) We use UIU to measure computation costs. UIU is defined as $\frac{\text{Computation Time}}{\text{Update Interval}}$. If the UIU of a solution is higher than 1, the solution is not suitable for real-time optimization as the computation cannot be completed within the update interval. Conversely, a solution is readily deployable in terms of computation efficiency if its UIU is below 1.

5.1.3 Parameter settings

Based on a range of experiment parameters, we evaluate the performance of all the solutions under different combinations of parameter settings. When varying the value of one parameter, all other parameters are set to their default values. The value ranges of parameters are shown in Table 2.

For a particular parameter combination, we run one simulation using each of the solutions. At the start of the simulation, a set of vehicle routes is loaded into the simulator. The simulation runs until all vehicles finish their journeys. The TTC values at each time step and the travel times of vehicles are computed and recorded. At the end of the simulation, we measure the performance of the solution based on the records.

To make a fair comparison between the solutions, all the solutions are tested with the same set of vehicle routes for a specific combination of parameter settings. The set of routes is generated based on the values of three parameters: road network, traffic volume and traffic distribution. We run a one-hour simulation using the **FPG-1** baseline to generate

the set of routes. The traffic volume is kept constant during this simulation, which means a new vehicle is put into the road network when a vehicle reaches its destination.

More details of the experiment parameters are shown below.

Road Network We use four synthetic road networks in our experimental simulations. The area of each of the networks is $3.75km \times 3.75km$. In the simulations with the double-lane road networks, a vehicle does not switch lanes during its trip. This constraint helps eliminate the random effects of lane-changing on the performance of the solutions.

- **RG-1** - Single Lane, 10×10 grid with 100 intersections
- **RG-2** - Single Lane, 15×15 grid with 225 intersections
- **RG-3** - Double Lane, 10×10 grid with 100 intersections
- **RG-4** - Double Lane, 15×15 grid with 225 intersections

Traffic Distribution The origin and destination of vehicle routes are randomly picked using two traffic distributions, **Uniform Traffic Distribution (UTD)** and **Gaussian Traffic Distribution (GTD)**. GTD can lead to a higher traffic density in the centre of the network compared to UTD. Given a pair of origin and destination, we use *Dijkstra's Algorithm* to compute a vehicle route.

Other Parameters In all our experiments, the time step (Δt) is 1s, the minimum allowed TTC (TTC_{mn}) is $2s$, the TTC threshold (TTC_{thr}) is $10s$, the maximum velocity allowed (V_{mx}) is $72kmh$, the delay increment sequence (D) in Algorithm 2 is [$8s$, $4s$, $2s$], the increment of IDM's TH parameter (Δs) in Algorithm 3 is $1s$, and the update interval (U) is $10s$. In IDM model, $vl_0 = V_{mx}$, $s_0 = 2m$, $AC_{mx} = 1.4ms^{-2}$, $B = -2.0ms^{-2}$ and $\delta = 4$.

5.2 Real data based experiment setup (RDE-Setup)

In RDE-Setup, we use the same baselines and evaluation measures that are used in SDE-Setup. To set up the realistic road network and the traffic distribution, we consider two real-world taxi datasets, New York taxi dataset[4] and Beijing (T-Drive) taxi dataset[5] [45] along with the real maps available from Open Street Maps[6].

New York Traffic Scenarios Based on the area covered by New York taxi dataset, we consider two map areas approximately $3km \times 3km$ in size as shown in Fig. 6a and b. For each map, we extract the taxi trips in the map area within a randomly selected hour of a randomly selected day (For example, we consider 7 am - 8 am on 2015/04/11 for New York Midtown map). The two areas have different numbers of taxi trips, as shown in the real data. Then we generate background traffic (non-taxi vehicles) using the uniform traffic distribution (UTD) generation technique that we used in SDE-Setup. We consider three background traffic levels (3600, 5400 and 7200 vehicles/hour) for both maps considering the capacities of the road networks. The selected background traffic levels avoid potential gridlock scenarios for all the baselines as

[4] https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[5] https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/
[6] https://www.openstreetmap.org/

Fig. 6 New York and Beijing road networks used in RDE-Setup

such scenarios could not provide a comparative result. Figure 6a shows the Central Business District (CBD) in New York Midtown area. There were nearly 3600 taxi trips within the hour we consider. For this map when the background traffic level is 3600 vehicles/hour scenario, we simulate 3600 taxi trips with 3600 non-taxi vehicles (total of 7200 vehicles) within the one-hour period. Figure 6b shows the New York Manhattan Bridge area, which consists of nearly 2400 taxi trips within the period.

Beijing Traffic Scenarios Based on the area covered by Beijing taxi dataset (T-Drive), we consider two map areas approximately $3km \times 3km$ in size as shown in Fig. 6c and d. For each map, we extract the taxi trips in the map area within a randomly selected hour of a randomly selected day. Then we generate background traffic considering three background traffic levels (5400, 7200 and 9000 vehicles/hour) similar to the New York traffic scenarios. Compared to the New York scenarios, we were able to add higher background traffic levels into the Beijing traffic scenarios, while avoiding potential gridlocks for baselines. Figure 6c

shows an area near the Beijing railway station. There were nearly 1500 taxi trips within the period. Figure 6d shows a major intersection in the Beijing Chaoyang area. There were nearly 1200 taxi trips within the one-hour period.

Other parameters We set the look-ahead distance (LA) to 600m. As the view of drivers is normally limited in real scenarios, especially for dense road networks in large cities like New York and Beijing, the look-ahead distance in RDE-Setup is shorter compared to SDE-Setup. For example, a 600m look-ahead distance in the New York Mid Town scenario covers around six intersections ahead, which is sufficient for city driving. The remaining parameter values are the same as default values in SDE-Setup.

5.3 Results

We present experimental results in three parts: comparative test results with SDE-Setup, comparative test results with RDE-Setup and parameter sensitivity test results with SDE-Setup.

5.3.1 Comparative tests with SDE-Setup

Impact of Traffic Volume As shown in Fig. 7, CZMP outperforms other baselines in terms of road safety level and travel time in most of the cases. In that figure, lower ATTR values are better travel-time wise, lower ADTTC values are better safety-wise, and lower UIU values are better in terms of computation efficiency. The dotted red line in the bottom chart shows the UIU of 1, which is the maximum allowed update time utilization for real-time usage. CZMP's advantage over other solutions is most visible when the traffic volume is 750. For example, CZMP achieves a 60% reduction of ATTR and a 68% reduction of ADTTC compared to the state-of-the-art solution ICR. This shows the benefit of our approach that optimizes delays first then makes motion plan for all vehicles at once. This approach is only used by CZMP and CZMP-U. The only case where this approach does not work better than others is when traffic volume is at the lowest level (250). In that case, SPG and FPG-L perform better than CZMP in terms of road safety. CZMP only considers delaying a vehicle when there is one or more conflict zones within the look ahead distance from the vehicle. If the number of vehicles is low, the number of conflict zones tends to be low as well, leading to a lower effectiveness of CZMP.

The result shows that CZMP and CZMP-U are computationally efficient. For example, even with 1500 vehicles, the UIU of CZMP is still under 0.1, which means the computation time is under 1 second as the update interval is 10 seconds. This shows that CZMP-based solutions are suitable for real-time usage. The UIU of SPG grows over 1 when the traffic volume is higher than 750. For example, the UIU of SPG reaches 3.44 for 1500 vehicles, which means SPG runs more than 30 times slower than CZMP with that traffic volume. The high computation cost of SPG is due to the repetitive computation of motion plans as motion planning is combined with delay planning in SPG. We do not present the UIU of ICR and ATSC in Fig. 7 because the computation time of these two baselines are negligible due to the fact that they do not process network-level traffic information during optimization. All other solutions need to process network-level information, which can consume a considerable computation time. We also found that the construction of CZG only consumes a small portion of the computation time of CZMP. For example, in the 1500-vehicle scenario, only 12.5% of CZMP's computation time is spent on constructing CZG while the remaining computation time is spent on running Algorithm 2

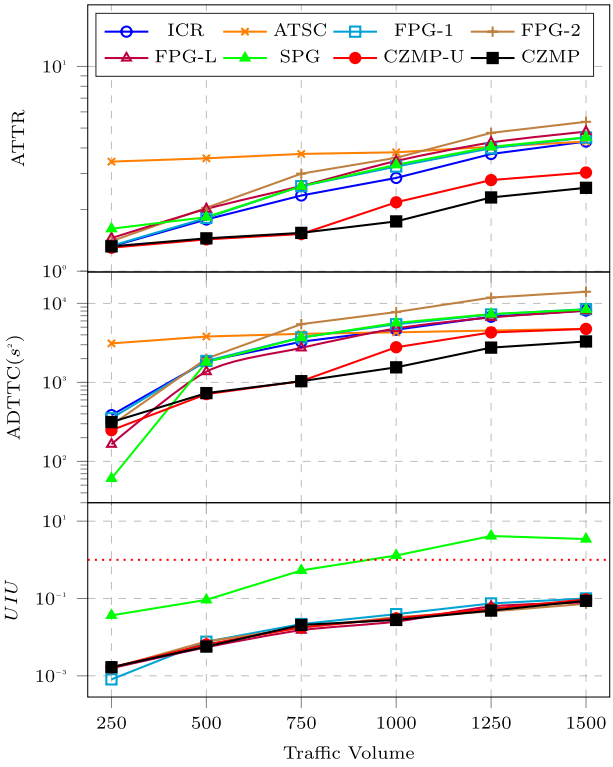


Fig. 7 Results on traffic volume

and Algorithm 3. Although it would be possible to save some computation time if the CZG is updated incrementally rather than being constructed from scratch at each interval, the time saving can be limited based on the results.

Impact of Road Network The results on different road networks are shown in the left part of Fig. 8. Same as Fig. 7, lower values are better in terms of ATTR, ADTTC and UIU. CZMP and CZMP-U achieve better safety level and travel time than other baselines in most cases. For example, in road network RG-3, CZMP achieves a 60% reduction of ATTR and a 67% reduction of ADTTC compared to ICR. CZMP and CZMP-U show better performance in double-lane networks (RG-3 and RG-4) than in single-lane networks (RG-1 and RG-2). As the density of vehicles is lower when the number of lanes is doubled, CZMP-based solutions have more opportunities to increase the inter-vehicular spacing, resulting in better safety levels and better travel times. Interestingly, in single-lane scenarios (RG-1 and RG-2), FPG-L achieves a better safety level compared to CZMP and CZMP-U. This is because the position-based variable spacing values in FPG-L encourage frequent changes of inter-vehicular spacing as vehicles move, which can help improve safety in dense traffic. CZMP-U and CZMP achieve similar ATTR and ADTTC except for road network RG-3, where CZMP performs better as inter-vehicular spacing is optimized using different delay increment values, which allows fine-tuning of the inter-vehicular spaces.

All the solutions, except SPG, can complete optimization in less than one tenth of the update interval. The UIU of SPG is above 1 in some of the scenarios, which makes it unsuitable for real-time usage in those scenarios. The results show that CZMP-based

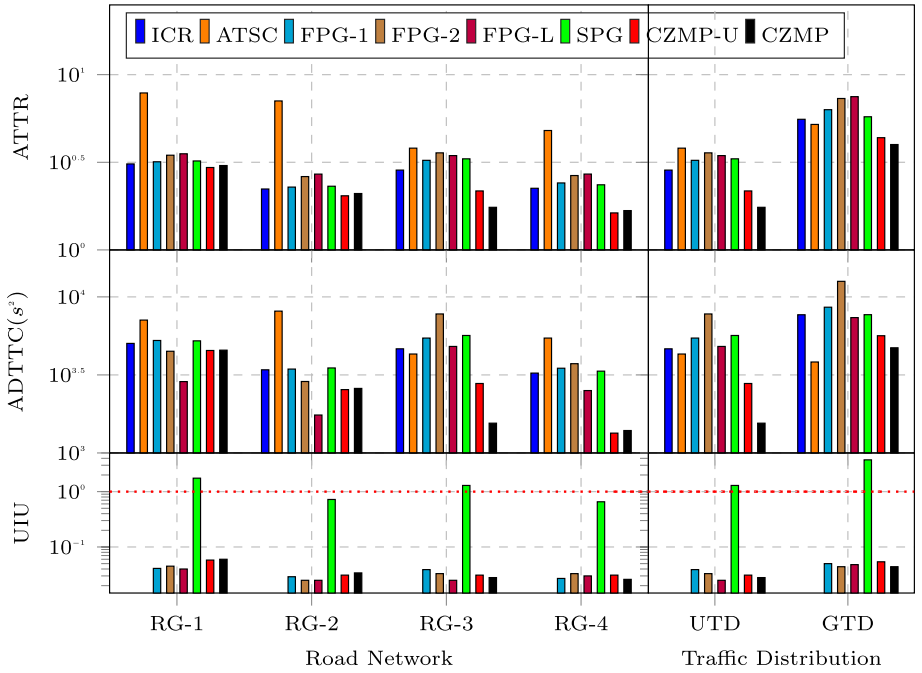


Fig. 8 Results on traffic network and traffic distribution

solutions can achieve a better balance between travel time and safety compared to others. They also have a good computation efficiency. Similar to the results in Fig. 7, the UIU of ICR and ATSC is not shown in Fig. 8 due to the fact that they do not process network-level traffic information, leading to negligible computation time.

Impact of Traffic Distribution The right part of Fig. 8 compares the solutions with two traffic distributions. When the distribution is Gaussian (GTD), ATSC achieves a slightly better safety level compared to the CZMP-based solutions. In GTD, the traffic at the centre of the road network is highly congested, leading to a high probability of vehicle conflicts. ATSC achieves a better safety level in this case as it helps to eliminate a large portion of safety risks at road junctions using traffic signals. However, ATSC performs worse than CZMP-based solutions safety-wise in all other cases in the experiments.

Discussion on the Results of the Tests with SDE-Setup As mentioned in Section 3, our network-level safety optimization problem aims to find the best solution for minimizing a safety metric, Distance to TTC Threshold (DTTC). DTTC is aggregated from the events where the time-to-collision (TTC) of vehicles is below the TTC threshold, beyond which the safety risks are negligible. As DTTC measures the distance to the threshold, a lower DTTC is more preferable because a lower DTTC indicates that vehicles' TTCs are higher, which means vehicles are less likely to collide with each other. The lowest value of DTTC is 0, which is the hypothetical optimum of the metric. When presenting our experimental results, we show the safety level on a per vehicle basis using ADTTC, which is DTTC

divided by the number of vehicles. We use ADTTC because comparing DTTCs directly is less meaningful as the number of vehicles can vary in different scenarios.

Although the ADTTCs of CZMP and CZMP-U are not exceptionally close to the hypothetical optimum, they are closer to the hypothetical optimum than other baselines in most cases (Figs. 7 and 8). In some cases, the reduction of ADTTC achieved by CZMP-based solutions is significant. For example, when the traffic volume is 1500, the ADTTC of CZMP is 59% smaller than that of the state-of-the-art solution ICR. There are only a few cases that the safety levels achieved by some other baselines are closer to the hypothetical optimum than CZMP-based solutions. However, in those cases, CZMP-based solutions outperform other baselines in terms of the travel time metric ATTR. For example, FPG-L achieves a lower ADTTC than CZMP in the RG-1 network but CZMP reduces ATTR from FPG-L by 16.5% in the same network. Overall, CZMP-based solutions achieve a better balance between road safety and travel time. CZMP-based solutions also have low computation costs comparable to other baselines.

5.3.2 Comparative tests with RDE-Setup

Results of New York Traffic Scenarios As shown in Fig. 9, CZMP and CZMP-U outperform other baselines in terms of road safety level and travel time in most cases, especially in moderate to high traffic volume cases. For example, in New York Midtown scenario (left), when background traffic rate is 5400, CZMP-U shows a 27% reduction of ATTR and a 63% reduction of ADTTC compared to the FPG-1 baseline. It shows a similar result with the 7200 background vehicle rate as well. Though FPG-L shows a better safety level at 3600 background vehicle rate, CZMP-U shows a better result travel time-wise. Although SPG shows a safety-wise better result compared to FPG-1 when the background traffic rate is 5400 and 7200, it is not a feasible solution at these two settings, where its UIU is greater than 1. In New York Manhattan Bridge scenario (right), CZMP-U outperforms all the other methods at all the background traffic levels. CZMP-U's advantage over other solutions is most visible when the background traffic rate is 5400. For example, CZMP-U achieves a 15% reduction of ATTR and a 39% reduction of ADTTC compared to FPG-1. Different to the results with SDE-Setup, where ICR showed better performance compared to FPG-1, FPG-1 performs better than ICR in the New York traffic scenarios. However, our solutions CZMP-U and CZMP are significantly better than both FPG-1 and ICR in New York traffic scenarios. When the background traffic rate is 3600, SPG's safety level is similar to CZMP. However, SPG takes a significantly longer computation time compared to CZMP. The result also shows that CZMP and CZMP-U are computationally efficient. In all the scenarios, the maximum value of UIU of our methods is under 0.25, which means the computation time is under 2.5 seconds as the update interval is 10 seconds.

Results of Beijing Traffic Scenarios As shown in Fig. 10, CZMP and CZMP-U outperform other baselines in terms of road safety level and travel time in most cases, especially in moderate to high traffic volume cases, similar to the New York traffic scenarios. For example, in Beijing station area scenario, when back ground traffic rate is 7200, it shows a 2% reduction of ATTR and a 34% reduction of ADTTC compared to the FPG-1 baseline. Although FPG-2 and SPG perform better than CZMP in terms of safety, when the background vehicle rate is 5400, CZMP and CZMP-U show better results travel time-wise.

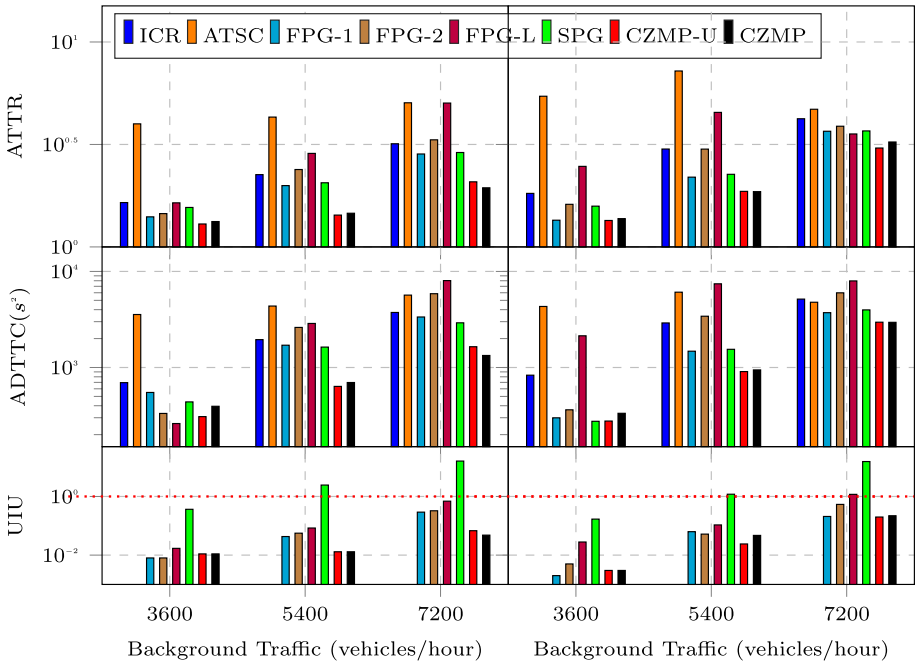


Fig. 9 Results of the traffic scenarios in New York Midtown area (left) and Manhattan Bridge area (right)

In the Beijing Chaoyang area scenario (right), CZMP outperforms all the other methods in terms of safety at all the background traffic levels. In all cases, CZMP shows a minor ATTR reduction of around 1% with an ADTTC reduction ranging between 11%-22%. Although, the ICR baseline is the best method in terms of travel time in this scenario, it increases ADTTC by up to 66% compared to CZMP. SPG shows better safety levels compared to FPG-1 at 7200 and 9000 background traffic rates. However, SPG's computational complexity is several magnitudes higher than FPG-1 so it is not a feasible solution for these settings. The result shows that CZMP and CZMP-U are computationally efficient. In all the scenarios, the UIU of our methods is under 0.06, which means the computation time is under 0.6 seconds as the update interval is 10 seconds.

Discussion on the Results of the Tests with RDE-Setup Overall our methods CZMP and CZMP-U show a better result when the road network is based on a grid plan, such as in the New York Midtown scenario. When the road network is dense (have intersections in closer proximity), the performance of other methods, especially ATSC and ICR, tend to degrade significantly. However, both CZMP and CZMP-U are capable of maintaining a good balance between efficiency and safety even in dense road networks.

5.3.3 Parameter sensitivity tests with SDE-Setup

In this section we discuss the impact of two parameters that are used by CZMP-based solutions. The results on the two parameters are shown in Fig. 11. Lower values are

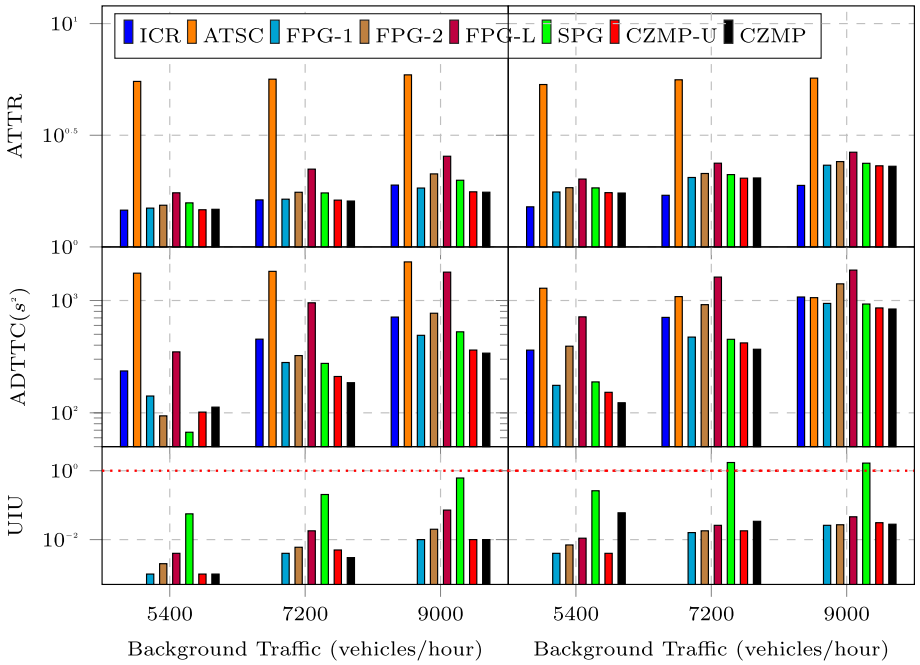


Fig. 10 Results of the traffic scenarios in Beijing station area (left) and Beijing Chaoyang area (right)

better in terms of ATTR, ADTTC and UIU. The dotted red lines in the bottom charts show the *UIU* of 1.

Impact of Look-Ahead Distance The left part of Fig. 11 shows the results on look-ahead distance *LA*. When *LA* increases from 300m, the performance of CZMP improves but the improvement becomes smaller as LA increases. The result shows that a long look-ahead distance, such as 2100m, does not provide significant benefits to safety and travel time as it is difficult to predict traffic far into the future. The computation cost (UIU) increases with the look-ahead distance in a linear fashion, which is as expected because the amount of information that needs to be processed tends to increase when the look-ahead distance increases. For example, a longer look-ahead distance can lead to more conflict zones, which increases the computation cost of optimization. Based on the result, the default value of the look-ahead distance is set to 900m. We also observe that CZMP performs better than CZMP-U as the former solution uses multiple delay increment values to optimize inter-vehicular spacing while the later one only uses a fixed increment value.

Impact of Desired Maximum Travel Time Factor The right part of Fig. 11 shows the results on the desired maximum travel time factor α. When α is 1, the framework cannot add delay to vehicles due to the constraint on desired maximum travel time (Section 4.2). With the increase of α, CZMP tends to consider more vehicles into the optimization. The result shows that CZMP can improve safety level and achieve a good balance between safety and travel time when a limited set of vehicles get delayed. For example, when α

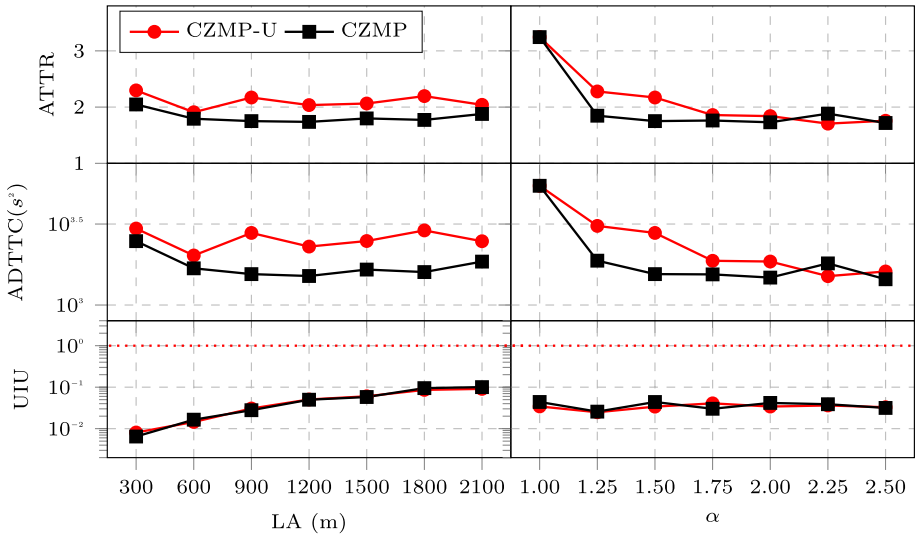


Fig. 11 Results on CZMP parameters LA and α

increases from 1 to 1.5, CZMP's ATTR reduces by 67% and its ADTTC reduces by 72%. When increasing α beyond 1.5, both ATTR and ADTTC do not change significantly for CZMP. In terms of safety and travel time, CZMP-U performs worse than CZMP when α is between 1.25 and 1.75. This shows that using a fixed delay increment (CZMP-U) is not as effective as using multiple delay increment values (CZMP). *UIU* remains largely unchanged when varying α for both solutions.

6 Conclusions and future work

Our proposed CZMP framework can achieve a good balance between road safety and travel time using network-level spatio-temporal data. The framework is particularly suitable for real-time applications due to its low computation costs. Our experimental results show the significant advantages of the framework over other data management solutions. CZMP improves the safety level by up to 68% while reducing the travel time by up to 60% against the state-of-the-art traffic-efficiency-oriented solution ICR. CZMP also runs more than 30 times faster than our preliminary solution SPG in certain scenarios.

The current design of CZMP assumes that all the vehicles would follow the instructions given by the framework. A future work can evaluate the performance of the framework when a portion of the vehicles cannot follow the instructions. It would also be interesting to incorporate more types of information, such as weather conditions, into the framework. A future work can also extend CZMP for multi-modal transport that includes cars, trains and trams. Finally, in the current design of CZMP, we generate the conflict zone graph from scratch at each update interval. A future extension to CZMP can consider incremental updates of the graph. However, as we have discussed in the experimental results, only 12.5% of the computation time is spent on building the graph in our current design so the benefit of the incremental

version may be limited. The framework decouples each major step of the workflow, and it enables better usability and extendability for future applications.

Author Contributions Not applicable.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Data availability The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

Code availability The code of our method(s) will be made publicly available on acceptance.

Declarations

Conflicts of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethics approval We have considered the ethical implications and have none to report.

Consent to participate Not applicable.

Consent for publication Not applicable.

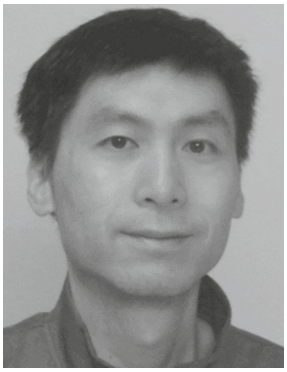
References

1. Siegel JE, Erb DC, Sarma SE (2018) A survey of the connected vehicle landscape - architectures, enabling technologies, applications, and development areas. IEEE Trans ITS 19(8):2391–2406. https://doi.org/10.1109/TITS.2017.2749459
2. Giridhar A, Kumar PR (2006) Scheduling automated traffic on a network of roads. IEEE Trans Vehicular Technol 55(5):1467–1474. https://doi.org/10.1109/TVT.2006.877472
3. Jin Q, Wu G, Boriboonsomsin K, Barth MJ (2013) Platoon-based multi-agent intersection management for connected vehicle. In: ITSC, pp 1462–1467 https://doi.org/10.1109/ITSC.2013.6728436
4. Gunarathna U, Xie H, Tanin E, Karunasekera S, Borovica-Gajic R (2020) Real-time lane configuration with coordinated reinforcement learning. ECML PKDD 12460:291–307. https://doi.org/10.1007/978-3-030-67667-4_18
5. Motallebi S, Xie H, Tanin E, Qi J, Ramamohanarao K (2021) Route intersection reduction with connected autonomous vehicles. GeoInformatica 25:99–125. https://doi.org/10.1007/s10707-020-00420-z
6. Xu L, Wang LY, Yin GG, Zhang H (2014) Communication information structures and contents for enhanced safety of highway vehicle platoons. IEEE Trans Vehicular Technol 63(9):4206–4220. https://doi.org/10.1109/TVT.2014.2311384
7. Dresner KM, Stone P (2008) Mitigating catastrophic failure at intersections of autonomous vehicles. In: AAMAS, vol 3, pp 1393–1396. https://dl.acm.org/citation.cfm?id=1402881. Accessed 21 Nov 2021

8. Xie H, Tanin E, Karunasekera S, Qi J, Zhang R, Kulik L, Ramamohanarao K (2019) Quantifying the impact of autonomous vehicles using microscopic simulations. In: IWCTS, pp 5–1510. https://doi.org/10.1145/3357000.3366145
9. Khondaker B, Kattan L (2015) Variable speed limit: A microscopic analysis in a connected vehicle environment. Transportation Research Part C: Emerging Technologies 58:146–159. https://doi.org/10.1016/j.trc.2015.07.014
10. Liu H, Wei H, Zuo T, Li Z, Yang YJ (2017) Fine-tuning ADAS algorithm parameters for optimizing traffic safety and mobility in connected vehicle environment. Transportation Research Part C: Emerging Technologies 76:132–149. https://doi.org/10.1016/j.trc.2017.01.003
11. Muthugama L, Karunasekera S, Tanin E (2020) Platooning graph for safer traffic management. In: ACM SIGSPATIAL, pp 453–456. https://doi.org/10.1145/3397536.3422272
12. Liu C, Lin C, Shiraishi S, Tomizuka M (2018) Distributed conflict resolution for connected autonomous vehicles. IEEE T-IV 3(1):18–29. https://doi.org/10.1109/TIV.2017.2788209
13. Namazi E, Li J, Lu C (2019) Intelligent intersection management systems considering autonomous vehicles: A systematic literature review. IEEE Access 7:91946–91965. https://doi.org/10.1109/ACCESS.2019.2927412
14. Guo Q, Li L, Jeff X (2019) Urban traffic signal control with connected and automated vehicles : A survey. Transportation Research Part C: Emerging Technologies 101:313–334. https://doi.org/10.1016/j.trc.2019.01.026
15. Chen L, Englund C (2016) Cooperative intersection management: A survey. IEEE Trans ITS 17(2):570–586. https://doi.org/10.1109/TITS.2015.2471812
16. Jeong J, Jeong H, Lee E, Oh TT, Du DHC (2016) SAINT: Self-adaptive interactive navigation tool for cloud-based vehicular traffic optimization. IEEE Trans Vehicular Technol 65(6):4053–4067. https://doi.org/10.1109/TVT.2015.2476958
17. Islam SMABA, Hajbabaie A (2017) Distributed coordinated signal timing optimization in connected transportation networks. Transportation Research Part C: Emerging Technologies 80:272–285. https://doi.org/10.1016/j.trc.2017.04.017
18. Yu L, Yu J, Zhang M, Zhang X, Liu Y, Zhang H, Min W (2019) Large scale traffic signal network optimization - A paradigm shift driven by big data. In: ICDE, pp 1832–1840. https://doi.org/10.1109/ICDE.2019.00199
19. Dresner KM, Stone P (2008) A multiagent approach to autonomous intersection management. JAIR 31:591–656. https://doi.org/10.1613/jair.2502
20. Yu C, Feng Y, Liu HX, Ma W, Yang X (2019) Corridor level cooperative trajectory optimization with connected and automated vehicles. Transportation Research Part C: Emerging Technologies 105L:405–421. https://doi.org/10.1016/j.trc.2019.06.002
21. Mehrabipour M, Hajbabaie A (2017) A cell-based distributed-coordinated approach for network-level signal timing optimization. Computer-Aided Civil and Infrastructure Engineering 32(7):599–616. https://doi.org/10.1111/mice.12272
22. Keyvan-Ekbatani M, Yildirimoglu M, Geroliminis N, Papageorgiou M (2015) Multiple concentric gating traffic control in large-scale urban networks. IEEE Trans ITS 16(4):2141–2154. https://doi.org/10.1109/TITS.2015.2399303
23. Vegamoor VK, Darbha S, Rajagopal KR (2019) A review of automatic vehicle following systems. Journal of the Indian Institute of Science 99(4):567–587. https://doi.org/10.1007/s41745-019-00143-7
24. Axelsson J (2017) Safety in vehicle platooning: A systematic literature review. IEEE Trans ITS 18(5):1033–1045. https://doi.org/10.1109/TITS.2016.2598873
25. AlDwyish A, Tanin E, Karunasekera S (2017) Follow the best: Crowdsourced automated travel advice. In: MobiQuitous, pp 272–281. https://doi.org/10.1145/3144457.3144475
26. Pan B, Demiryurek U, Gupta C, Shahabi C (2015) Forecasting spatiotemporal impact of traffic incidents for next-generation navigation systems. KAIS 45(1):75–104. https://doi.org/10.1007/s10115-014-0783-6
27. Galbrun E, Pelechrinis K, Terzi E (2016) Urban navigation beyond shortest route: The case of safe paths. Inform Syst 57:160–171. https://doi.org/10.1016/j.is.2015.10.005
28. Hendawi AM, Rustum A, Ahmadain AA, Hazel D, Teredesai A, Oliver D, Ali MH, Stankovic JA (2017) Smart personalized routing for smart cities. In: ICDE, pp 1295–1306. https://doi.org/10.1109/ICDE.2017.172
29. Islam FT, Hashem T, Shahriyar R (2021) A privacy-enhanced and personalized safe route planner with crowdsourced data and computation. In: ICDE, pp 229–240. https://doi.org/10.1109/ICDE51399.2021.00027
30. Zhou F, Li X, Ma J (2017) Parsimonious shooting heuristic for trajectory design of connected automated traffic part I: Theoretical analysis with generalized time geography. Transportation Research Part B: Methodological 95:394–420. arXiv:1511.04810. https://doi.org/10.1016/j.trb.2016.05.007

31. Ma J, Li X, Zhou F, Hu J, Park BB (2017) Parsimonious shooting heuristic for trajectory design of connected automated traffic part II : Computational issues and optimization. Transportation Research Part B 95:421–441. https://doi.org/10.1016/j.trb.2016.06.010
32. Wang Y, Li X, Yao H (2019) Review of trajectory optimisation for connected automated vehicles. IET Intell Trans Syst 13(4):580–586. https://doi.org/10.1049/iet-its.2018.5184
33. Rios-Torres J, Malikopoulos AA (2017) Automated and cooperative vehicle merging at highway on-ramps. IEEE Trans ITS 18(4):780–789. https://doi.org/10.1109/TITS.2016.2587582
34. Xu Z, Wang Y, Wang G, Li XS, Bertini RL, Qu X, Zhao X (2021) Trajectory optimization for a connected automated traffic stream: Comparison between an exact model and fast heuristics. IEEE Trans ITS 22(5):2969–2978. https://doi.org/10.1109/TITS.2020.2978382
35. Wei Y, Avcı C, Liu J, Belezamo B, Aydın N, Li P, Zhou X (2017) Dynamic programming-based multi-vehicle longitudinal trajectory optimization with simplified car following models. Transportation Research Part B: Methodological 106:102–129. https://doi.org/10.1016/j.trb.2017.10.012
36. Essa M, Sayed T (2020) Self-learning adaptive traffic signal control for real-time safety optimization. Accident Analysis & Prevention 146:105713. https://doi.org/10.1016/j.aap.2020.105713
37. Perronnet F, Buisson J, Lombard A, Abbas-Turki A, Ahmane M, Moudni AE (2019) Deadlock prevention of self-driving vehicles in a network of intersections. IEEE Trans ITS 20(11):4219–4233. https://doi.org/10.1109/TITS.2018.2886247
38. Lin Y, Hsu H, Lin S, Lin C, Jiang IH, Liu C (2019) Graph-based modeling, scheduling, and verification for intersection management of intelligent vehicles. ACM TECS 18(5s):95–19521. https://doi.org/10.1145/3358221
39. Morando MM, Tian Q, Truong LT, Vu HL (2018) Studying the safety impact of autonomous vehicles using simulation-based surrogate safety measures. Journal of Advanced Transportation 2018. https://doi.org/10.1155/2018/6135183
40. Minderhoud MM, Bovy PH (2001) Extended time-to-collision measures for road traffic safety assessment. Accident Analysis & Prevention 33(1):89–97. https://doi.org/10.1016/S0001-4575(00)00019-1
41. Sharon G, Stern R, Felner A, Sturtevant NR (2015) Conflict-based search for optimal multi-agent path-finding. Artificial Intelligence 219:40–66. https://doi.org/10.1016/j.artint.2014.11.006
42. Kesting A, Treiber M, Helbing D (2010) Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 368(1928):4585–4605. arXiv:0912.3613
43. Ramamohanarao K, Xie H, Kulik L, Karunasekera S, Tanin E, Zhang R, Khunayn EB (2017) SMARTS: Scalable microscopic adaptive road traffic simulator. ACM TIST 8(2):26–12622. https://doi.org/10.1145/2898363
44. Newell GF (1989) Theory of highway traffic signals. https://escholarship.org/uc/item/7zn2b9bc. Accessed 21 Nov 2021
45. Yuan J, Zheng Y, Xie X, Sun G (2013) T-drive: Enhancing driving directions with taxi drivers' intelligence. IEEE Trans Knowl Data Eng 25(1):220–232. https://doi.org/10.1109/TKDE.2011.200

Lakmal Muthugama received his B.Sc. (Hons) degree in computer science and engineering from the University of Moratuwa, Sri Lanka, in 2015. He worked as a Specialist Software Engineer at MillenniumIt Software Pvt. Ltd., Sri Lanka, the capital markets technology section of the London Stock Exchange Group. He is currently working towards his Ph.D. degree in computer science at the University of Melbourne. His current research interests include spatial algorithms, traffic management with autonomous vehicles and road safety optimization.



Hairuo Xie is a research fellow at the School of Computing and Information Systems, University of Melbourne. His current research interests include spatial algorithms, traffic management, traffic simulation and distributed computing. He architected a distributed microscopic traffic simulator that has been used extensively in the research community.

Egemen Tanin finished his PhD at the University of Maryland at College Park before joining the University of Melbourne. He is a professor at the School of Computing and Information Systems. He is an Associate Editor of ACM TSAS and served as a PC Chair for ACM SIGSPATIAL 2011 and 2012. He was elected to serve as the Treasurer for ACM SIGSPATIAL and served in this role till 2017. Dr Tanin was also elected to be the Secretary of the SIG, 2017-2020. He was elected to be the next Vice-Chair of SIG in 2020. He is the co-founder of ACM SIGSPATIAL Australia.

Shanika Karunasekera received the B.Sc. degree in electronics and telecommunications engineering from the University of Moratuwa, Sri Lanka, in 1990 and the Ph.D. degree in electrical engineering from the University of Cambridge, Cambridge, U.K., in 1995. From 1995 to 2002, she was a Software Engineer and a Distinguished Member of Technical Staff with Lucent Technologies, Bell Labs Innovations, USA. In January 2003, she joined the University of Melbourne, Victoria, Australia, and currently she is a professor in the School of Computing and Information Systems. Her current research interests include distributed system engineering, distributed data-mining and social media analytics.