Real-Time Road Network Optimization with Coordinated Reinforcement Learning

UDESH GUNARATHNA, University of Melbourne, Australia
HAIRUO XIE, University of Melbourne, Australia
EGEMEN TANIN, University of Melbourne, Australia
SHANIKA KARUNASEKERA, University of Melbourne, Australia
RENATA BOROVICA-GAJIC, University of Melbourne, Australia

Dynamic road network optimization has been used for improving traffic flow in an infrequent and localized manner. The development of intelligent systems and technology provides an opportunity to improve the frequency and scale of dynamic road network optimization. However, such improvements are hindered by the high computational complexity of the existing algorithms that generate the optimization plans. We present a novel solution that integrates machine learning and road network optimization. Our solution consists of two complementary parts. The first part is an efficient algorithm that uses reinforcement learning to find the best road network configurations at real time. The second part is a dynamic routing mechanism, which helps connected vehicles adapt to the change of road networks. Our extensive experimental results demonstrate that the proposed solution can substantially reduce the average travel time in a variety of scenarios, whilst being computationally efficient and hence applicable to real-time situations.

CCS Concepts: • **Information systems → Spatial-temporal systems**; • **Computing methodologies → Reinforcement learning**.

Additional Key Words and Phrases: Spatial Data Management, Autonomous Vehicles, Dynamic Lane-Reversal

ACM Reference Format:
Udesh Gunarathna, Hairuo Xie, Egemen Tanin, Shanika Karunasekera, and Renata Borovica-Gajic. 2018. Real-Time Road Network Optimization with Coordinated Reinforcement Learning. *J. ACM* 37, 4, Article 111 (August 2018), 28 pages. https://doi.org/10.1145/1122445.1122456

1 Introduction

Dynamic road network optimization, such as changing the speed limit of roads during peak hours, has been widely used for improving traffic flow. The optimization is normally performed at a low frequency, e.g., a few times a day, and in a localized manner as the optimizations are normally focused on a small part of a road network. With the development of intelligent systems and technology, the real-time data from cameras, counters, inductive loops and smartphones provides an increasingly precise, real-time and global view of the entire road network. This creates the possibility to improve the frequency and scale of dynamic road network optimization. However, there exists a challenge that hinders such improvements, which is the high computational complexity of the algorithms

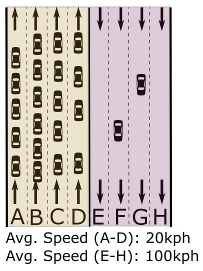
Authors' addresses: Udesh Gunarathna, University of Melbourne, Melbourne, Australia, pgunarathna@student.unimelb.edu.au; Hairuo Xie, University of Melbourne, Melbourne, Australia, xieh@unimelb.edu.au; Egemen Tanin, University of Melbourne, Melbourne, Australia, etanin@unimelb.edu.au; Shanika Karunasekera, University of Melbourne, Melbourne, Australia, karus@unimelb.edu.au; Renata Borovica-Gajic, University of Melbourne, Melbourne, Australia, renata.borovica@unimelb.edu.au.

0004-5411/2018/8-ART111 $15.00
https://doi.org/10.1145/1122445.1122456

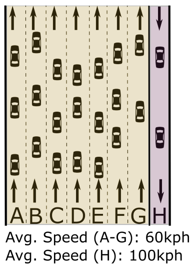
for computing road network configurations. Most of the existing algorithms in this area are based on bi-level/linear programming that aims to find the exact optimal solution to an optimization problem [8, 14, 32]. Due to the inherently high computational complexity, they are not suitable for optimizing road network holistically in a large network, e.g., a network of a city, in real time [8].

The advancements in reinforcement learning [27] provide an opportunity to address the computational challenge. Reinforcement learning has been used for solving many dynamic and real-time sequential decision-making problems effectively and efficiently. The application of reinforcement learning has achieved a big success in many fields such as games, robotics, and finance. For road network optimization, reinforcement learning-based methods have an advantage over traditional mathematical programming-based methods in that reinforcement learning can generate approximate results that are close to the optimal without sophisticated mathematical models. In other words, reinforcement learning can be used as a heuristic, which helps improve the efficiency of the optimization process significantly. Therefore, in this work we are interested in using reinforcement learning for large-scale optimization of road networks.

We focus on an example type of road network optimization, which is the reversal of traffic lane directions for balancing the usage of traffic lanes [18]. Such dynamic changes are particularly important for emerging traffic environments dominated by connected and autonomous vehicles (CAVs), where vehicles can more readily alert and adapt to lane changes. The benefit of lane-direction changes can be shown in the following example, where 20 vehicles are moving north-bound and 2 vehicles are moving south-bound (Figure 1). The original lane configuration assigns 4 lanes for north-bound traffic and 4 lanes for south-bound traffic (Figure 1a), resulting in the congestion of north-bound traffic. Figure 1b shows a better lane configuration where the direction of lane E, F and G are reversed, which leads to a significantly faster north-bound traffic. Our aim is to make these changes at a larger scale and in a dynamic way.



(a) Traffic before lane-direction change



(b) Traffic after lane-direction change

Fig. 1. The impact of lane-direction change on traffic flow. There are 20 vehicles moving in the north-bound direction and 2 vehicles moving in the south-bound direction.

In our previous work, we proposed Coordinated Learning-based Lane Allocation (CLLA) [12] for optimizing lane-directions in dynamic traffic environments. CLLA uses a set of *Reinforcement Learning Agents (RL Agents)* that work with an intelligent transportation system (ITS) to control lane directions across a road network. An RL Agent uses reinforcement learning to determine the best lane-direction change for specific road segments based on the real-time traffic flow on those road segments. The proposed changes are then sent to a set of *Coordinating Agents* that evaluate the global impact of the proposed lane-direction changes based on the predicted traffic demand, which can be represented by the number of vehicle routes that pass through specific road segments. The traffic demand information is maintained in a data structure called Path Dependency Graph (PDG). After evaluating the proposed changes, the coordinating agents approve the changes that have a positive impact on the traffic efficiency at the global level.

Although CLLA can help improve traffic efficiency to a certain extent, it has two limitations, which motivates us to develop a more advanced solution presented in this paper. The first limitation of CLLA is that the RL Agents are allocated on a per intersection basis, which means the sub-area controlled by an RL Agent includes all the road segments that are connected to a specific intersection. We observe that this implementation reduces the flexibility of handling different types of intersections. To address this limitation, we develop an extended version of CLLA, named **CLLA+**, which reduces the size of the area controlled by an RL Agent from CLLA. Specifically, we allocate RL Agents on a per road segment basis which allows decision making at a more granular level. As a result, the RL Agent is able to assign better lane-direction configurations that further reduce the travel time of vehicles. The second limitation of CLLA is that it does not consider the impact of dynamic lane-direction changes on the quality of vehicle routes. The change of lane directions can lead to the change of traffic conditions. The original route plan of vehicles may not suit the new conditions, which can at times result in less optimal travel times [5]. This limitation also applies to other existing solutions. To address this problem, we develop a routing mechanism, named **lane-configuration aware routing**, to approximate the optimal traffic assignment when both lane-configuration and traffic conditions change in real-time. The routing mechanism periodically computes the shortest path based on the latest lane configurations and traffic conditions. Leveraging CLLA+ and the lane-configuration aware routing, we propose a complete road network optimization solution, where CLLA+ optimizes lane configurations and the routing mechanism optimizes vehicle routes.

We evaluate the effectiveness of our proposed road network optimization solution in a vast variety of traffic scenarios using microscopic simulations. The proposed solution is able to substantially reduce the travel time in all the simulated traffic scenarios. For example, in the experiments carried out using the Manhattan road network with real traffic data, the proposed solution achieves a 8% travel time reduction compared with the baseline solution. Note that such levels of traffic flow improvement are seen as a significant improvement in Traffic Engineering terms (and should not be confused with orders of magnitude gain in computation time that is expected and common in Computer Science [6]).

Our contributions can be summarized as follows: 1. We formalize the real-time lane-direction optimization problem in road networks. 2. We introduce CLLA+, a coordinated, scalable, reinforcement learning-based algorithm to optimize lane directions at a highly granular level. 3. We introduce a real-time lane-configuration aware traffic routing mechanism to compute the fastest routes under dynamic lane-configurations. 4. We propose a complete road network optimization solution that combines CLLA+ with lane-configuration aware routing for real-time large-scale road network optimization. 5. We present an extensive experimental evaluation of the road network optimization solution and provide insights into how dynamic lane-direction allocation can influence the traffic.

2 Related Work

2.1 Learning-based Traffic Optimization

A majority of the existing traffic optimization methods are based on traffic flow optimization with linear programming [11, 17]. These methods are effective if traffic demand and congestion levels are relatively static. When there is a significant change in the network, the optimization solutions need to be re-computed from scratch. Due to the high computational complexity of finding an optimal solution, these methods are not suitable for highly dynamic traffic environments where real-time traffic information should be considered in the optimization process.

With the rise of reinforcement learning [26], a new generation of traffic optimization methods have emerged [13, 28]. For example, Arel et al. show that a multi-agent system can optimize the timing of adaptive traffic lights based on reinforcement learning [2]. In reinforcement learning, an agent can find the rules to achieve an objective by repeatedly interacting with an environment. The interactive process can be modelled as a finite Markov Decision Process, which requires a set of states S and a set of actions A per state. Given a state s of the environment, the agent takes an action a. As the result of the action, the environment state may change to s' with a reward *r*. The agent then decides on the next action in order to maximize the reward in the next round. Reinforcement learning-based approaches can suggest the best actions for traffic optimization given a combination of network states, such as the queue size at intersections [2]. They have an advantage over linear programming-based approaches, since if trained well, they can optimize traffic in a highly dynamic network. In other words, there is no need to re-train the agent when there is a change in the network. Different to the aforementioned approaches, our solution uses reinforcement learning for optimizing lane-directions.

A common problem with reinforcement learning is that the state space can grow exponentially when the dimensionality of the state space grows linearly. The fast growth of the state space can make reinforcement learning unsuitable for large scale deployments. This problem is known as the *curse of dimensionality* [3]. A common way to mitigate the problem in traffic optimization is by partitioning the road network into small sub-areas and using different learning agents to optimize for different sub-areas. This approach has been used for dynamic traffic signal control [10]. We apply this approach to optimize citywide traffic at real time.

2.2 Lane Direction Optimization

An important type of road network optimization is focused on lane-direction configurations. Dynamic lane-direction changes can be an effective way to improve traffic efficiency [16]. However, existing approaches for optimizing lane-directions are based on linear programming [8, 14, 32], which are unsuitable for dynamic traffic environments due to their high computational complexity. For example, Chu et al. use linear programming to make lane-allocation plans by considering the schedule of CAVs [8]. Their experiments show that the total travel time can be reduced. However, the computational time grows exponentially when the number of vehicles grows linearly, which can make the approach unsuitable for highly dynamic traffic environments. The high computational costs are also inherent to other approaches [14, 32]. Furthermore, all these approaches assume that the exact knowledge of traffic demand over the time horizon is known beforehand, which does not hold when traffic demand is stochastic [19]. On the contrary, our proposed approach CLLA+ is lightweight and can adapt to highly dynamic situations based on reinforcement learning. The reinforcement learning agents can find effective lane-direction changes for individual road segments even when traffic demand changes dramatically.

Some existing approaches can work with real-time data where traffic demand is not known *a priori*, but do not focus on network-level lane allocations. For example, Ampountolas *et al.* [1] discuss lane allocation at the road segment level using an expressway. Mitrovic *et al.* [21] develop a solution to compute lane allocation of road segments at an intersection-level. *To the best of our knowledge, we are the first to propose an efficient lane-direction allocation solution based on real-time traffic information at the road network level.*

2.3 Traffic Assignment

Traffic assignment concerns the allocation of vehicle routes. There are two major traffic assignment paradigms namely *system optimal* and *user equilibrium* [24]. The objective of the system optimal traffic assignment is to minimize the total travel time of all vehicles, whereas the user equilibrium traffic assignment refers to assigning routes to vehicles so that no vehicle can gain a better travel time by switching to an alternate route. Some existing traffic optimization solutions are based on user equilibrium traffic assignment along with lane-direction allocations [14, 32]. There is also research on system optimal traffic assignment along with lane-direction allocations [19].

To compute both types of traffic assignments, traffic information for the entire time horizon is needed. Unfortunately, in real-time lane-allocation, such information is not available. User equilibrium traffic assignment however can be approximated by dynamically recomputing the shortest paths (i.e., traffic assignment) at regular time intervals [7]. In this paper, we employ a similar approach in lane-configuration aware traffic assignment.

3 Problem Definition

A road network can be represented as a graph $G(V, E)$, where each edge $e \in E$ represents a road segment and each vertex $v \in V$ represents a start/end point of a road segment.

Definition 1: The lane configuration of an edge $e \in E$, lc_e, contains two numbers, each of which is the number of lanes in a specific direction on the edge. The sum of the two numbers is always equal to the total number of lanes on the edge.

Definition 2: The dynamic lane configuration of an edge $e \in E$ at time t, $lc_e(t)$, is the lane configuration at the time point.

Definition 3: A dynamic road network configuration at time t can be represented as a graph $G_t(V, E)$, with dynamic lane configuration $lc_e(t)$ applied to all the edges.

Definition 4: All the vehicles in the road network $G(V, E)$ at time t that are yet to finish their route are represented as a set U_t. The total number of vehicles in the road network at time t, $n(t)$ ($= |U_t|$), is equal to the size of the set U_t.

Definition 5: The estimated travel cost of a vehicle $i \in U_t$, $ETC_i(t)$, is the estimated travel time that the vehicle takes to move from its source to its destination, which is computed with respect to the dynamic road network configuration $G_t(V, E)$ at time t.

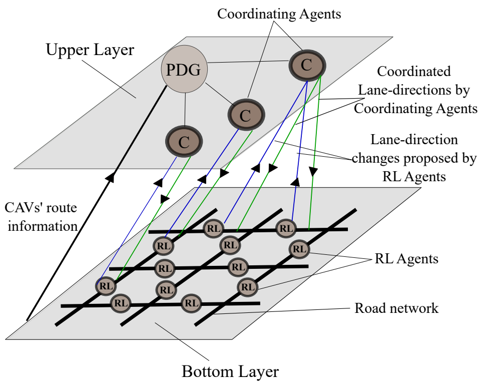
Definition 6: The actual travel cost of a vehicle i, ATC_i, is the actual travel time taken by the vehicle to move from its source to its destination. The actual travel time can be affected by many factors, such as the waiting time at intersections and traffic congestion.

Definition 7: The actual total travel cost of the entire network, $ATTC$, is the sum of the actual travel costs of all the vehicles. That is, $ATTC = \sum_{i=1}^{N} ATC_i$, where N is the total number of vehicles that existed in the road network.

Problem Statement: Given a set of vehicles U_t at time t, where each vehicle $i \in U_t$ has an estimated travel time $ETC_i(t)$ and the road network configuration $G_{t-1}(V, E)$ from time $t - 1$, find the new configuration $G_t(V, E)$ by computing dynamic lane configuration ($lc_e(t)$) for all the edges of E such that the actual total travel cost $ATTC$ is minimized.

4 Lane-Configuration based Road Network Optimization

A complete solution to the aforementioned optimization problem needs to have two parts, a real-time lane-configuration optimization method and a real-time rerouting mechanism. The former part optimizes for road infrastructure while the later part optimizes for individual CAVs. There is a strong interplay between the two parts. When lane-configuration of the road network changes, the original route of CAVs may not work well because future traffic conditions can be impacted by the lane-configuration changes. Consequently, CAVs may need to reroute in order to minimize their own travel time under the new traffic conditions. On the other hand, the rerouting of CAVs can also change future traffic conditions. To adapt to the new traffic conditions, lane-configurations may need to be updated. Our complete


Coordinating Agents
Upper Layer
PDG
C
C
C
Coordinated Lane-directions by Coordinating Agents
Lane-direction changes proposed by RL Agents
CAVs' route information
RL
RL Agents
Road network
Bottom Layer
Fig. 2. The CLLA+'s architecture which consists of two layers. The Path Dependency Graph is shown as PDG.

Algorithm 1: CLLA+

```
   Parameter: $ti$, time interval between two coordinating operations(Assignment interval)
   Parameter: Local Lane-direction Changes ($LLC$) proposed by the RL Agent
   Parameter: $G$, road network
   Parameter: Coordinated Lane-direction Changes ($CLC$) by the Coordinating Agents
1  $t \leftarrow 0$, $t_{step} \leftarrow 0$
2  while True do
3      foreach $agent \in RL\ Agents$ do
4          get best lane-configuration for the edge $e$ (road), $lc_e(t)$, controlled by the $agent$
5          if $lc_e(t)$ is different to the current configuration of the edge then
6              $LLC.insert(\{e, lc_e(t)\})$
7      if $t_{step} = ti$ then
8          $CLC \leftarrow$ Global Impact Evaluation($LLC$)
9          $LLC \leftarrow \emptyset$, $t_{step} \leftarrow 0$
10         foreach $\{e, lc_e(t)\}$ in $CLC$ do
11             apply the lane-direction change to $e$
12     $t \leftarrow t + 1$
13     $t_{step} \leftarrow t_{step} + 1$
```

solution consists of the two parts. In this section, we first describe the lane-configuration optimization solution (CLLA+), followed by the rerouting mechanism (Lane-configuration Aware Routing) and then the interplay between the two parts.

4.1 Enhanced Coordinated Learning-based Lane Allocation

Enhanced Coordinated Learning-based Lane Allocation (CLLA+) provides a highly scalable solution for dynamic lane configuration at the road network level by using two sets of software agents, Reinforcement Learning (RL) Agents and Coordinating Agents. The architecture of CLLA+ is shown in Figure 2, where RL Agents work in the bottom layer and Coordinating Agents work in the upper layer. There is also a data structure named Path Dependency Graph (PDG) in the upper layer. PDG maintains the predicted traffic flow of the entire road network. It is updated periodically based on route plan of vehicles, which would be readily accessible to a traffic management system in the era of CAVs.

Different RL Agents optimize lane-direction configurations for different road segments independently. When RL Agents recognize lane-direction changes that are beneficial to the local traffic, they propose the changes to the Coordinating Agents, which evaluate whether the proposed change would be beneficial to traffic flow at the global level based on the PDG. The Coordinating Agents then ask the RL Agents to execute the globally beneficial changes. The Coordinating Agents may also decide further changes in addition to the proposed changes.

4.1.1 Optimization Procedure

The procedure for optimizing lane-direction configuration is detailed in Algorithm 1. During one iteration of the algorithm, each RL Agent can propose a lane-direction change at a specific road segment using the process detailed in Section 4.1.2. The Coordinating Agents evaluate the proposed changes periodically. When it is time to evaluate the proposed changes, the Coordinating Agents use the *Global Impact Evaluation* algorithm (Section 4.1.3) to quantify the conflicts between the proposed changes and find the globally beneficial changes (Line 8). The coordinated lane-direction changes are then applied to the road segments (Line 10-11).

4.1.2 Reinforcement Learning Agent (RL Agent)

In CLLA+, the RL Agents use Q-learning technique [30] to find suitable lane-direction changes based on real-time traffic conditions. The Q-learning algorithm aims to find a policy that maps a state to an action. The algorithm relies on an *action value function*, $Q(s, a)$, which computes the quality of a state-action combination. Q-learning tries to find the optimal policy that leads to the maximum action value. Q-learning updates the action-value function using an iterative process as shown in Equation 1 [27].

$$Q(s_t, a_t) \leftarrow (1 - \alpha).Q(s_t, a_t) + \alpha(r_t + \gamma.max_a Q(s_{t+1}, a)) \tag{1}$$

where s_t is the current state, a_t is the action taken, s_{t+1} is the next state as a result of the action, $max_a Q(s_{t+1}, a)$ is the estimated action value in the next state, value r_t is an observed reward as a result of the action at t, α is a learning rate and γ is a discount factor. The iterative update of Equation 1 approximates the optimal action value function $Q^*(s_t, a_t)$ for every state-action pair.

In our previous work [12], an RL Agent controls a set of road segments connected to a specific intersection. However, allocating RL Agents per intersection basis reduces the flexibility when there are road segments with a different number of lanes connected to an intersection. In this extension, RL Agents are allocated per road segment basis. Though this increases the number of RL Agents, the benefit of the new approach are two-fold. First, it avoids the situation where the same road is controlled by two RL Agents. So there is no duplicate decision-making for a road segment. Second, the state-space of an RL Agent is reduced significantly, which can lead to a faster learning process. The reduced state-space makes CLLA+ more adaptive to dynamic environments. The states, actions, and rewards used by the RL Agents are defined as follows.

States: First, we denote the two possible directions of a lane as *upstream* and *downstream*. Assume an edge points from vertex v_1 to vertex v_2 and the vertices have identification numbers, we say that the direction $v_1 \longrightarrow v_2$ is **upstream** if v_1 is lower than v_2 or **downstream** if v_1 is higher than v_2. An RL Agent works with three types of states as shown below.

1. The first state represents the number of vehicles in the upstream direction of the road segment at a given time step. To avoid high variances in the measurement of the number of vehicles, the moving average of the number of vehicles from the last n time-steps is used.

2. The second state represents the number of vehicles in the downstream direction of the road segment. The moving average of the number of vehicles is used.

3. The third state represents the lane configuration of the road segment.

Although it is possible to add other types of states, we find that the combination of these states works well.

Actions: There are three possible actions which include increasing the number of upstream lanes by 1, increasing the number of downstream lanes by 1, and keeping the current configuration. When the number of lanes in one direction is increased, the number of lanes in the opposite direction is decreased at the same time.

Once an RL Agent executes a lane-direction change action, the effect of the action may not be reflected on the road segment immediately for road safety reasons. For example, all the vehicles already travelling in a lane should move away from the lane before the lane can be used by vehicles traveling from the opposite direction. There is a transition period between the time that a lane-direction action is taken and the time that vehicles start entering that lane from the opposite direction. We call this transition period the *lane clearing time*. This transition period is an important factor which affects the efficiency of our overall solution as an increase of lane clearing time increases the total travel time. The impact of this factor is examined in our experiments.

Rewards: We define the reward based on output flow, which is the number of vehicles leaving a road segment during a specific period. When the output flow is high, more vehicles are able to leave a road segment per unit time [14]. A higher output flow at a road segment can lead to a lower travel time, or a lower travel cost, at the segment. Therefore, based on our objective of minimizing the total travel cost (Section 3), the reward function is defined as:

$$R = -abs(\frac{n_{up}}{lc_{up}} - \frac{n_{down}}{lc_{down}})/\frac{(n_{up} + n_{down})}{(lc_{up} + lc_{down})} \tag{2}$$

In this function, n_{up} and n_{down} represent the number of upstream vehicles and the number of downstream vehicles in a road segment. The number of upstream lanes and the number of downstream lanes are denoted as lc_{up} and lc_{down}. The denominator term is used to normalize the reward function.

Even though the reward function is described in terms of the flow, it helps to reduce the travel time (as in the objective of the problem definition) on the road segment. The reward balances the traffic per lane in both directions such that one direction is not congested compared to the other. Allocating more lanes to the direction with a higher volume of traffic increases the throughput in that direction allowing a larger number of vehicles to leave the road segment in a unit-time. This reduces the average time spent by a vehicle in that road segment as more vehicles can leave the road segment [8].

4.1.3 *Coordinating Agent*

Given a locally optimized lane-direction change, Coordinating Agents check whether the change can help improve traffic efficiency in the surrounding areas based on the predicted traffic demand and the current traffic conditions. If a proposed change is beneficial, it can be actioned. Otherwise, it is not allowed by CLLA+.

The coordinating process considers the predicted traffic demand at the global level based on the routes of vehicles. This can be explained with a simple example shown in Figure 3, where 2 vehicles are moving from A to D while 4 vehicles are moving in the opposite direction. In this scenario, the overall traffic demand would be from right to left. Without considering the global traffic trend, the RL Agent for road segment e_1 may propose to increase the number of lanes from A to B because there is no vehicle in the opposite direction on e_1 now. Although such a lane-direction change would help reduce the travel time on e_1, it would conflict with the global traffic trend, which can lead to congestion in the opposite direction. Consequently, increasing the number of lanes from A to B is not beneficial at the global level and should not be actioned.

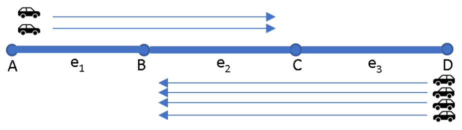


Fig. 3. The vehicles on a road with three road links, e_1, e_2 and e_3.

Due to the dynamic nature of traffic, especially in the era of CAVs when road network optimization can be performed frequently, the Coordinating Agents may not need to consider the full route of vehicles as the route may change dynamically at real time. The Coordinating Agents can collect the partial route within a *lookup distance*. For example, assuming the lookup distance is 10 road segments, the Coordinating Agents only need to know the next 10 road segments that the vehicles will pass.

The coordination of lane-direction changes is performed at a certain time interval. The time between two coordinating operations is called *assignment interval*, within which the proposed lane-direction changes and the information about vehicle routes are collected from the road network. The vehicle route information is fed into a data structure named *Path Dependency Graph*. Based on this data structure, the coordinating agents compute the current traffic condition and predict future traffic condition across the whole road network.

Path Dependency Graph (PDG): The main responsibility of PDG is to predict the amount of traffic flow from one road segment to another. In PDG, a vertex corresponds to a road segment. The flow between two road segments, which may or may not be connected to the same intersection, is represented as an edge between two vertices. PDG is constructed from a road network as follows.

Let us denote a road network as a graph $G(V, E)$ where V is the set of intersections, and E is the set of road segments. Let us denote P as a set of vehicle paths at a given time t. A vehicle path $p \in P$ is represented through a set of edges, as following $p = \{e_0, e_1, ..., e_n\};\ e_0, e_1, ..., e_n \in E, p \in P$. The first edge on the path is the edge of the vehicle at time t.

We construct *Path Dependency Graph*, $PDG(V^{pdg}, E^{pdg})$, as a directed graph. Vertices of PDG represent the roads in the road network. i.e. edges of road network graph G are mapped to vertices of PDG, vice versa. This relationship can be described with a function $f : e \in E \Leftrightarrow v \in V^{pdg}$.

In real-world road networks, traffic conditions tend to change with high randomness in a brief period, but the conditions tend to show a stable trend over an extended period. To make a more accurate prediction of traffic, PDG maintains several attributes that consider both the current and historic values. Specifically, such attributes are based on the exponentially weighted moving average ($EWMA$) function, which is defined in Equation 3.

Given a time series s, the value of s in the current time step t, r_t, the EWMA of the series in the previous step $t-1$, $EWMA_{t-1}$, and a weight μ between 0 and 1, the EWMA of the series in the current time step is:

$$EWMA_t = \mu \times r_t + (1-\mu) \times EWMA_{t-1} \tag{3}$$

As shown in Equation 3, EWMA is a recursive function. The EWMA of the current step can be influenced by the value of earlier steps. The weight μ controls the level of influence from previous time steps.

In PDG, a vertex v has two types of attributes (note that v maps to a road segment).

1. Weighted traffic loads $x_{up,v}(t)$ and $x_{down,v}(t)$. These are the EWMA of the number of vehicles in both upstream and downstream directions at road segment v at time step t. These two attributes show the level of traffic flow in two directions. The value of the two attributes can be computed using the EWMA function described above. For example, assuming that r_t in Equation 3 is the number of vehicles in the upstream direction in time step t, $x_{up,v}(t)$ can be computed as $x_{up,v}(t) = \mu \times r_t + (1-\mu) \times x_{up,v}(t-1)$.

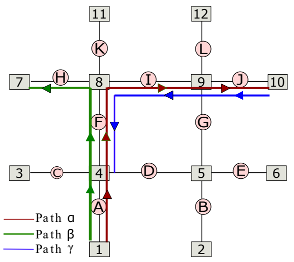
2. Lane configuration $lc_v(t)$: The lane configuration of road v at time t.

An edge (v_1, v_2) in PDG has two attributes, upstream flow ($f_{up,(v_1,v_2)}(t)$) and downstream flow ($f_{down,(v_1,v_2)}(t)$). These edge attributes represent the expected number of vehicles (traffic flow) that move from one road segment (v_1) to another road segment (v_2) according to their route plan. Here, *up* and *down* indicate from which direction (upstream or downstream) traffic flow starts from v_1. To limit the impact of the variances of traffic flow, the attributes are computed as EWMA of traffic flow. We believe it is possible to acquire accurate values for these attributes in the era of connected vehicles, where vehicle route information can be sent to Coordinating Agents from the vehicles directly or be relayed to the agents through the transportation infrastructure [23]. Based on the vehicle route information, the EWMA of traffic flow can be computed. For example, assuming r_t in Equation 3 is the number of vehicles that are travelling in the upstream direction on road segment v_1 at time t and will enter road segment v_2 at a later time, the upstream flow $f_{up,(v_1,v_2)}(t)$ can be computed as $f_{up,(v_1,v_2)}(t) = \mu \times r_t + (1-\mu) \times f_{up,(v_1,v_2)}(t-1)$.

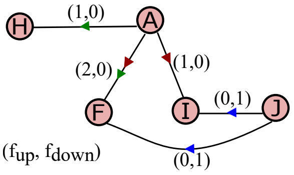
At any time step t, we construct an edge of PDG, denoted as (v_1, v_2), between two vertices ($v_1, v_2 \in V^{pdg}$), if there is at least one vehicle path that covers the corresponding road segments of both v_1 and v_2 (note that v_1 is mapped to the current road segment of the vehicle and the road segment may not be adjacent to the corresponding segment of v_2). In other words, an edge in PDG denotes a relationship between a pair of road segments on a vehicle's path. We construct such an edge if the edge does not already exist in the PDG. It may not be necessary to consider all the road segments on vehicles' routes when updating PDG. This is because the impact of lane-direction change at a road segment is generally small on road segments that are far away. Due to this reason, we set an upper limit to the number of road segments in vehicle paths when building PDG. This limit is set to the aforementioned *lookup distance*, denoted as l_d. Only the next l_d number of road segments that a vehicle needs to pass through are considered when updating PDG. For a given path p, v_1 should be the current edge of the vehicle on the path and v_2 should be any of the remaining edges on path p.

The process of generating a PDG can be explained with the following example. We show a set of vehicle paths in an example road network (Figure 4a) and the corresponding PDG (Figure 4b). The road network has 12 roads segments (A to L). The paths of the vehicles are Path α, Path β and Path γ. Path α passes through 4 edges (A, F, I, J). Path β passes through 3 edges (A, F, H). Path γ passes through 3 edges (J, I, F). In the corresponding PDG, there are 5 vertices, each of which is mapped to a road segment with one or more vehicle paths. For each edge in PDG, f_{up} and f_{down} are shown in brackets near the corresponding edge. For simplicity, we set μ to 1. The lookup distance is set to 3.

The example PDG contains an edge from A to I with ($f_{up} = 1, f_{down} = 0$), because the vehicle in Path α will pass A then pass I and Path α starts from the upstream direction of A. There is an edge from A to F with ($f_{up} = 2, f_{down} = 0$) because there are two vehicles (Path α and β) that will go from A to F, starting from the upstream direction of A. There is an edge from J to I with ($f_{up} = 0, f_{down} = 1$), because the vehicle in Path γ passes from J then passes I starting from the downstream direction of J. The edge attributes of other edges in Figure 4a can also be explained similarly. We should note that there is no edge from A to J even though the vehicle in Path α goes from A to J. This is because J is the 4th edge in Path α, thus J is not within the lookup distance, which is 3, from the vehicle's current edge A.



(a) A simple road network with three paths (red, green and blue).



(b) Path Dependency Graph (PDG) based on the road network in Figure 4a. Red arrows, green arrows and blue arrows correspond to Path α, Path β and Path γ in Figure 4a, respectively.

Fig. 4. Path Dependency Graph example

PDG is similar to the dual graph [31] in terms of how the edges of a primal graph are represented as vertices in the transformed graph. However the relationship between the edges in PDG is more complex and based on the current vehicle paths. This is different from the dual graph, where the edges are constructed if they share a common vertex in the primal graph.

Global Impact Evaluation Algorithm (GIE): The Coordinating Agents use Global Impact Evaluation Algorithm (see Algorithm 2) to quantify the conflicts between lane-direction changes. The algorithm takes lane-direction changes that are proposed by the RL Agents as input (*LLC*). A lane-direction change in the input consists of a road ID and the configuration of the lane-directions at the road segment. The coordination is performed as the below steps.

Algorithm 2: Global Impact Evaluation (GIE)

```
Input: Local Lane-direction Changes ($LLC$) proposed by the RL Agent
Input: $t$, current time
Input: $PDG$, Path Dependency Graph
Output: Coordinated Lane-direction Changes ($CLC$) approved by the Coordinating Agents
1  $q \leftarrow \emptyset$; $CLC \leftarrow \emptyset$
2  $pf_{up,r}, pf_{down,r} \leftarrow 0$
3  foreach $(r, lc_r(t)) \in LLC$ do
4      $roads \leftarrow$ get neighboring road segments for $r$ from $PDG$
5      foreach $r_{new} \in roads$ do
6          if $change\ in\ r\ affects\ r_{new}\ in\ upstream$ then
7              $pf_{up,r_{new}} += f_{up,(r,r_{new})}$
8          else
9              $pf_{down,r_{new}} += f_{down,(r,r_{new})}$
10         if $r_{new}\ not\ in\ q$ then
11             $q.add(r_{new})$
12 foreach $r_{new} \in q$ do
13     $lc_{r_{new}}(t) \leftarrow \emptyset$; $is\_conflict \leftarrow False$;
14     $target\_dir \leftarrow$ direction which has higher $pf$ value; (up or down)
15     $oppose\_dir \leftarrow$ opposite direction to $target\_dir$; (up or down)
16     if $(pf_{target\_dir,r_{new}} > x_{oppose\_dir,r_{new}})$ then
17         $lc_{r_{new}}(t) \leftarrow$ add one lane to $target\_dir$ direction
18     else
19         set $is\_conflict$ to $True$
20     if $(is\_conflict == True)$ then
21         $r_{new}$ cannot accommodate predicted traffic flows;
22         mark corresponding change in $LLC$ as a conflict
23     if $lc_{r_{new}}(t)\ contains\ a\ lane\ direction\ change$ then
24         $CLC.add([r_{new}, lc_{r_{new}}(t)])$
25 foreach $(r, lc_r(t)) \in LLC$ do
26     if $no\ conflicts\ for\ r$ then
27         $CLC.add([r, lc_r(t)])$
```

First, the algorithm finds the road segments that are affected by a proposed lane-direction change at a road segment (r) within the lookup distance. This is done by using PDG's edges from r in Line 4 (note that r_{new} is a vertex in PDG). For each affected road segment, the algorithm finds the predicted traffic flow (pf) using the edge attributes f_{up} and f_{down} of PDG (Line 6-9), which maintain the average flow at a road segment with a proposed lane-direction change to an affected road segment. If a road segment is affected by multiple proposed changes, the traffic flow relevant to all the changes is aggregated when computing pf_{up}/pf_{down}. Then the algorithm adds the affected road segments to a queue (Line 11). The queue contains all the road segments affected by the changes proposed by RL Agents.

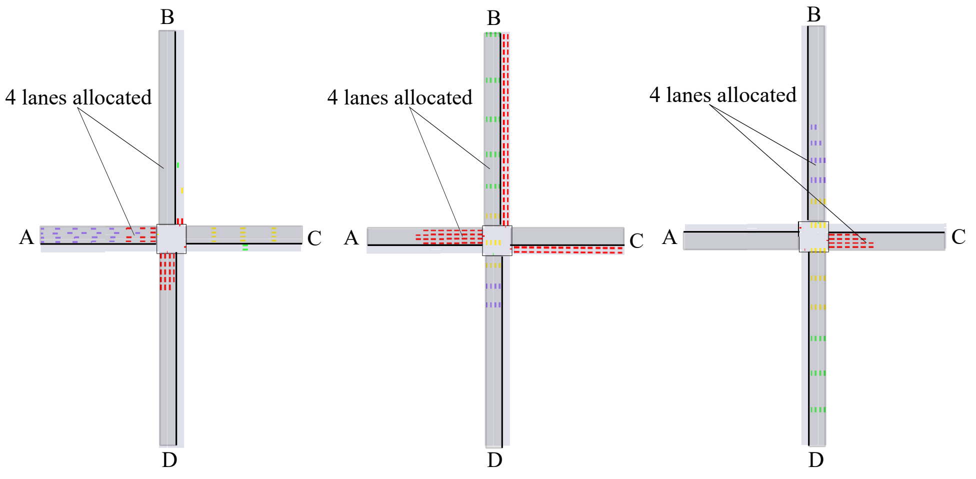
In the next step, the algorithm visits each road segment in the queue to determine the appropriate lane-direction configuration ($lc_{r_{new}}(t)$) by checking whether the road segment can accommodate the predicted traffic flow (pf) using PDG's x_{up} and x_{down} attributes (Line 13-19). Here, x_{up} and x_{down} represent the weighted traffic load in a road segment that corresponds to a PDG vertex. The general rationale behind this part is that the affected road segments, rather than the segments where the proposed changes would happen, may also need to change lane-direction configurations to adapt to the change of traffic flow in the near future. But increasing traffic lanes at one direction may deteriorate traffic condition in the opposite direction if the traffic flow in the opposite direction is expected to be heavier. Therefore one needs to determine the appropriate lane-direction change based on the expected traffic flow in both directions. The decision-making process of Line 13-19 can be explained in the following manner. First, we select the direction of road segment r_{new}, in which the predicted traffic flow is higher. This direction is denoted as $target_{dir}$. This is the direction in which we should increase the number of lanes. For example, if the proposed changes will lead to the increase of both pf_{down} and pf_{up} at the same road segment r_{new}, we set the direction which has the higher pf value as the $target_{dir}$ and the opposite direction as the $opposite_{dir}$ (Line 13-15). If the predicted flow for the target direction ($pf_{target\_dir,r_{new}}$) is higher than the opposite direction's average traffic flow ($x_{oppose\_dir,r_{new}}$) then we register an additional change of lane configuration ($lc_{r_{new}}$) by adding a new lane in the target direction (Line 16-17). Otherwise, if ($x_{oppose\_dir,r_{new}}$) is higher we cannot register a change for r_{new} because there is a larger flow in the opposite direction. In such situations, we should not approve the original proposed change at r (Line 18-19). In such a case, we mark it as a conflict by making the variable $is\_conflict$ to true. If there is a conflict in an affected road segment, the algorithm marks the corresponding change proposed by an RL Agent as a conflict (Line 20-21). Otherwise, the additional lane-direction change for the affected road segment r_{new} is added to Coordinated Lane Changes (CLC) (Line 24). At the end of the loop in Line 24, CLC contains all the additional changes that need to be made.

In the last step, the algorithm adds each of the original lane-direction changes proposed by the RL Agents to CLC if the change does not involve a conflict (Line 25-27).

Complexity of Coordinating Process: Let us use m to denote the number of requests from the RL Agents. The complexity of visiting the relevant road segments is $O(m \times neb)$ where neb is the number of neighboring road segments that connect to a road segment at a road junction. Since the number of road segments connecting with the same junction is normally a small value, neb can be seen as a constant value with a given lookup distance (l_d). Hence the algorithm complexity can be simplified to $O(m)$. In the worst case, there is a lane-change request for each road segment of $G(V,E)$, leading to a complexity of $O(|E|)$.

Example Policy by CLLA+: In this example, we focus on a synthetic intersection with two roads, AC and BD. Both roads are two-way with 6 lanes. The synthetic network is simulated in the SMARTS simulator. We present screenshots of the simulation at various stages (Figure 5). In the screenshots, each vehicle is shown as a colored dot. Red vehicles are stopped, and green ones are moving at free-flow speed. Vehicles with other colors are moving at speeds between 0 and the free-flow speed.

At the start of the scenario (the first subfigure), direction AC and direction DB are allocated with 4 lanes each due to the high demand of traffic in the two directions. Then, the trend of traffic demand has a sudden change, where many vehicles start to travel in direction CA and BD. The second subfigure shows the moment of the traffic demand change. We can see that there are still many vehicles in the AC direction that need to be cleared. After those vehicles are cleared, the RL framework applies a policy to change the direction of some lanes. As shown in the last subfigure, we can see that direction CA and BD get 4 lanes each. When the screenshot is taken, the congestion in direction BD has reduced significantly, demonstrating the positive effect of lane configuration change.



(a) Initial traffic condition (b) Right after traffic demand is reversed (c) New lane-configuration after traffic demand is reversed

Fig. 5. Change of traffic demand and lane directions around an intersection.

4.2 Lane-Configuration Aware Routing

When lane-directions are changed, the travel time on road segments may increase or decrease, which can affect the choice of the fastest route between two locations. This is the reason that lane-direction configuration solutions are normally coupled with traffic assignment that optimizes vehicle routes. As mentioned in Section 2.3, an ideal traffic assignment can reach a system optimal or a user equilibrium state. Both are difficult to achieve in a highly dynamic traffic environment due to the high computational complexity of the problem. However, user equilibrium-based traffic assignment can be approximated by recomputing the fastest path at small time intervals for all the vehicles whenever traffic conditions are changed [5, 7]. Such incremental route changes typically yield a similar route choice given by the traditional user equilibrium traffic assignment. Existing work on lane-direction changes do not consider such frequent rerouting of vehicles. We detail a lane-configuration aware routing mechanism that allows CAVs to reroute frequently based on dynamic lane-direction changes.

Dynamic Computation of Fastest Route: As described in Section 2.3, user equilibrium traffic assignment aims to route vehicles such that no vehicle can gain a shorter travel time by switching to an alternate route. Traditional user equilibrium-based traffic assignment requires all the traffic information within the entire time-horizon to be known at the beginning of the computation. Differently, as CLLA+ works with real-time data, we do not assume that the future traffic information is available. Thus we cannot compute user equilibrium traffic assignment in a traditional manner.

In order to compute the fastest path in a dynamic traffic environment, one should be able to estimate the travel time on a road segment based on the traffic conditions at any time. The well known Bureau of Public Roads (BPR) function can be used for this purpose [9]. The BPR function estimates the travel time through a road segment by considering the current traffic flow and the flow capacity. Note that the travel time on the two directions of a road segment may be different. For simplicity, in this section, we only consider one direction of a road segment. Equation 4 shows how to estimate the travel time based on the traffic flow at a road segment.

$$T_e(t) = T_{f,e} \times (1 + \zeta(\frac{I_e(t)}{O_e(t)})^{\delta}) \tag{4}$$

In this equation, $T_e(t)$ is the travel time estimated for road segment e (for a specific direction) and $T_{f,e}$ is the free flow travel time. $I_e(t)$ is the actual flow attempting to use road segment e and $O_e(t)$ is flow capacity of road segment e at time t respectively. We use the standard values for ζ and δ (0.15 and 4 respectively [9]) in our implementation.

The BPR function can be modified to account for the lane configuration as follows, where the flow capacity $O_t(e)$ is expressed as a function of the number of lanes [14].

$$T_e(t) = T_{f,e} \times (1 + \zeta(\frac{I_e(t)}{u_e * lc_e(t)})^{\delta}) \tag{5}$$

In this equation, u_e is the maximum flow capacity of a lane and $lc_e(t)$ is the number of lanes in road segment e in the considered direction at time t.

The modified BPR equation can be used to update the travel time on road segments. The updated travel time can be used as the weight of road segments when computing the fastest path using a shortest path algorithm such as the Dijkstra's algorithm.

4.3 Interplay between CLLA+ and Lane-Configuration Aware Routing

In our proposed solution, CAVs use Lane-Configuration Aware Routing to compute the shortest path whenever lane-configurations or traffic conditions change. We assume frequent re-routing is possible in the era of CAVs as the vehicles can receive traffic management information constantly and reroute accordingly. On the other hand, the PDG in CLLA+ is updated based on the route changes of vehicles. The updating of PDG is performed as follows. Whenever vehicles change their routes, the upstream flow (f_{up}) and the downstream flow (f_{down}) (Section 4.1.1) of the affected edges in PDG are updated. When running the **GIE** algorithm in CLLA+, Coordinating Agents always use the up-to-date PDG to compute the coordinated lane allocation.

5 Experimental Methodology

We compare the proposed road network optimization solution, which includes CLLA+ and lane-configuration aware routing, against four baseline algorithms. The results are collected from traffic simulations with synthetic traffic data and real traffic data. We use SMARTS (Scalable Microscopic Adaptive Road Traffic Simulator) [25] to perform traffic simulations.

5.1 Datasets

Our experiments are conducted with two datasets. To evaluate how the algorithms perform in customized traffic conditions, we use a synthetic 7x7 grid network which is similar to the road network used in a recent study on lane-direction changes with CAVs[8]. During an experiment, vehicles are generated in the first 40 minutes into the simulation. After 40 minutes, the simulation keeps running until all vehicles that are generated reach their destination.The shortest path between the source and the destination is computed using the Dijkstra's algorithm with Equation 5. We simulate four traffic patterns using the synthetic road network. The traffic patterns are as follows.

1. Rush hour traffic (RH): In this setup, the traffic demand is directionally imbalanced to represent rush hour traffic patterns. Directionally imbalanced traffic means there is a large difference in the traffic demand between the two directions at individual road segments.

2. Bottleneck traffic (BN): This setup generates a high volume of traffic passing through the centre of the grid network. All the vehicles generated in this setup need to pass the centre of the grid during their journey. This type of traffic patterns create bottleneck links at the centre of the network. This can simulate the traffic around the downtown area of many large cities.

3. Mixed traffic (MX): Mixed traffic contains both **rush hour** and **bottleneck** traffic conditions in the same network with each traffic pattern accounts for 50% of total traffic.

4. Random traffic (RD): Traffic is generated randomly where the source node and the destination node of each vehicle are selected uniformly at random from the road networks nodes.

We also use a real-world traffic dataset that contains the taxi trip records from the New York City[1]. The data includes the source, the destination, and the start time of the taxi trips in the city. We pick an area of Manhattan for simulation (Figure 6) because the area contains a larger amount of taxi trip records compared to other areas. The area consists of 656 road segments and 366 intersections. We consider two versions of Manhattan road network. One is the **original Manhattan road network** where the number of lanes in each road segment is the same as the value in the original road map. For this road network, there is no lane-direction change in two-lane road segments because we assume that there should be at least one lane in either direction at any time. The other version is an **up-scaled Manhattan road network** where we up-scaled the number of lanes in road segments such that all the road segments have 6 lanes. We create this version for two reasons. First, we want to simulate a large volume of traffic in a real-world road network. Second, the up-scaling allows us to change the lane-directions at all road segments in the road network.

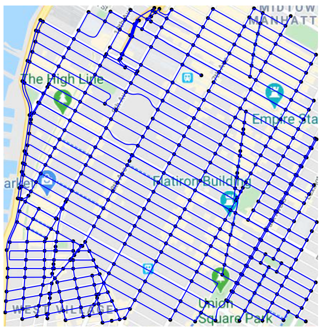


Fig. 6. The road network of Midtown Manhattan used in the experiments.

The road network of the simulation area is loaded from OpenStreetMap[2]. For a specific taxi trip, the source and the destination are mapped to the nearest OpenStreetMap nodes. Since the taxi data can not represent all the traffic in the road network (according to the reports [1,3], yellow taxi trips account for around 10% of traffic), we up-sampled the data to create a certain amount of traffic. More specifically, for each taxi trip record, we generate 10 vehicles with the same source-destination pair in the simulation. We have up-sampled the traffic further in the **up-scaled Manhattan road network** so that the both **up-scaled Manhattan road network** and **original Manhattan road network** have same traffic congestion levels. Upsampling is a common practice for estimating the total traffic volume based on a limited set of real data. For example, a recent work uses upsampling to simulate realistic traffic volume based on the same Manhattan TAXI dataset as our work [6]. Another work uses upsampling to estimate traffic volume in Hong Kong based on taxi counts [20].

The shortest path between the source and the destination is computed using the Dijkstra's algorithm. The travel cost of the road segments is the travel time computed using Equation 5. During an experiment, vehicles are generated during the first 30 minutes of simulation. After the 30 minutes, the simulation keeps running until all vehicles finish their routes (since the Manhattan network is much larger than the synthetic 7x7 grid network, traffic is generated only in the first 30 minutes so that all vehicles finish their routes within an hour). In our experiments, the traffic signals by default use static timing in all solutions unless otherwise specified as dynamic timing.

5.2 Comparison Baselines

Different to our proposed solution, existing lane-direction configuration methods [8, 14, 32] assume the arrival time of all vehicles (future traffic dynamics) are known at the beginning of the simulation. As this assumption is not valid when working with real-time traffic data, in our work, we do not

[1]https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page
[2]https://www.openstreetmap.org
[3]http://www.nyc.gov/html/dot/downloads/pdf/mobility-report-singlepage-2019.pdf

assume such future traffic dynamics are known a *priori.* Therefore, we cannot compare our solution with them in this work. Instead, we conduct comparative tests against the following solutions which can work with real-time data.

By default the baselines contain the following settings unless specified in the baseline description. 1) traffic signal timing is fixed; 2) cars can reroute; 3) car routes are computed with lane-configuration aware routing; 4) lane directions are changed. We denote our proposed solution as CLLA+ for the rest of the section. The baseline solutions are as follows.

1. No Lane-direction Allocations (**noLA**): This solution does not do any lane-direction change. Since noLA does not change the lane-configuration of the road network, lane-configuration aware routing mechanism is only used for re-routing when traffic conditions change. This solution uses fixed traffic signals.

2. Demand-based Lane Allocations (**DLA**): This solution assumes that the full knowledge of traffic demand at given time step and vehicle routes are known at any given time step. DLA computes expected traffic flow for every edge for both directions by projecting the current vehicle paths to each edge. This way we compute the number of vehicles that plan to go from each edge. Then in this method, we allocate more lanes for a specific direction when the traffic demand per lane in the direction is higher than the traffic demand per lane in the opposite direction. Same as CLLA+, DLA configures lane-directions at a certain time intervals. This solution uses fixed traffic signals and car routes are computed with lane-configuration aware routing.

3. Local Lane-direction Allocations (**LLA**): This solution uses multiple learning agents to decide lane-direction changes. The optimization is performed using the approach described in Section 4.1.2. LLA is similar to CLLA+ but there is no coordination between the agents. This solution uses fixed traffic signals and car routes are computed with lane-configuration aware routing.

4. Dynamic Traffic Signals (**DTS**): All the above solutions and CLLA+ by default use fixed traffic signals. This solution uses DTS and does not change lane-configurations. The lane-configuration aware routing mechanism is only used for re-routing when traffic conditions change.

5. Dynamic Traffic Signals and CLLA+ (**DTS_CLLA+**): CLLA+ by default uses fixed traffic signals. This solutions combines CLLA+ with dynamic traffic signals.

5.3 Evaluation Metrics

We measure the performance of the solutions based on the following metrics.

Deviation from Free-Flow Travel Time: The free-flow travel time of a vehicle is the shortest possible travel time achieved when the vehicle travels at the speed limit of the roads without slowing down at traffic lights during its entire trip. Deviation from Free-Flow Travel Time ($DFFT$) is defined as in Equation 6, where $t_a(i)$ is the actual travel time of vehicle i, $t_f(i)$ is the free-flow travel time of the same vehicle and n is the total number of vehicles.

$$DFFT = \frac{1}{n} \times \sqrt{\sum_{i=1}^{n} (1 - (t_a(i)/t_f(i)))^2} \tag{6}$$

The value (t_a/t_f) shows the extent that a vehicle's travel time deviates from the free flow travel time. Since the minimum value of (t_a/t_f) is 1, we can measure traffic efficiency by computing the standard deviation of the gaps between 1 and (t_a/t_f) for all vehicles. A lower DFFT means most of the vehicles in the simulation reach their destination without taking much longer time than free flow travel time.

Average Travel Time: We compute the average travel time based on all the vehicles that complete their trips during a simulation. A higher average travel time indicates that the traffic is more congested during the simulation. Our solution aims to reduce the average travel time.

Travel Time Gain: Travel time gain (TT_gain) is used to evaluate the advantage of CLLA+ over the baselines. Since the range of average travel time changes between different traffic scenarios, it is difficult to compare CLLA+'s advantage between different scenarios by just using the average travel time. *TT_gain* provides a way to measure improvement from CLLA+ over baselines across different traffic scenarios. The metric is defined with the following equation. $TT\_gain(x) = (t_x - t_{CLLA+})/t_x$. In this equation, x refers to any baseline defined in Section 5.2. The variables t_x and t_{CLLA+} refer to the average travel time achieved by the baseline x and CLLA+ respectively.

Total Number of Lane-Direction Changes: Frequent changes to lane directions may lead to chaotic scenes, especially in cities with complex road networks. It would be difficult to manage traffic congestion and traffic accidents when lane-direction changes become frequent. This metric is used to evaluate the stability of road networks. When the total number of lane-direction changes across the entire network is lower, the stability of the road network is higher.

5.4 Parameter Settings

For LLA and CLLA+, the learning rate α is 0.001 and the discount factor used by Q-learning is 0.75. These settings are found through a parameter sweep which maximises the reward function in Section 4.1.2. The RL Agents are pre-trained using the above parameters, based on the traffic at a single road segment before they are deployed to all the roads in a road network. We conduct two types of tests, Comparative Test and Sensitivity Test. In Comparative Test, we compare baseline solutions with CLLA+ in the synthetic road network and the real-world road network with different traffic patterns. The values of the parameters, except for the traffic pattern parameter, are set to the default values given in Table 1. In Sensitivity Test, we analyse the performance of CLLA+ for each parameter shown in Table 1. For each experiment in the sensitivity analysis, we vary one parameter within the given *Range* in Table 1 while other at their *Default Value* unless specified otherwise in the experiment. All the sensitivity tests use the synthetic road network in the simulations. The details of the parameters are as follows.

5.4.1 CLLA+ parameters

1. Lane-configuration Aware Routing: This parameter indicates whether CLLA+ uses lane-configuration aware routing mechanism. CLLA+ uses lane-configuration aware routing mechanism by default. When CLLA+ does not use lane-configuration aware routing we denote the solution as **CLLA_FR**. When CLLA+ uses the average travel speed of a road segment as the rerouting metric rather than the lane-configuration aware routing we denote the solution as **CLLA_ATS**.

2. Assignment Interval: Assignment interval is the time interval in which the proposed lane-direction changes are coordinated and executed.

3. Lookup Distance: As mentioned earlier, CLLA+ uses only the edges within the lookup distance (l_d) in a vehicle path when building the PDG. In other words, lookup distance affects the global influence of traffic from neighboring road segments, which may be beyond the immediate neighbors of a road segment. A very large value of lookup distance may not lead to a better traffic efficiency. To determine the meaningful range of this parameter, we perform a parameter sweep until we observe a steady travel time performance.

4. Moving Average Window Size for PDG: Using the number of vehicles of one time step leads to a high variance of the value across adjacent time steps. To reduce the variance, we use the moving average of the number of vehicles as the input to CLLA+. The computation of moving average values uses a recursive function that can consider multiple previous time steps (Equation 3). The moving average window size (the length of the period covering the previous steps for computation) should be set to a meaningful level. If the size is zero, the computation does not consider the traffic conditions of any previous step. If the size is significantly large, the moving average may not reflect the recent trend of traffic flow. We experiment with the moving average window size between 0 and 20 minutes.

5. Maximum Number of Conflicts Allowed in CLLA+: CLLA+ rejects the lane-changes proposed by RL Agents if the changes conflict with the traffic in other road segments. In Global Impact Evaluation algorithm, a proposed change is discarded if there is at-least one conflict (Algorithm 2 Line 15-17). Therefore the default value of the maximum number of allowed conflicts is 0. In this experiment, we increase the threshold to evaluate the impact of this parameter.

6. Communication Delay: The communication between RL agents may experience delays in the real world due to numerous factors such as noisy communication channels. This parameter controls the level of communication delay in the experiments. We simulate the effect of communication delay as follows. With a non-zero delay value, lane direction changes are not applied immediately when the decisions are made. Instead, they are applied after the delay period. In addition to the default setting with no communication delay, we experiment with three other delay settings. The first is a fixed 30-second delay. The second is a fixed 60-second delay. The third is a random delay within a 60-second window. Under the last setting, we create a random delay value between 0s to 60s each time that a lane-direction change is scheduled.

5.4.2 *Traffic parameters*

1. Traffic Demand: Traffic demand refers to the number of vehicles that follow the same path per minute. To determine the meaningful range of this parameter, we increased the traffic demand from a low level in the synthetic network with fixed traffic signal timing until the road segments' output flow capacity is reached, i.e., when the congestion levels are at the peak of the fundamental flow diagram [15]. The output flow capacity for a fixed lane-configuration was around 24 vehicles per minute and with increased lanes, the output flow capacity was around 32 vehicles per minute. We set the default value to 28 vehicles per path per minute. To simulate traffic with a variety of congestion levels, we vary the value of this parameter from 20 to 36 vehicles per path per minute.

2. Traffic Demand Change Interval: Traffic demand can change significantly during a short period of time. Consider two regions of a road network where one region generates more traffic while the other region generates a lower amount of traffic. After some time interval, the traffic generation rates from two regions can be interchanged, resulting a major change in traffic flow in the network. This interchanging time interval is denoted as the traffic demand change interval.

3. Total Number of Lanes in a Road Segment: We vary the total number of lanes in a road between 4-8 as most road segments in real-world networks are within this range.

4. CAV Percentage: The CAV percentage refers to the percentage of fully CAVs with respect to the total number of vehicles in the simulation. The default value for this parameter is 100%.

5. Traffic Pattern: The experiments include the four traffic patterns described earlier, which are Rush hour traffic (RH), Bottleneck traffic (BN), Mixed traffic (MX) and Random traffic (RD).

6 Experimental Results

6.1 Comparative Tests

This section presents the performance of CLLA+ and other baselines defined in Section 5 in terms of average travel time and deviation from free-flow travel time using real-world and synthetic data.

Average Travel Time: As shown in Table 2, CLLA+ achieves the best performance under all traffic conditions with the synthetic road network. LLA performs well in rush hour traffic conditions (**RH**) as it uses reinforcement learning as same as CLLA+. However, LLA performs poorly when there are bottleneck traffic links (**BN**) due to the lack of coordinating process. This trend is also observed with DLA. In addition, when traffic pattern changes frequently (as in **RD**), DLA performs poorly as it cannot estimate the demand. The baseline noLA performs poorly in **RH** traffic because it cannot change lane configurations to accommodate the traffic that is directionally imbalanced during the rush hours. However, noLA performs well in **RD** when most roads do not have directionally imbalanced traffic.

Parameter	Range	Default
Lane-configuration Aware Routing	On, Off	On
Assignment interval in CLLA+	0.5 - 5 (minutes)	1m
Assignment interval in DLA	0.5 - 5 (minutes)	4m
Lookup distance in CLLA+	0 - 8 (# edges)	7
Moving average window size for PDG	0 - 20 (minutes)	1m
Maximum number of conflicts allowed in CLLA+	0 - 8	0
Traffic demand	20-36 vehicles/path/min	28
Traffic demand change interval	5 - 30 (minutes)	20m
CAV percentage	0-100%	100%
Total number of lanes in a road segment	4-8	6
Traffic pattern	RH, BN, MX, RD	RH
Communication delay of CLLA+	0s, 30s, 60s, Random (0-60s)	0s

Table 1. Parameter Settings

Baseline	Travel Time(s)				DFFT			
	RH	BN	MX	RD	RH	BN	MX	RD
noLA	600.83	607.86	603.33	320.97	3.39	3.43	3.38	2.40
LLA	563.73	705.86	573.24	325.04	3.09	3.93	3.08	2.42
DLA	580.67	617.34	595.09	342.58	3.02	3.37	3.36	2.71
CLLA+	**517.9**	**599.69**	**514.88**	**318.76**	**2.75**	**3.30**	**2.78**	**2.39**

Table 2. Travel times and their deviation from free-flow travel time (DFFT) achieved by baselines and CLLA+ with four synthetic traffic patterns (**RH, BN, MX, RD**) in the synthetic 7x7 grid network.

Baseline	Travel Time		Congested Travel Time		Imbalanced Travel Time		DFFT	
	7am	12pm	7am	12pm	7am	12pm	7am	12pm
noLA	954.71	1007.81	1231.93	1414.07	1051.01	1554.80	9.19	12.18
LLA	915.76	938.46	1142.08	1276.99	759.84	1183.38	8.36	10.07
DLA	923.24	1002.55	1165.19	1386.44	859.37	1450.40	8.46	12.05
CLLA+	893.79	925.42	1019.34	1214.69	726.04	834.13	8.04	10.05
DTS	815.03	849.19	1325.81	1320.26	957.10	1124.46	7.65	9.43
DTS_CLLA+	**764.95**	**802.79**	**1014.94**	**1196.21**	**666.17**	**706.26**	**7.12**	**9.01**

Table 3. Travel times achieved in the up-scaled Manhattan road network using New York taxi data.

We also conduct experiments using real world taxi data in the Manhattan road network. The average travel time achieved with the up-scaled version of the road network are shown in the **Travel Time** column in Table 3. To get a deeper understanding of CLLA+'s performance travel time-wise, we introduce two metrics in addition to the Average Travel Time. First, we compute the average travel time of vehicles with a travel time that is more than 8 times[4] as long as their free-flow travel time. These vehicles move in highly congested roads. These results are shown in Table 3 in the **Congested Travel Time** column. Second, we analyse the average travel time of vehicles that are travelling in road segments in which the traffic is directionally imbalanced. This is because vehicles in this type of road segments are the ones that mostly benefit from lane-direction changes. For example, directionally imbalanced traffic normally exists during rush hours, such as when most of the commuters are traveling towards a city centre for work. These results are shown

[4]The multiplication factor of 8 is chosen since this value is valid for roughly a half of all the vehicles.

in the **Imbalanced Travel Time** column in Table 3. Traffic from two times of the day, 7 am and 12 pm, are used to represent rush hour traffic and normal traffic in Table 3.

The experiments on the up-scaled Manhattan network measure the performance of CLLA+ and 5 other solutions. First we analyze the results for the baselines which use fixed traffic signals (from row 1 to row 4 in Table 3). For these results, CLLA+ achieves an average of 4% travel time improvement compared to DLA. In traffic engineering terms, this is a significant improvement as the delay of travel time is normally a small percentage of the free-flow travel time, which is not reducible. For example, it is estimated that the delay of travel time between 10 biggest US cities varies from 1.6% of the minimum travel time to 4.4% of the minimum travel time [4]. As another example, the construction of a new road infrastructure is deemed as highly beneficial if the infrastructure can lead to a 2.5% travel time improvement for the population of an entire city [22]. The result in Table 3 also shows that CLLA+ achieves a travel time improvement of around 3% from the LLA algorithm, which highlights the importance of the coordination between RL Agents. Compared to noLA, the improvement achieved by CLLA+ is around 8%. Based on the difference in the total average travel time between CLLA+ and noLA, we can estimate the total time savings for all the vehicles. For example, as shown in Table 3, the average travel time achieved by CLLA+ is 61 seconds lower than that achieved by noLA per vehicle. As there are 11k vehicles during the simulation, the total time savings is 61*11000=671000 seconds, which is approximately 186 hours. Then we analyse the results for the same solutions (from row 1 to row 4 in Table 3) using the Congested Travel Time and the Imbalanced Travel Time. In the Congested Travel Time column in Table 3, CLLA+ achieves the lowest travel time compared to the other baselines. The travel time achieved by CLLA+ in congested situations is higher than the travel time improvement in the situation with mixed congestion levels, which can be seen in the **Travel Time** column. For example, at 12pm, the travel time improvement of CLLA+ compared to noLA is 9% in the Travel Time column and is 14% in the Congested Travel Time column. In the Imbalanced Travel Time column, CLLA+ performs the best in terms of travel time gain compared to Travel Time or Congested Travel Time. It is worthwhile to note that CLLA+ achieves around *30-45%* travel time improvement over noLA. This highlights the importance of real-time lane-allocation in rush hour type traffic conditions. Finally, the last two rows of Table 3 show the performance of road network optimization that uses dynamic traffic signals. DTS achieves a lower average travel time than the CLLA+ in both 7 am and 12 pm traffic. This is because, unlike lane-allocation, dynamic traffic signals can reduce every type of traffic condition not limited to directionally imbalanced traffic. However, we should note that the Imbalanced Travel Time for CLLA+ is much lower than that of DTS. This is because CLLA+ is a better solution for directionally imbalanced traffic compared to DTS. The goal of our research is not to compete against the traffic signal optimization but rather to propose an additional traffic management solution and therefore we combine dynamic traffic signals with CLLA+ in the last row denoted as DTS_CLLA+. DTS_CLLA+ achieves significantly lower travel times than other baselines in many cases, which shows that these two complementary road network optimization solutions (DTS and CLLA+) can work well when combined together.

We also conduct experiments on the original Manhattan network. Compared to the results with the up-scaled Manhattan network, the original network achieves similar results except that the travel time is slightly higher in some scenarios. This is due to the fact that a part of the road segments in the original network, i.e., the road segments with only two lanes, is not suitable for lane-direction changes.

In addition, we evaluate the solutions across a wide range of upsampling factors that may suit various traffic conditions. A larger upsampling factor leads to a higher level of traffic volume. The results demonstrate that CLLA+ helps reduce travel time across different upsampling factors. Table 4 shows that using CLLA+ with DTS reduces travel time from using DTS alone in all cases. When the upsampling factor increases, the travel time reduction becomes more significant. For example, the ratio of travel time reduction over the travel time achieved by using DTS alone changes from 0.3% to 17.7% as the upsampling factor increases from 1 to 20. When the upsampling factor is higher, there tends to be more congestion when using DTS alone due to the increase of traffic volume in a fixed set of lanes. CLLA+ can help balance traffic flow in different directions regardless of the level of traffic load, leading to lower travel times even with large upsampling factors.

Deviation from Free-Flow Travel Time (DFFT): The last columns in Table 2 and Table 3 show DFFT values. The results show that CLLA+ is able to achieve a lower deviation from the free-flow travel time compared to DLA and LLA. It is worth noting that in Table 3 DFFT is lower for CLLA+ and LLA compared to DLA. This is because learning-based methods (CLLA+ and LLA) can adapt faster to real-time changes compared to DLA, which works based on estimating the demand. Because noLA is unable to allocate lanes dynamically, the DFFT for noLA is higher than all other solutions in all cases in Table 3. DTS_CLLA+ achieves the lowest DFFT out of all, which further demonstrates that the traffic signal and the lane-configuration optimization can work well together.

Upsampling Factor	CLLA+ with DTS	DTS
1	561.69s	563.44s
5	620.34s	631.72s
10	768.84s	808.14s
15	1150.55s	1208.57s
20	1569.9s	1906.41s

Table 4. Average travel time in up-scaled Manhattan Road network with RH traffic pattern when the upsampling factor increases from 1 to 20.

Total Number of Lane-Direction Changes: We collect the total number of lane-direction changes from simulations on the 7 × 7 synthetic network. All the simulations simulate traffic for one hour. We compare CLLA+ against LLA and DLA in two traffic scenarios. The results are shown in Table 5.

Compared to LLA with no coordination between agents, CLLA+ decreases the number of changes by 34% as its global impact evaluation algorithm skips local changes that are not globally beneficial. Interestingly, DLA makes less changes than CLLA+ in the rush hour scenario but makes significantly more changes than CLLA+ in the random traffic scenario. This is because DLA simply makes changes based on traffic demand known beforehand without considering real-time traffic conditions. Therefore, when most of the vehicles plan to travel in one direction during rush hour, DLA sees less need to make lane-direction changes. On the contrary, when route plans show irregular traffic demand in the random traffic scenario, DLA makes significantly more changes to cope with it. Different to DLA, CLLA+ makes decisions based on real-time traffic conditions. This allows it to make necessary changes and avoid redundant ones. We should also note that in both cases CLLA+ achieves the lowest travel times among the three solutions (Table 2). In Rush Hour scenario, CLLA+ achieves 517.9s travel time while LLA and DLA achieve 563.7s and 580.7s respectively. In Random Traffic scenario, CLLA+ achieves 318.8s while LLA and DLA achieve 325s and 342.6s respectively.

	Rush Hour Traffic	Random Traffic
CLLA+	735	1149
LLA	1119	1749
DLA	626	2006

Table 5. Number of lane-directions changes made by CLLA+, LLA and DLA in two traffic scenarios.

Computation Time: In addition, we evaluate the computation time of CLLA+ and compare the results against an existing work that uses linear programming for lane-direction optimization [8]. Like CLLA+, the existing work optimizes lane directions and vehicle routes simultaneously. However, the existing solution makes a one-off plan of lane directions and vehicle routes, assuming future travel schedule of all vehicles is known beforehand, while our approach performs an iterative optimization based on real-time information at a regular time interval (1 minute in the experiments).

For fairness, we collect the maximum computation time per interval with CLLA+. To get this value, we first measure the computation times spent at different optimization rounds during a simulation. Then, among all the computation times, we pick the highest value. We use the same road network as in the existing work [8], which covers a small area of Manhattan. We run simulations with different traffic loads.

As shown in Table 6, the maximum time per interval in CLLA+ stays at 0.003 second when the number of vehicles changes from 500 to 1500. A further increase of traffic load leads to an increase of the maximum computation time. With 4000 vehicles, which is significantly high for the small simulation region, the maximum computation time per interval is only 0.011 second. As the time interval between two optimization rounds is 1 minute, the computation time is negligible. In comparison, the linear programming-based solution spends 8-400 seconds on computation when there are only 5-50 vehicles respectively [8], making it unpractical for real-time optimization. The results are not surprising as reinforcement learning has a significant advantage over classical linear programming in terms of computational efficiency.

Number of Vehicles	Maximum Time per Interval
500	0.003s
1000	0.003s
1500	0.003s
2000	0.008s
2500	0.009s
3000	0.01s
3500	0.011s
4000	0.011s

Table 6. The maximum computation time per interval in CLLA+ under different traffic loads.

6.2 Sensitivity Test

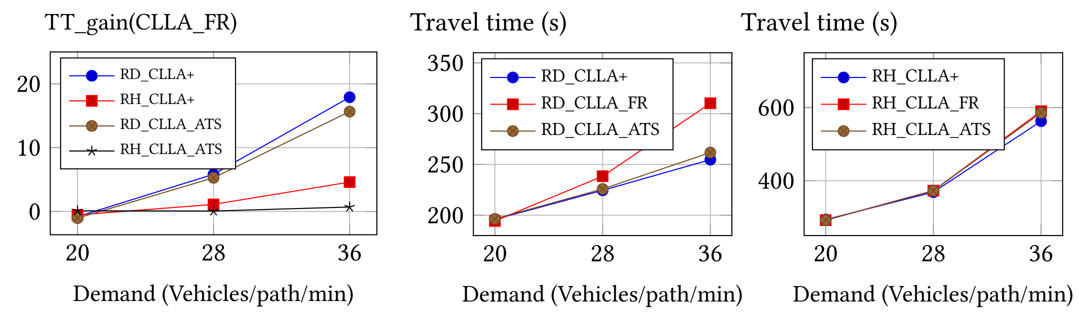
6.2.1 CLLA+ parameters

Lane-configuration Aware Routing: Figure 7a shows TT_gain(CLLA_FR) achieved by CLLA+ and CLLA_ATS on two traffic patterns, Rush hour (RH) and Random (RD), in the synthetic road network. The prefixes **RH_*** and **RD_*** in the labels in the figure indicate the simulated traffic pattern. As defined earlier, CLLA_FR refers to the solution wherein CLLA+ does not use lane-configuration aware routing, while CLLA_ATS refers to the CLLA+ that uses the average travel speed of a road segment for the rerouting matrix (i.e. weight of the segment when computing routes). In this figure, TT_gain(CLLA_FR) shows the advantage of four baselines over CLLA_FR.

CLLA+ achieves a higher TT_gain under the random traffic conditions (RD_CLLA+) compared to the rush hour (RH_CLLA+), because lane-configurations are changing more frequently in the random traffic scenario compared to the rush hour scenario. TT_gain(CLLA_FR) for CLLA_ATS is slightly lower than CLLA+. This is because, for rerouting, CLLA+ considers both the lane-configuration and traffic level, while CLLA_ATS only considers the information about the traffic level, resulting in a lower TT_gain compared to CLLA+.

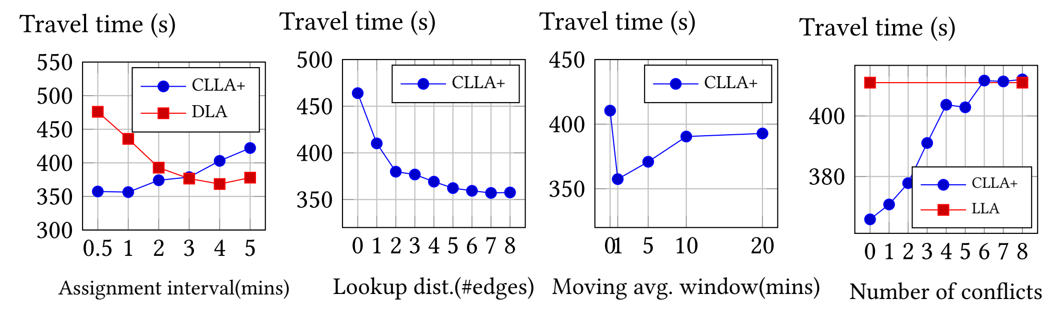
Figure 7b and 7c demonstrates the performance improvement of the lane-configuration aware routing in terms of the actual travel time for the same scenarios. With dynamic rerouting, CLLA+ achieves a substantial reduction in travel time compared to CLLA_FR (5% and 17% in the rush hour traffic and random traffic, respectively). Also, since CLLA+ considers the lane-configuration in addition to traffic conditions, CLLA+ achieves lower travel times compared to CLLA_ATS.

Assignment Interval: As this parameter affects both CLLA+ and DLA, we include both solutions in the result (Figure 8a). The figure shows the average travel time of CLLA+ and DLA, when assignment interval is changed from 0.5-minutes to 5-minutes. When the assignment interval increases, travel time achieved by DLA decreases because DLA is more likely to get a good estimation of traffic demand when the assignment interval is larger, which can lead to more effective optimizations. The default value for DLA is set to 4-minutes which corresponds to the lowest travel time over assignment interval range. The result shows that 1-minute time interval is good



(a) TT_gain over CLLA_FR (b) Travel times of CLLA variants (c) Travel times of CLLA variants

Fig. 7. Travel time improvement with lane-configuration aware routing for different traffic patterns



(a) Travel time vs assignment interval (b) Travel time improvement vs lookup distance (c) Travel time vs window size for PDG (d) Travel time vs maximum number of conflicts

Fig. 8. Sensitivity analysis for CLLA+ parameters

enough for CLLA+ to guarantee a stable performance (as the average travel time at 0.5-minutes and 1-minute is the same for CLLA+, the default value for CLLA+ is set to 1-minute to reduce the number of computations per unit time). The performance at shorter assignment intervals is more important because when real-time traffic changes frequently, the lane directions may also need to be changed frequently. Therefore CLLA+ is a better choice than DLA as it achieves a lower travel time with short assignment intervals.

Lookup Distance: Figure 8b shows the average travel time of CLLA+ when lookup distance of PDG changes from 0 to 8 (the lookup distance is measured in terms of the number of edges). The figure shows that a larger lookup distance can result in a lower average travel time. When the lookup distance increases, CLLA+ considers more road segments in a vehicle path for conflict evaluation. This helps identify more conflicting lane-direction changes on the path within the lookup distance. By resolving more conflicts, CLLA+ is able to reduce the average travel time. We should note that the reduction in the average travel time becomes less significant when the lookup distance is higher than 6. This is because the impact of a lane-direction change reduces when the change is further away.

Moving Average Window: Figure 8c shows how the average travel time of CLLA+ changes when moving average window for PDG changes from 0 to 20 minutes. When the window size is small (close to zero), only the number of vehicles of the current time step in a road segment is used as input to the CLLA+ algorithm. This results in a high variance of the measurement of traffic flow because the number of vehicles in a road segment can vary significantly across consecutive time steps. Consequently, a small window leads to a higher travel time as the lane-configurations are not based on a meaningful trend of traffic flow. On the other hand, when the moving average window is very large, e.g., 10 minutes, the moving average is unable to capture the dynamic changes in the measurement of the number of vehicles in a road segment. This also results in a higher travel time. In this experiment, the default window size is set to 1 minute to balance both high variance in traffic flow and dynamic changes in traffic load.

Maximum Allowed Conflicts: Figure 8d demonstrates how the average travel time of CLLA+ varies with the maximum conflicts threshold. The average travel time of LLA for the same scenario is shown in the figure for comparison. We can see that as the threshold increases the average travel time increases as well. When the number of maximum conflicts is set to 6 or above, the performance becomes similar to LLA that has no coordination between lane-direction changes. At higher values of this parameter, the Coordinating Agents in CLLA+ algorithm approves more changes proposed by RL Agents even if there are conflicts, which reduces the coordinated behaviour between RL Agents. In such situations, the performance of CLLA+ becomes much similar to LLA as LLA operates locally and does not take conflicts into account. The result shows a general trend that the reduction of coordination leads to an increase of average travel time.

Communication Delay: The impact of communication delay is evaluated on the 7×7 grid network. As expected, with a larger communication delay, the travel time increases (Table 7). Due to the dynamic nature of traffic, a lane-direction change optimized for a specific traffic condition can become less effective after a short period. When communication delay increases, there tends to be more significant changes in traffic conditions, reducing the effectiveness of CLLA+. We can see that a random delay within 60s is acceptable as it only increases the average travel time by 3.4% compared to the best case (with no delay).

Communication Delay	Average Travel Time
0s	414.2s
30s fixed	420.08s
60s fixed	545.41s
Random (0-60s)	428.29s

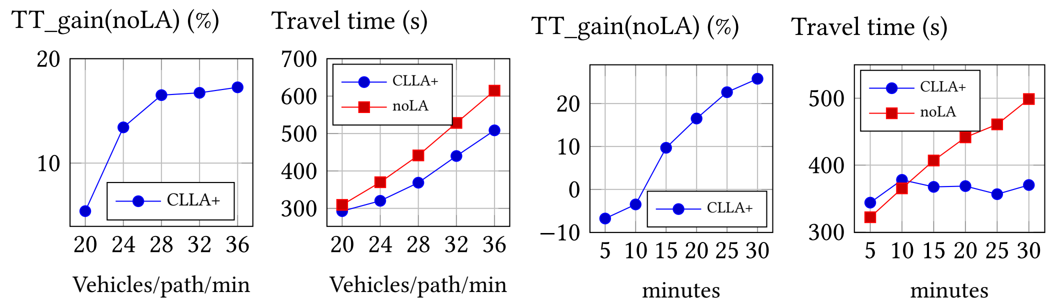
Table 7. Average travel time achieved by CLLA+ with different communication delays between the agents.

6.2.2 *Traffic parameters*

Traffic Demand: Figure 9a shows the Travel Time gain (TT_gain) achieved by CLLA+ over noLA when demand changes. When demand is low, a system with no lane-direction allocation does not observe significant traffic congestion, leading to a low TT_gain. When the demand increases, a road network with no lane-direction allocation is likely to become congested. A similar trend is shown in Figure 9b in terms of average travel time achieved by CLLA+ and noLA. When the traffic demand is low, allocating more lanes does not reduce the travel time. As the traffic demand increases the gap between travel time achieved by CLLA+ over noLA increases.

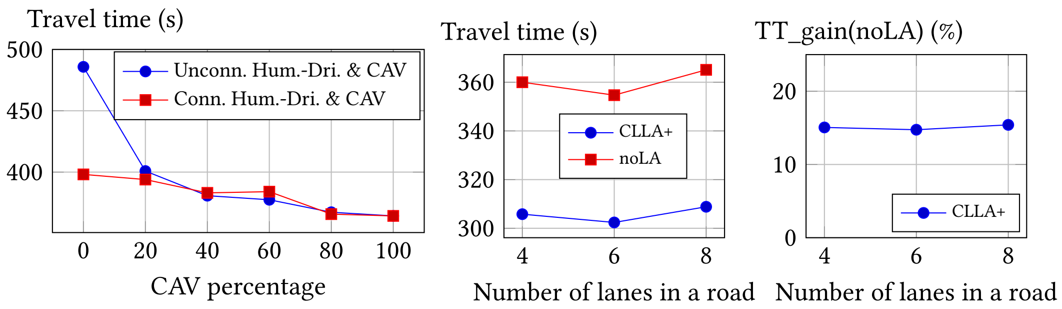
Traffic Demand Change Interval: Figure 9c shows how TT_gain changes with the traffic demand change interval. Here, the TT_gain is computed between CLLA+ and noLA. As shown in the figure, TT_gain gradually decreases when traffic demand change interval reduces from 30 minutes to 10 minutes. TT_gain becomes negative when the time interval is less than 10 minutes. The reason behind this poor performance in CLLA+ (at 10-minute interval) can be explained as follows. Due to *lane clearing time* associated with lane-direction changes, the travel time of vehicles temporally increases. When traffic demand changes frequently, *lane clearing time* becomes a major factor influencing the travel time of vehicles.

Figure 9d demonstrates the same trend in terms of the average travel time. As the traffic demand change interval increases, traffic becomes more directionally imbalanced. The average travel time



(a) TT_gain vs traffic demand (b) Travel time vs traffic demand (c) TT_gain vs traffic demand change interval (d) Travel time vs traffic demand change interval

Fig. 9. Sensitivity analysis: traffic demand change interval



(a) Travel time vs percentage of CAVs (b) Travel time vs num. lanes (c) TT_gain vs num. lanes

Fig. 10. Sensitivity analysis for CAV percentage and the number of lanes in a road

of noLA increases at the same time because it cannot adapt to the change of the traffic pattern. We should note that CLLA+ is able to keep the average travel time within a small range with various settings of the traffic demand change interval. This is because CLLA+ is able to change the lane-directions as needed in directionally imbalanced traffic.

CAV Percentage: We conduct experiments to evaluate the effectiveness of CLLA+ when CAVs are mixed with human-driven vehicles. The key difference between CAVs and human-driven vehicles is that human-driven vehicles may not be able to adapt to lane-direction changes as quick as CAVs. Therefore, in our simulations, when human-driven vehicles present in a road segment, the traffic management system does not perform lane-direction changes. We consider two types of human-driven vehicles. One is unconnected human-driven vehicles; another is connected human-driven vehicles. Unconnected human-driven vehicles do not reroute based on the latest traffic information and do not send its path information to the agents. Connected human-driven vehicles can reroute and can also send path information to the agents. The blue line in Figure 10a shows the average travel time with a mix of unconnected human-driven vehicles and CAVs. The red line in the figure shows the average travel time with a mix of connected human-driven vehicles and CAVs.

In both scenarios, a higher percentage of CAVs leads to a lower average travel time. For example, when the percentage of CAVs increases from 0% to 20%, the average travel time drops from 490s to 400s when CAVs are mixed with unconnected human-driven vehicles. With a larger percentage of CAVs, the traffic management system has a better understanding of the traffic conditions across the whole road network, resulting in more effective traffic control decisions. A difference between the two scenarios is that the average travel time with 0% CAVs is significantly lower in the connected human-driven scenario than in the unconnected human-driven scenario (there is an 18% reduction of travel time when the human-driven vehicles are changed from unconnected to connected). This shows that CLLA+ is applicable to a variety of traffic scenarios if vehicles are connected. We expect that the applicability of CLLA+ will grow significantly with the increasing connectedness of vehicles, given the fact that the market share of CAVs is growing fast [29].

Number of Lanes: We use three 7x7 grid networks in which the total number of lanes per road segment is 4, 6 and 8, respectively. Generating the same amount of traffic in different networks may not result in the same congestion levels due to the difference of the number of lanes. For instance, creating the same amount of traffic in a network with 8 lanes per road segment may lead to significantly lower travel time compared to creating the same amount of traffic in a network with 4 lanes per road segment, no matter which solution is used. To minimize the impact of this problem, the traffic generation rate is set to 8 vehicles per path per lane per minute for all three networks. Figure 10b shows that travel time does not change significantly with different numbers of lanes, meaning that the level of congestion does not change when varying the number of lanes. Figure 10c shows that the travel time improvement achieved by CLLA+ does not change significantly with different number of lanes. This demonstrates that lane-allocation is robust to the total number of lanes in a road segment and can work well with any number of lanes.

6.3 Extension of RL States

Due to the complexity of traffic systems, it may be necessary to include more RL states when optimizing lane directions. Based on our preliminary results, we select the states that have the most significant impact on the performance of CLLA+ as the default set of states for the solution. To demonstrate the possibility of extending the set of RL states, we implement two new states related to traffic signals and present the results here. The two states reflect the green time ratio at a road segment, one for upstream direction and another for downstream direction. The green time ratio is computed based on the total green time within a time window, which is set to 5 minutes. For example, let us assume that the upstream direction of a road segment was granted green light for 2 minutes during the past 5 minutes. Then the upstream green time ratio for this road segment is 2/5=0.4. To further reduce the impact of random factors, the two states are computed based on the moving average of green time ratio. We compare CLLA+ with the original set of RL states and the version with the two additional states. We run simulations on the 7×7 grid network with two traffic patterns, rush hour traffic and random traffic. The results are shown in Table 8. We can see that there is a small improvement when the traffic signal states are included. For example, the improvement is 1.5% in rush hour traffic. Like the traffic signal states, it is possible to add other states into the solution. However, as adding more states generally impacts learning efficiency significantly, it may not be worth adding states for a small gain in traffic time.

Traffic Pattern	With Traffic Signal States	Without Traffic Signal States
Rush Hour	408s	414s
Random	279s	284s

Table 8. Travel times achieved by CLLA+ with and without the additional traffic signal states.

7 Conclusion

We have shown that effective road network optimization can be achieved with dynamic lane-direction configurations. At the core of our solution is an intelligent data management and decision making mechanism. Our proposed multi-agent solution, CLLA+, can help reduce the travel time by combining adaptive learning and the global coordination of lane-direction changes. The proposed solution adapts to significant changes in traffic demand in a timely manner, making it a viable choice for realizing the potential of connected autonomous vehicles on road network optimization. Compared to the state-of-the-art solutions based on lane-direction configuration, CLLA+ runs more efficiently and is scalable to large networks. We also introduce the lane-configuration aware routing technique to account for the greedy route choice behaviour of vehicles when lane-direction changes in real-time. We have shown that our CLLA+ algorithm is able to adapt to such changes and our empirical results show that users' route changes leads to further travel time reduction. Our experimental results also demonstrate that CLLA+ can work with partially autonomous vehicles, proving that CLLA+ can be applied in real road networks in the near future.

With the advancement of intelligent traffic management technologies, one can explore innovative road network optimization solutions that involve other aspects of traffic ecosystems, such as intelligent traffic signal control systems and optimized speed limits on roads, along with lane-direction allocations. Finally, coordinating human-driven vehicles under various lane-direction configurations is another interesting direction for future research.

References

[1] Konstantinos Ampountolas, Joana Alves dos Santos, and Rodrigo Castelan Carlson. 2020. Motorway Tidal Flow Lane Control. *IEEE Transactions on Intelligent Transportation Systems* 21, 4 (2020), 1687–1696.

[2] I. Arel, C. Liu, T. Urbanik, and A.G. Kohls. 2010. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems* 4, 2 (2010), 128–135.

[3] Craig Boutilier. 1996. Planning, Learning and Coordination in Multiagent Decision Processes. In *Theoretical Aspects of Rationality and Knowledge*. 195–210.

[4] Henrik Braconier, Mauro Pisu, and Debra Bloch. 2013. The Performance of Road Transport Infrastructure and its Links to Policies. *OECD Economics Department Working Papers* 1016 (2013).

[5] Zhiguang Cao, Hongliang Guo, and Jie Zhang. 2017. A Multiagent-Based Approach for Vehicle Routing by Considering Both Arriving on Time and Total Travel Time. *ACM Transactions on Intelligent Systems and Technology* 9, 3 (2017).

[6] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. 2020. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3414–3421.

[7] Yi-Chang Chiu, Jon Bottom, Michael Mahut, Alexander Paz, Ramachandran Balakrishna, Travis Waller, and Jim Hicks. 2011. Dynamic traffic assignment: A primer. *Dynamic Traffic Assignment: A Primer* (2011).

[8] K. F. Chu, A. Y. S. Lam, and V. O. K. Li. 2017. Dynamic lane reversal routing and scheduling for connected autonomous vehicles. In *International Smart Cities Conference*. 1–6.

[9] Mark S. Daskin. 1985. Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods. *Transportation Science* 19, 4 (1985), 463–466.

[10] Samah El-Tantawy, Baher Abdulhai, and Hossam Abdelgawad. 2013. Multiagent Reinforcement Learning for Integrated Network of Adaptive Traffic Signal Controllers (MARLIN-ATSC): Methodology and Large-Scale Application on Downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems* 14, 3 (2013), 1140–1150.

[11] L. R. Ford and D. R. Fulkerson. 1958. Constructing Maximal Dynamic Flows from Static Flows. *Oper. Res.* 6, 3 (1958), 419–433.

[12] Udesh Gunarathna, Hairuo Xie, Egemen Tanin, Shanika Karunasekera, and Renata Borovica-Gajic. 2021. Real-time Lane Configuration with Coordinated Reinforcement Learning. *European Conference on Machine Learning and Knowledge Discovery in Databases* (2021), 291–307.

[13] Pengzhan Guo, Keli Xiao, Zeyang Ye, and Wei Zhu. 2021. Route Optimization via Environment-Aware Deep Network and Reinforcement Learning. 12, 6 (2021).

[14] M. Hausknecht, T. Au, P. Stone, D. Fajardo, and T. Waller. 2011. Dynamic lane reversal in traffic management. In *IEEE Conference on Intelligent Transportation Systems (ITSC)*. 1929–1934.

[15] Mehdi Keyvan-Ekbatani, Anastasios Kouvelas, Ioannis Papamichail, and Markos Papageorgiou. 2012. Exploiting the fundamental diagram of urban networks for feedback-based gating. *Transportation Research Part B: Methodological* 46, 10 (2012), 1393 – 1403.

[16] Sangho Kim, Shashi Shekhar, and Manki Min. 2008. Contraflow transportation network reconfiguration for evacuation route planning. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 20, 8 (2008), 1115–1129.

[17] Ekkehard Köhler, Rolf H. Möhring, and Martin Skutella. 2009. Traffic Networks and Flows over Time. *Algorithmics of Large and Complex Networks: Design, Analysis, and Simulation. Lecture Notes in Computer Science* (2009), 166–196.

[18] Laurence Lambert and Brian Wolshon. 2010. Characterization and comparison of traffic flow on reversible roadways. *Journal of Advanced Transportation* 44, 2 (2010), 113–122.

[19] Michael W Levin and Stephen D Boyles. 2016. A cell transmission model for dynamic lane reversal with autonomous vehicles. *Transportation Research Part C: Emerging Technologies* 68 (2016), 126 – 143.

[20] Fanyu Meng, Sze Chun Wong, W. Wong, and Y.C. Li. 2017. Estimation of scaling factors for traffic counts based on stationary and mobile sources of data. *International journal of intelligent transportation systems research* 15 (2017), 180–191.

[21] N. Mitrovic, I. Dakic, and A. Stevanovic. 2020. Combined Alternate-Direction Lane Assignment and Reservation-Based Intersection Control. *IEEE Transactions on Intelligent Transportation Systems* 21, 4 (2020), 1779–1789.

[22] Carlos A Moncada, Santiago Cardona, and Diego A Escobar. 2018. Saving travel time as an urban planning instrument. Case study: Manizales, Colombia. *Modern Applied Science* 12, 6 (2018), 44–57.

[23] United States Department of Transportation. 2022. TI04: Infrastructure-Provided Trip Planning and Route. https://www.arc-it.net/html/servicepackages/sp163.html. Accessed: 2023-02-01.

[24] Srinivas Peeta and Hani S Mahmassani. 1995. System optimal and user equilibrium time-dependent traffic assignment in congested networks. *Annals of Operations Research* 60, 1 (1995), 81–113.

[25] Kotagiri Ramamohanarao, Hairuo Xie, Lars Kulik, Shanika Karunasekera, Egemen Tanin, Rui Zhang, and Eman Bin Khunayn. 2016. SMARTS: Scalable Microscopic Adaptive Road Traffic Simulator. *ACM Transactions on Intelligent Systems and Technology* 8, 2, Article 26 (2016), 22 pages.

[26] N. R. Ravishankar and M. V. Vijayakumar. 2017. Reinforcement Learning Algorithms: Survey and Classification. *Indian Journal of Science and Technology* 10, 1 (2017), 1–8.

[27] Richard S. Sutton and Andrew G. Barto. 1998. *Introduction to Reinforcement Learning* (1st ed.). Vol. 135. MIT Press.

[28] David Alexander Tedjopurnomo, Zhifeng Bao, Baihua Zheng, Farhana Choudhury, and AK Qin. 2020. A survey on modern deep neural network for traffic prediction: Trends, methods and challenges. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* (2020).

[29] E. Uhlemann. 2016. Connected-Vehicles Applications Are Emerging [Connected Vehicles]. *IEEE Vehicular Technology Magazine* 11, 1 (2016), 25–96.

[30] Christopher JCH Watkins and Peter Dayan. 1992. Technical Note: Q-learning. *Machine learning* 8 (1992), 279–292.

[31] Stephan Winter. 2002. Route Specifications with a Linear Dual Graph. In *Advances in Spatial Data Handling*. 329–338.

[32] J. J. Wu, H. J. Sun, Z. Y. Gao, and H. Z. Zhang. 2009. Reversible lane-based traffic network optimization with an advanced traveller information system. *Engineering Optimization* 41, 1 (2009), 87–97.