GeoInformatica manuscript No.
(will be inserted by the editor)

Route Intersection Reduction with Connected Autonomous Vehicles

Sadegh Motallebi · Hairuo Xie · Egemen Tanin · Jianzhong Qi · Kotagiri Ramamohanarao

Received: date / Accepted: date

Abstract A common cause of traffic congestions is the concentration of intersecting vehicle routes. It can be difficult to reduce the intersecting routes in existing traffic systems where the routes are decided independently from vehicle to vehicle. The development of connected autonomous vehicles provides the opportunity to address the intersecting route problem as the route of vehicles can be coordinated globally. We prototype a traffic management system for optimizing traffic with connected autonomous vehicles. The system allocates routes to the vehicles based on streaming traffic data. We develop two route assignment algorithms for the system. The algorithms can help to mitigate traffic congestions by reducing intersecting routes. Extensive experiments are conducted to compare the proposed algorithms and two state-of-the-art route assignment algorithms with both synthetic and real road networks in a simulated traffic management system. The experimental results show that the proposed algorithms outperform the competitors in terms of the travel time of the vehicles.

Keywords Road Networks · Traffic Management Systems · Streaming Traffic Data · Route Assignment · Shortest Path

1 Introduction

A common cause of traffic congestions is the concentration of intersecting vehicle routes at road junctions [7, 29]. The intersection of the routes can lead to long waiting times of the vehicles at junctions. The problem caused by intersecting routes can be particularly severe in metropolitan areas, where traffic congestions are at their worst [25]. In our view, the main reason for

S. Motallebi [✉] E-mail: s.motallebi@student.unimelb.edu.au · H. Xie E-mail: xieh@unimelb.edu.au · E. Tanin E-mail: etanin@unimelb.edu.au · J. Qi E-mail: jianzhong.qi@unimelb.edu.au · K. Ramamohanarao E-mail: kotagiri@unimelb.edu.au
University of Melbourne, Australia

the concentration of intersecting routes is that vehicle routes are generated without coordination. For example, many vehicle drivers prefer to follow the shortest path from source to destination without considering that the path may intersect with the routes of other vehicles. It is difficult to solve the intersecting route problem with existing traffic systems, where vehicle drivers decide their routes independently. Although the impact of intersecting routes can be mitigated with innovative junction designs [1,3], the transformation of an existing junction normally requires significant financial input and time. In addition, due to spatial constraints or heritage conservation, it may not be possible to alter a junction.

Fortunately, the rise of *connected autonomous vehicles (CAVs)* provides a great opportunity to address the problem. It is estimated that over 85% vehicles sold on the automobile market will be *connected autonomous vehicles (CAVs)* within two decades [2,27]. The rapid growth of CAVs is encouraging innovation of traffic management solutions [22], which can bring significant benefits, such as the improvement of traffic efficiency [19] and the reduction of fuel consumption [21]. In this work, we focus on reducing route intersection with CAVs. Compared to human-driven vehicles, CAVs have two significant advantages in addressing the intersecting route problem. First, CAVs can report their trip information, such as the source and destination of a trip, to a central *traffic management system (TMS)*. CAVs can also report detailed traffic data to the TMS at real time. By collecting the information from all the vehicles, the TMS can accurately predict traffic demands and traffic conditions over a road network, allowing route planning that optimizes system-wide traffic efficiency. The second advantage of CAVs is that they can follow the exact routes given by the TMS without making unpredictable changes to the routes during their trips. This makes the traffic optimization highly effective as the traffic can evolve as planned.

The architecture of a prototype TMS is illustrated with Figure 1. CAVs and a route allocator are connected with each other through a communication infrastructure. Real-time traffic data, such as the travel time at a road link, are continuously fed into the route allocator. Upon receiving a navigation request from a CAV, the route allocator assigns a route to the vehicle such that the vehicle will avoid the areas with a large number of intersecting routes. As all the routes are allocated in this way, the chance of traffic congestions caused by the intersecting routes is minimized across the whole network.

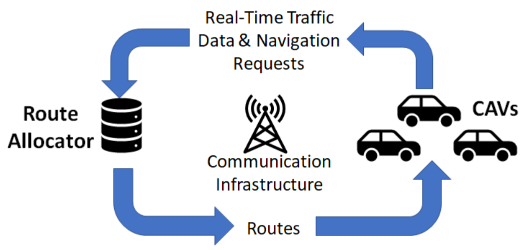
Real-Time Traffic Data & Navigation Requests
Route Allocator
Communication Infrastructure
CAVs
Routes

Fig. 1: The architecture of a TMS for traffic optimization with CAVs based on route allocation

In order to reduce the impact of intersecting routes, a future TMS needs to coordinate the route of CAVs based on innovative route allocation algorithms. Figure 2 illustrates the effects of an ideal route allocation algorithm. The left sub-figure shows three suboptimal routes with two route-intersections. As vehicles may have to wait at the junctions with intersecting routes, there can be considerable delay of traffic flow going through the junctions. The right sub-figure shows the altered routes. There is no intersecting route and the overall length of the routes does not increase significantly from the previous case. As a result, the travel time of the vehicles can be reduced from the previous case.

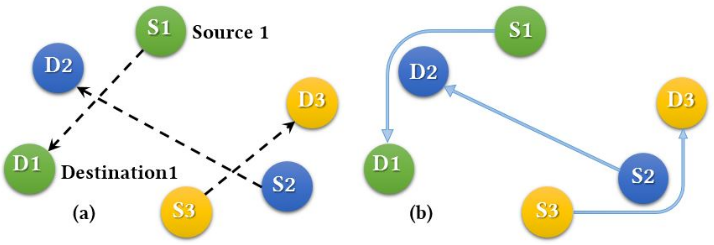

Fig. 2: (a) Intersecting routes between three pairs of source-destination, S1-D1, S2-D2, and S3-D3. (b) Routes between the same source-destination pairs without intersections. Routes are shown on Euclidean space for the sake of simplicities

We propose two route allocation algorithms. Due to the NP-hardness of traffic optimization problems [14], our algorithms use heuristics for efficient computation. The two algorithms differ in the scope of traffic information that is used for searching the routes. The first algorithm, *Local Detour Algorithm (LDA)*, uses traffic information that is confined to specific road links or road junctions, which are explicitly explored during route allocation. The second algorithm, *Multiple Intersection Reduction Algorithm (MIRA)*, enlarges the scope of traffic information by using a global view of traffic conditions based on a heatmap, which is constructed based on a grid partitioning of the road network. The heatmap cells contain the average travel time in the corresponding city blocks. With this heatmap, new routes can bypass entire city blocks that are affected by intersecting routes. This work is extended from our previous study on streaming route allocations [17].

The contributions of our work are summarized as follows.

– We show a coordinated traffic system architecture with connected autonomous vehicles that can lead to significant improvements in traffic management.
– We define a route optimization problem based on the streaming traffic data provided by CAVs.
– We propose two dynamic route assignment algorithms, LDA, and MIRA, which can address the optimization problem by reducing the intersections among the allocated routes.
– We evaluate the proposed algorithms with a prototype traffic management system based on traffic simulation. Our experiments are conducted with a synthetic road network and a real road network against two state-of-

the-art route assignment algorithms. The experimental results show that MIRA returns the routes with the best travel times in most situations. Compared to other algorithms, MIRA also does a better job in avoiding gridlocks in the road networks. The results show that MIRA is a ready-deployable algorithm if traffic congestion information is available for the route allocator.

This work is extended from our previous work [17] in several aspects. First, we formally define a streaming route assignment problem that is important in advanced traffic management. Second, we show the details of two algorithms for route intersection reduction. Third, we show the details of a prototype traffic management system that uses the proposed algorithms in traffic optimization. Fourth, the experiment section is extended comprehensively by including the results on the effects of stochastic delays of queries, the proportion of CAVs and human-driven vehicle, and the batch size of queries. The extended experiments also compare the computation times of the algorithms.

2 Related Work

Route assignment concerns the generation of routes for a large number of vehicles. Route assignment based optimization algorithms can be oriented toward an equilibrium or a disequilibrium. In an equilibrium state, any change to the existing routes will not bring benefit to the traffic. Equilibrium can be further divided into user-equilibrium [28] and system-equilibrium [4]. In a user-equilibrium state, all the users with the same source-destination pair will be assigned the same route. Differently, system-equilibrium requires the minimization of the total travel time of all the users. The users with the same original-destination pair may be assigned different routes. A comparison of the two equilibrium types shows that system-equilibrium can lower the total travel time of all vehicles with a slight increase of the total travel distance [5]. Disequilibrium-oriented route assignment does not aim to optimize traffic based on the full knowledge of traffic conditions and traffic demand [6]. Instead, it is focused on the evolution of route choices based on the users' experience and their limited knowledge of traffic conditions and traffic demand. Our work aims to achieve system-equilibrium.

Route assignment can be based on static settings or dynamic settings. Under static settings, traffic flow is assumed to be constant when computing all the routes [16]. Static route assignment is suitable when the traffic flow is nearly constant during a period. Dynamic route assignment, on the other hand, is suitable when traffic conditions change frequently [6,11]. Unlike static route assignment, dynamic route assignment needs to consider the time-varying traffic conditions that affect the vehicles during their travel. To reach equilibrium in dynamic route assignment, solution procedures may be needed such that the system basically becomes a single global navigation system for all the vehicles [23,24]. We focus on dynamic route assignment. However, we do not use iterative procedures due to the high computational costs of the procedures.

There are route assignment methods based on traffic diversification. These methods use certain random factors in route assignment to prevent traffic congestions. Nguyen et al. [18] propose an algorithm that can reduce the average travel time by more than 30% while keeping the routes close to the shortest routes. The algorithm aims to diversify traffic at junctions but it does not consider the intersection of routes. Another algorithm, Self-Adaptive Interactive Navigation Tool (SAINT), can also help to reduce travel time significantly [15]. For a specific source-destination pair, SAINT creates k candidate routes, which can lead to the minimum increase of congestion levels. Then, the algorithm tries to assign different candidate routes to different vehicles that will travel from the source to the destination. Zhang et al. [30] propose an approach, named DIFTOS, which diversifies traffic based on congestion predictions. For a given pair of source and destination, the approach first computes a shortest path. Then, all the road links on the path are checked against the existing routes that pass through the links. If a traffic congestion would happen at a link on the new route, a reroute is performed such that the route will bypass the link. DIFTOS needs to maintain a data structure for checking the traffic load of individual road links at different time slots. Due to this reason, it can be more expensive than other approaches in terms of computation time and storage space. Among these methods, SAINT is the best approach for comparison with our algorithms. We compare the proposed algorithms with SAINT in the experiments.

It is a common goal for route assignment methods that the routes are not significantly deviated from shortest paths. There are many algorithms for computing shortest paths, such as Dijkstra's algorithm [10], A* algorithm [12], and Overdo A* algorithm [13]. Shortest paths can also be computed in different ways, e.g., using time-dependent edge weights, which means the weights can change depending on the time that a vehicle will arrive at the edges [9].

3 Problem Definition

We consider route assignments for vehicles moving in a road network, which can be represented as a directed graph, $G = (V, E)$, where V is the set of vertices (road junctions or road ends) and E is the set of edges (road links). A vehicle submits a navigation request to a traffic management system when it is ready to start a trip in the road network. The navigation request contains a source s and a destination d. The traffic management system then assigns a route r to the vehicle. We assume that all the vehicles follow the assigned routes as they are CAVs.

The travel time of a vehicle with route r_i can be delayed by another vehicle with route r_j. One major factor that causes travel time delays is the existence of intersecting routes, which was not considered in aforementioned earlier work. We say that two routes r_i and r_j intersect at a common vertex v, i.e., $v \in r_i \cap r_j$, if the routes cross v from two *conflicting* approaches. Two approaches conflict if the traffic from only one of the approaches can pass the junction at any time. Travel time delays can also be related to the traffic load of a road. When the traffic load exceeds the capacity of the road, the flow speed of the road drops, resulting in a longer travel time. The effect of one vehicle on the travel time of another vehicle can be generalized by a **delay function** $\epsilon(r_i|r_j)$, which is the travel time delay added to a vehicle with route r_i due to the presence of a vehicle with route r_j. Currently we can model this function empirically but it would be interesting to mathematically model this function from basic principles. We also note $\epsilon(r_i|r_j) = 0$ when $r_i = r_j$ as the vehicle will not affect itself in travel time. The epsilon function can yield a non-zero value when i and j are different. That is, the existence of a vehicle can cause a certain level of delay to the travel time of another vehicle.

Assuming vehicles with routes in set R' are affected by vehicles with routes in set R, the **delayed travel time** of the former set of vehicles can be denoted as $DTT(R'|R)$. The shortest travel time of a vehicle with route r is $DTT(\{r\}|\emptyset)$, which is achieved when the road network is empty of vehicles. Assuming n other vehicles are already in the road network when the vehicle starts its trip, that is $R = \{r_1, r_2, ..., r_n\}$, the travel time of the vehicle can potentially be affected by all the n vehicles. This can be modelled using the following equation.

$$DTT(\{r\}|R) = DTT(\{r\}|\emptyset) + \sum_{j=1}^{n} \epsilon(r|r_j) \tag{1}$$

Streaming Route Assignment Problem Given the travel times on the edges in a road network graph, a set of existing routes and a source-destination pair, the problem asks for a route between the source and the destination such that the total travel time of all existing vehicles is minimized. The problem is different from other route assignment problems in that the travel times on the edges and the set of existing routes can change in a streaming fashion. Assuming $\mathcal{R}_{candidate}$ is the set of all candidate routes for a source-destination pair and n is the total number of routes, the optimal route $r*$ can be defined as follows.

$$r* = \operatorname*{arg\,min}_{r \in \mathcal{R}_{candidate}} \; DTT(\{r\}|\emptyset) + \sum_{j=1}^{n} \epsilon(r|r_j) \tag{2}$$

Similar to many existing route assignment problems, the streaming route assignment problem is also computationally intractable for immediate use. Our proposed algorithms do not aim to find the optimal routes. Instead, the algorithms find the routes that effectively help to reduce traffic congestions based on heuristics. The algorithms are suitable for handling streaming navigation requests and they are computationally inexpensive.

4 Streaming Route Assignment Algorithms

We propose two route assignment algorithms that work with streaming traffic data. The algorithms can be used in a traffic management system that receives navigation requests from users and assigns routes to the users. We assume that the users will follow the allocated routes. Our algorithms can work with three types of streaming data. The first is the source and the destination of vehicle trips. A vehicle submits the information when it is ready to start a trip. Based on the source-destination pairs, the algorithms generate routes. A newly-generated route will be stored as it may affect the generation of routes in the future. The second type of streaming data is the travel times on individual road links. When a vehicle passes through a road link, its travel time spent on the link is reported to the system. The average travel times on road links are updated periodically so they can show the most recent traffic flow conditions. The third type of streaming data is the updated routes. When a vehicle reaches the end of a road link, the vehicle's route is shortened as the link is removed from the route. The traffic management system considers the shortened route for generating routes in the future because there can be less intersecting routes and lower traffic load due to the change of the route.

The first algorithm, LDA, aims to avoid intersecting routes at individual junctions, which are explicitly explored during the search for a route. The second algorithm, MIRA, considers traffic information in a significantly larger scope by using a heatmap of travel times at the city level. The heatmap reflects the impact of intersecting routes in different city blocks. MIRA can help vehicles detour around entire city blocks affected by intersecting routes, allowing the discovery of routes at the global level, the level that is beyond individual road links and intersections. MIRA suggests longer detours instead of focusing on avoiding individual intersections. The reason is the surrounding area of a congested intersection tend to be affected by the congestion. Therefore, one can reduce the impact of the congestion by detouring the whole affected area.

Both algorithms use a **reservation graph** that maintains the count of routes at road links. LDA uses the graph to check the existence of intersecting routes. MIRA uses the graph to predict the traffic load at road links. The graph is same to the road network graph except that it keeps a **reservation count** at each edge of the road network. The count keeps the number of routes passing through the edge. By allocating a route, we increase the reservation count of all its road segments by one. A reservation count decreases by one when a vehicle leaves the corresponding road link. Due to the dynamic nature of traffic, reservation counts can become very different between road links over time, even if they are incremented at the same time initially.

An example of the reservation graph is shown in Figure 3. As shown in the left sub-figure, at time t_1, there are two vehicles (V_1 and V_2) travelling on two intersecting streets. Both vehicles are going to pass E. The right sub-figure shows the situation at time t_2. As vehicle V_1 has passed E, the reservation count on $e_{A,E}$ changes from 1 to 0. Similarly, we can see the change of reservation count at other edges. As we assume that time plays no role, the reservation

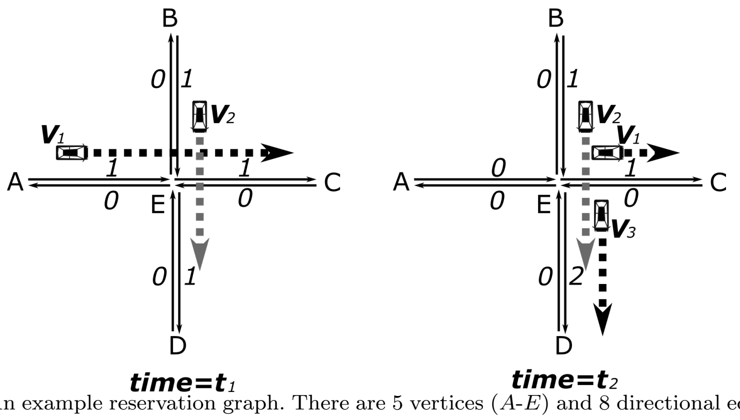

Fig. 3: An example reservation graph. There are 5 vertices (A-E) and 8 directional edges. A, E, and C are located on one street. B, E, and D are located on another street. Reservation counts are shown at the corresponding edges. The routes of the vehicles (V_1, V_2, and V_3) are shown as dashed lines. The left sub-figure shows the graph at time t_1. The right sub-figure shows the graph at a later time t_2.

graph may not provide a correct estimation of the intersecting routes and predicted traffic loads. For example, two vehicles with intersecting routes may arrive at a common junction at different times. If the time gap between the arrival of the two vehicles is significantly large, it is unlikely that one vehicle needs to wait for another at the junction. However, the impact of the estimation error on the results is limited when a large number of vehicles are considered.

4.1 Local Detour Algorithm (LDA)

The intuition behind LDA is that the travel time of a vehicle can be reduced if the vehicle spends less time waiting for conflicting traffic at road junctions. LDA (Algorithm 1) is based on A* algorithm [12], which considers a heuristic function in addition to the minimal travel cost from the source during the search for shortest paths. In order to detour junctions used by vehicles with intersecting routes, LDA uses a heuristic function based on a delay-value caused by intersecting routes. With the incorporation of the delay-value, the resultant route is less likely to intersect with the existing routes.

The details of LDA is shown in Algorithm 1. For any edge $e_{m,n}$ in the road network graph $G(V, E)$, there is a $Weight_{m,n}$, which is the average travel time on the edge. The weight is periodically updated by the traffic management system based on the travel times reported by CAVs. When two or more routes intersect at a junction, there is a conflict between the edges used by the routes. In Line 15, LDA identifies the edges that conflict with edge $e_{m,n}$. This step is done with the help of the reservation graph. An edge e conflicts with edge $e_{m,n}$ if all the following conditions are met: a) e ends in n; b) e and $e_{m,n}$ are on different roads, e.g., they have different street names; c) the reservation count of e in RG is equal to or above 1. For example, let us assume that we want to find the edges that conflict with $e_{B,E}$ at time t_1 using the scenario shown

Algorithm 1 Local Detour Algorithm

```
Input: Road network graph $G(V, E)$, reservation graph $RG(V, E)$, source $s$, destination $d$,
    and intersection factor $\alpha$. For any edge $e_{m,n}$ in $G$, there is a corresponding edge $e_{m,n}$
    in $RG$.
Output: Route $r$ from $s$ to $d$
 1: // Vertices in $Q$ are always sorted based on the travel cost between $s$ and the vertices.
 2: $Q \leftarrow$ Empty-Priority-Queue()
 3: for $m \in V$ do
 4:     $cost_m \leftarrow \infty$
 5:     $m.previous \leftarrow NIL$
 6:     $Q.insert(m)$
 7: end for
 8: $cost_s \leftarrow 0$
 9: while $Q$ is not empty do
10:     $m \leftarrow$ vertex in Q with the lowest cost to $s$
11:     remove $m$ from $Q$
12:     if $m = d$ then
13:         break;
14:     end if
15:     for $n \in$ End points of the edges starting from $m$ do
16:         $CE \leftarrow$Edges conflicting with $e_{m,n}$
17:         $Delay_{m,n} \leftarrow 0$
18:         if $|CE| > 0$ then
19:             $MATT \leftarrow$ Maximum average travel time in $CE$
20:             $Delay_{m,n} \leftarrow \alpha MATT$
21:         end if
22:         if $cost_n > cost_m + Weight_{m,n} + Delay_{m,n}$ then
23:             $cost_n \leftarrow cost_m + Weight_{m,n} + Delay_{m,n}$
24:             $n.previous \leftarrow m$
25:         end if
26:     end for
27: end while
28: $m \leftarrow d$
29: $L \leftarrow$ Empty-Linked-List()
30: $L.append(m)$
31: while $m \neq s$ do
32:     $m \leftarrow m.previous$
33:     $L.append(m)$
34: end while
35: Reverse $L$
36: Return $L$
```

in the left sub-figure of Figure 3. There are two edges, $e_{A,E}$ and $e_{C,E}$, which end in E and are on a different street than $e_{B,E}$. However, as the reservation count for $e_{A,E}$ is 1 but the reservation count for $e_{C,E}$ is 0, only $e_{A,E}$ can be regarded as a conflicting edge.

LDA uses a delay-value as an additional travel cost of an edge (Line19). The delay value is computed by multiplying an *intersection factor* α and the maximum average travel time on the conflicting edges, $MATT$ (Equation 3). Given an edge $e_{m,n}$, the $MATT$ is obtained by checking the weight (average travel time) of the edges that conflict with $e_{m,n}$. The maximum weight is set as the $MATT$ (Line 19). A vehicle tends to experience more delay at an intersection when there are more vehicles that need to cross the intersection from conflicting approaches. As the travel time of a link generally increases when there are more vehicles on the link, MATT is defined based on the maximum travel time on conflicting links to reflect the most significant impact from the conflicting traffic. The delay of a vehicle can also be affected by the traffic load of the vehicle's own road link. It would be interesting to consider the factor in addition to the impact from the conflicting traffic. We will consider the factor in our future work as we focus on the impact of intersecting routes in this study.

$$Delay = \alpha MATT \tag{3}$$

A longer average travel time on a road link generally implies that the traffic load at the link is higher. With an increase of the traffic load on conflicting edges at a road junction, the number of conflicting routes increases. As a result, a vehicle tends to spend a longer time waiting at the road junction. This is the reason that the Delay is proportional to the maximum average travel time $MATT$. By decreasing the intersection factor, the weight is decreased, which means the travel cost between two adjacent vertices in the road network graph is lower. Consequently, the search can be expanded to more vertices. However, there can be a higher chance that a returned route intersects with existing routes. On the contrary, increasing the intersection factor can help to avoid more intersecting routes. We show the effect of this hyper parameter α in Section 5.4.1. (Although we can update the Delay in a more sophisticated way, the results only show minor improvement over the aforementioned approach. Therefore we are not reporting the results of the more sophisticated version in the paper. We use LDA as our first algorithm and design a fundamentally different one with MIRA.)

As LDA utilizes Dijkstra's search procedure but with different edge weights, it has the same time complexity and storage cost which are $O(|V|log|V|+|E|)$ and $O(|V|+|E|)$, respectively. The cost of updating the reservation graph is $O(|E|)$. The cost for updating the reservation graph with a shortened route is $O(1)$.

4.2 Multiple Intersection Reduction Algorithm (MIRA)

Similar to LDA, MIRA requires a road network graph and a reservation graph. Details of the algorithm are shown in Algorithm 2. In addition to the two graphs, MIRA uses a heatmap based on the travel times on road links. The intuition behind MIRA is that the travel time of a route can be reduced more effectively by avoiding entire city blocks that are affected by intersecting routes. Existing route assignment algorithms, such as SAINT and other diversification-based approaches, do not attempt to make long detours at the city level. They focus on not assigning traffic to the same route. The routes given by these algorithms generally go in one direction, from source to destination, with minor deviations from the shortest paths. MIRA, on the other hand, enables a higher level of flexibility of choosing travel directions on the route. This is achieved based on two data structures. The first is a heatmap that shows normalized average travel times on road links in different city blocks.

As the concentration of vehicles (and the intersections of their routes) in a city block generally leads to a relatively high average travel time in the block, the difference between adjacent heatmap values shows the direction of traffic flows at the global level. The difference between heatmap values of adjacent cells can be considered as a direction from the cell with higher heat to the cell with lower heat. As observed in the real world, road users that are stuck in a traffic congestion tend to move away from the congestion where possible. As the heatmap values are proportional to the average travel time, the difference between adjacent heatmap values shows the direction of flows at the global level. The second data structure is the reservation graph, which helps to show the direction of traffic at individual road junctions. MIRA minimizes the product of the heatmap value and the reservation count in computing routes. This not only helps vehicles to avoid a large number of intersecting routes altogether by detouring around city blocks affected by those routes, but also helps vehicles to avoid individual road junctions that are affected by intersecting routes within a block. The heatmap is periodically updated based on the latest traffic information sent from CAVs. Line 15 of Algorithm 2 computes the weight of edge $e_{m,n}$, $Weight_{m,n}$, which is computed by multiplying the heatmap value for the edge ($H_{m,n}$) with the reservation count ($RC_{m,n}$).

The heatmap is constructed by mapping the whole road network space into a grid. Cells in the grid represent rectangular blocks in a city. For each cell of the grid, the average travel time on road links is computed. As we focus on mitigating congestions in metropolitan areas, we assume that the road links in the whole grid are homogeneous, which means the road links have a similar length, capacity, and speed limit. Consequently, the difference between heatmap values can be more relevant to the difference in the number of intersecting routes than the random attributes of road links. (A non-homogeneous version can be trivially derived based on the concept of a quadtree.) The average travel time of each cell is normalized by dividing the value by the total value of all the cells. The normalized values are filled into the heatmap. The heatmap value for an edge $e_{m,n}$, $H_{m,n}$, is the value of the heatmap cell that covers the edge. If m and n are in different cells, $H_{m,n}$ is the average of the two corresponding values.

MIRA also uses Dijkstra's search procedure. So, the time complexity of MIRA is $O(|V|log|V| + |E|)$. The storage cost is $O(|V| + |E|)$. The cost of updating the reservation graph is $O(|E|)$. The cost for updating heatmap is $O(|E|)$.

4.3 Difference Between MIRA and LDA

MIRA and LDA aim to minimize traffic congestion by reducing intersections between routes, but they achieve this with different approaches. LDA tries to avoid route-intersections by suggesting detours at the road-link level. On the other hand, MIRA can suggest longer detours as MIRA can capture the traffic conditions at the global level. Dividing map into cells and updating

Algorithm 2 Multiple Intersection Reduction Algorithm

Input: Road network graph $G(V, E)$, reservation graph $RG(V, E)$, source s, destination d, and heatmap H. For any edge $e_{m,n}$ in G, there is a corresponding edge $e_{m,n}$ in RG. For any edge $e_{m,n}$ in $RG(V, E)$, there is a reservation count $RC_{m,n}$, which is the number of the routes that pass through the edge.
Output: Route r from s to d

```
1: // Vertices in $Q$ are always sorted based on the travel cost between $s$ and the vertices.
2: $Q \leftarrow$ Empty-Priority-Queue()
3: for $m \in V$ do
4:     $cost_m \leftarrow \infty$
5:     $m.previous \leftarrow NIL$
6:     $Q.insert(m)$
7: end for
8: $cost_s \leftarrow 0$
9: while $Q$ is not empty do
10:    $m \leftarrow$ vertex in Q with the lowest cost to $s$
11:    remove $m$ from $Q$
12:    if $m = d$ then
13:        break;
14:    end if
15:    for $n \in$ End points of the edges starting from $m$ do
16:        $Weight_{m,n} \leftarrow H_{m,n} \times RC_{m,n}$
17:        if $cost_n > cost_m + Weight_{m,n}$ then
18:            $cost_n \leftarrow cost_m + Weight_{m,n}$
19:            $n.previous \leftarrow m$
20:        end if
21:    end for
22: end while
23: $m \leftarrow d$
24: $L \leftarrow$ Empty-Linked-List()
25: $L.append(m)$
26: while $m \neq s$ do
27:    $m \leftarrow m.previous$
28:    $L.append(m)$
29: end while
30: Reverse $L$
31: Return $L$
```

the heatmap for each cell makes MIRA more responsive to the change of traffic conditions. We should note that MIRA considers traffic conditions at both the link level and the global level by acquiring reservation counts and heatmap values. Therefore, it can suggest more effective routes than LDA. The experiments in Section 5 confirm the advantage of MIRA.

4.4 super-MIRA (sMIRA)

There can be many variations of MIRA based on the heatmap. We describe one variation, called super-MIRA (sMIRA). In sMIRA, we update reservation counts (RCs) by heatmap values rather than a constant value as in MIRA. The algorithm increments the reservation count of an edge on a newly allocated route by the edge's corresponding heatmap value. The heatmap value is deducted from the reservation count once the vehicle with the route leaves the edge. Our experiments show that sMIRA can work even better than MIRA in certain circumstances.

5 Experiments

Our experiments compare the proposed algorithms, LDA and MIRA, against two baseline algorithms in terms of the quality of the allocated routes. The first baseline algorithm is called *First-In-First-Assigned Fastest (FIFA-Fastest)*. This algorithm computes routes based on the current travel times using Dijkstra's algorithm. The second baseline algorithm is SAINT [15] as described in Section 2, which is a state-of-the-art traffic assignment algorithm. Our experiments also compare MIRA with its variation, sMIRA.

5.1 Experimental Platform

We build a prototype traffic management system that consists of two components, a route allocator and a traffic simulator. The route allocator receives navigation queries from the simulator and computes routes based on the queries. The routes are given to the traffic simulator, which simulates vehicles based on the routes and outputs the travel times of the vehicles. The simulator used in our experiments is SMARTS [20]. The simulator performs realistic simulations based on microscopic traffic models. It has been calibrated with real traffic [20] and its behaviour accurately reflects real traffic, as evidenced by our experiments with a real traffic dataset from TomTom [26]. It also simulates adaptive traffic lights as in the real world, which adjust traffic signal timing in real time based on dynamic traffic flow conditions. The architecture of the experimental platform (Figure 4) is similar to the aforementioned system architecture (Figure 1) except that the traffic system is replaced with SMARTS simulator. Three types of streaming data are fed into the route allocator, including navigation requests, updated routes and travel times on road links.

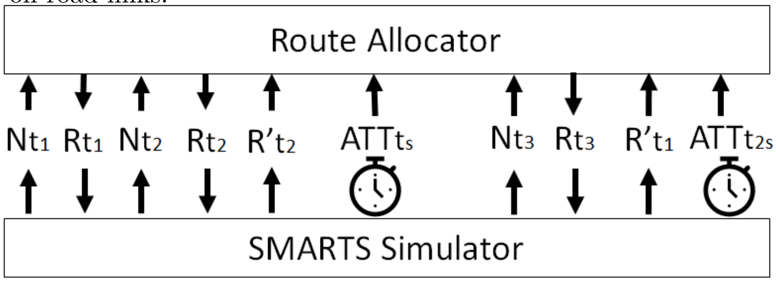

Fig. 4: Relationship between route allocator and SMARTS

Navigation requests are shown as N_{t_1}, N_{t_2}, and N_{t_3} in Figure 4. The queries are sent to the route allocator, which returns the corresponding routes, R_{t_1}, R_{t_2}, and R_{t_3}. Whenever a route is generated, it is stored by the route allocator as the new route may affect the generation of routes in the future. Whenever a simulated vehicle reaches the end of a road link, SMARTS removes the road link from the vehicle's route. The updated route is then sent to the route allocator, which replaces the old route with the updated one. The updates can happen at irregular times. For example, route R_{t_1} is assigned before route R_{t_2} but the update of the former route, R'_{t_1}, comes after the update of the later one, R'_{t_2}. SMARTS periodically reports *average travel time (ATT)* on all road links to the route allocator. ATT on a road link is calculated as $\frac{\sum_{i=1}^{n} TT_i}{n}$, where n is the number of vehicles that have passed the link since the last report and TT is the travel time that a vehicle spent on passing through the link. We should note that ATT is computed as link length divided by speed limit if there had been no vehicle passing through the link since the last report. In Figure 4, the reports are sent at time t_s and time t_{2s}, where s is the interval for reporting the information. The interval is set to 80 seconds in our experiments based on preliminary experiments and should be set based on city specific traffic conditions.

If MIRA is used as the route assignment algorithm, the route allocator creates a heatmap once it receives ATTs on all road links from SMARTS. The heatmap is based on a grid, which partitions the whole simulation area into a specified number of rows and columns. To compute the heatmap values, the route allocator first computes the average ATTs for individual grid cells. The heatmap values are then created by normalizing the average ATTs as follows. Assuming the heatmap has X rows and Y columns and the average ATT of a cell at row x and column y is $ATTC_{x,y}$, the heatmap value of the cell is computed as $\frac{ATTC_{x,y}}{\sum_{i=1,j=1}^{i=X,j=Y} ATTC_{i,j}}$. The route allocator will use the heatmap to generate routes until the next time that it receives ATTs from SMARTS.

The flowchart of experiments is shown in Figure 5, which demonstrates the relationship between the route allocator and the SMARTS simulator at experiment run-time. The figure shows that the route allocator computes a route for a generated trip query by utilizing the real-time heatmap (RTHM) and the reservation counts (RCs). Once a route is computed, the route allocator updates the RCs for the road links on the route. The SMARTS simulator updates RTHM and RCs during the simulations.

5.2 Performance Metrics

Traffic optimizations normally aim to minimize the travel time of vehicles. We identify two goals based on the reduction of travel times. The goals can be described with the delayed travel time DTT and the delay function ϵ shown in Section 3. If we divide both sides of Equation 1 by $DTT(\{r^{min}\}|\emptyset)$, the shortest possible travel time of a vehicle with the same source and destination of r, we get the ratio of the vehicle's delayed travel time to the shortest possible travel time.

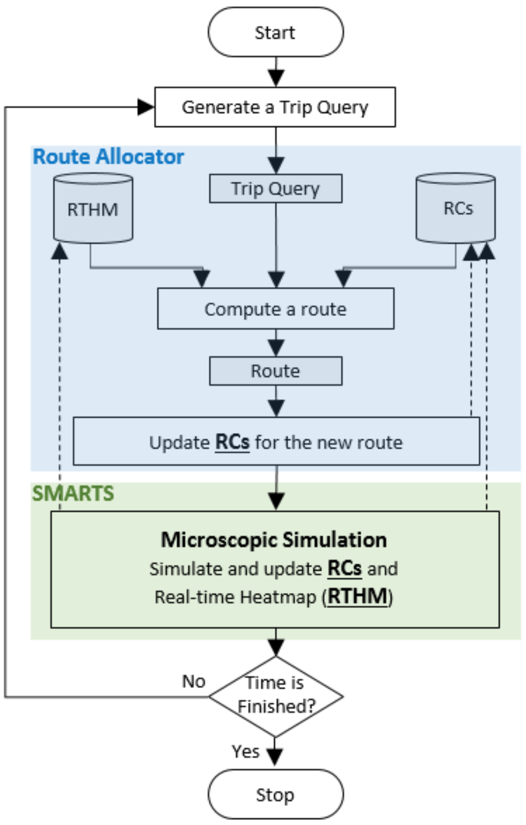

Fig. 5: Flowchart of experiments

$$\frac{DTT(\{r\}|R)}{DTT(\{r^{min}\}|\emptyset)} = \frac{DTT(\{r\}|\emptyset)}{DTT(\{r^{min}\}|\emptyset)} + \frac{\sum_{j=1}^{n}\epsilon(r|r_j)}{DTT(\{r^{min}\}|\emptyset)} \tag{4}$$

The first goal is to minimizing the average ratio for all individual vehicles.

$$min\left[\frac{1}{n}\sum_{i=1}^{n}\frac{DTT(\{r_i\}|R)}{DTT(\{r_i^{min}\}|\emptyset)}\right] = min\left[\frac{1}{n}\sum_{i=1}^{n}\left(\frac{DTT(\{r\}|\emptyset)}{DTT(\{r^{min}\}|\emptyset)} + \frac{\sum_{j=1}^{n}\epsilon(r_i|r_j)}{DTT(\{r_i^{min}\}|\emptyset)}\right)\right] \tag{5}$$

The total travel time of all the n vehicles in a road network can be represented with the following equation.

$$DTT(R|R) = \sum_{i=1}^{n} DTT(\{r_i\}|\emptyset) + \sum_{i=1}^{n}\sum_{j=1}^{n}\epsilon(r_i|r_j) \tag{6}$$

The second goal is to minimizing the ratio of the total travel time to the accumulated shortest travel times.

$$min\left[\frac{DTT(R|R)}{\sum_{i=1}^{n} DTT(\{r_i\}|\emptyset)}\right] = min\left[\frac{\sum_{i=1}^{n} DTT(\{r_i\}|\emptyset)}{\sum_{i=1}^{n} DTT(\{r_i^{min}\}|\emptyset)} + \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}\epsilon(r_i|r_j)}{\sum_{i=1}^{n} DTT(\{r_i\}|\emptyset)}\right] \tag{7}$$

Based on the two goals, we define two metrics, *Travel Time Ratio at Individual level (TTRI)* and *Travel Time Ratio at System level (TTRS)*. TTRI is defined in Equation 8, where $BTT(v_i)$ is the best theoretical travel time of the shortest (fastest) route for a vehicle v_i and $TT(v_i)$ is the actual travel time of the vehicle for the given route. The best theoretical travel time is calculated under the assumption that the road network is empty and the vehicle never encounters a red light at any junction. Therefore it is always equal to or smaller than the actual travel time. Consequently, TTRI is always equal to or higher than 1. The best value of TTRI is 1. Different to TTRI, TTRS is based on the total travel time of all the vehicles (Equation 9). TTRS is always equal to or higher than 1. The best value of TTRS is 1.

$$TTRI = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|}\frac{TT(v_i)}{BTT(v_i)} \tag{8}$$

$$TTRS = \frac{\sum_{i=1}^{|\mathcal{V}|} TT(v_i)}{\sum_{i=1}^{|\mathcal{V}|} BTT(v_i)} \tag{9}$$

We note that there is no meaningful TTRI and TTRS if there are gridlocks in the simulated traffic system. A gridlock appears once an area is so severely congested that no vehicle can make further movement. We observe that gridlocks will eventually appear when the number of vehicles is beyond a certain value, no matter which algorithm is used for routing. We call the maximum number of vehicles that exist in a network without a gridlock the **gridlock threshold**. Different algorithms can have different gridlock thresholds. A higher gridlock threshold is better as the traffic network can function with a larger number of vehicles. Our results do not show TTRI and TTRS when the number of vehicles is beyond the gridlock threshold.

5.3 Experimental Settings

We evaluate the algorithms based on a synthetic road network and a real road network. For a specific combination of settings, we first use our SMARTS simulator to generate source-destination data, which contains a number of time-stamped source-destination pairs. To generate the data, we run a one-hour simulation of peak-hour traffic. The traffic load, i.e., the number of vehicles in the network, is kept constant during the period. The starting time, source and destination of a vehicle are recorded when the vehicle starts to move in the network. Once the vehicle reaches its destination, it is replaced by a new vehicle with a random source and a random destination. The same set of source-destination data is used for experimenting any algorithm in the setting.

The experiments are conducted on a computer with an Intel Core i7 CPU running at 2.7 GHz. The computer has 16 GB of RAM. The operating system is the 64-bit version of Windows 7.

5.3.1 Synthetic Road Network

The synthetic network is based on a grid plan with two sets of streets. Each set has 12 streets that are parallel to each other. A street in one set runs at right angle to a street in another set. Traffic lights are installed at all road junctions. Adjacent junctions are connected by a two-way road segment that is 400 metres long. The speed limit for all the road segments is set to $40km/h$, which is the common speed limit for many central business areas across Australia. The default granularity of the heatmap for MIRA is 3×3 as the area can be fully divided at this granularity.

We evaluate the effects of two parameters on all algorithms. For each parameter, we run simulations with different values of the parameter while keeping the other parameter at its default value. The first parameter is the number of vehicles. The second parameter is the spatial distribution of source and destination.

Number of Vehicles This parameter can have a significant impact on travel times. As the number of vehicles increases, the number of intersecting routes increases, leading to a higher chance of traffic congestions. A good route assignment algorithm can suggest routes that lead to satisfactory travel times even when there are a large number of vehicles. We vary the value of this parameter between 1000 and 10000. The default value is 6000 because we cannot run SAINT with 7000 or more vehicles.

Source-Destination Distribution Sources and destinations are generated with uniform distribution or Gaussian distribution. If a source or a destination is generated with the uniform distribution, we randomly pick a point from the road network as the source or the destination. With Gaussian distribution, the sources and the destinations are more likely to be located around the center of the road network area. We choose four source-destination distribution scenarios: 1)Uniform-Uniform, 2)Uniform-Gaussian, 3)Gaussian-Uniform, and 4)Gaussian-Gaussian. These distributions represent various traffic situations. Uniform-Uniform represents traffic flows in off-peak hours. Uniform-Gaussian represents morning peak hours when vehicles go to the city center from the suburbs. Gaussian-Uniform represents afternoon peak hours when vehicles leave the city center to the suburbs. Gaussian-Gaussian represents the highly spatially concentrated traffic. As the Gaussian-Gaussian distribution (named as Gaussian in other experiments) shows the most significant impact of intersecting routes, it is used as the default value. Moreover, it is more realistic than the uniform distribution for the central area of a city.

5.3.2 Real Road Network

The real road network covers a $30km \times 30km$ area of Melbourne (Figure 6). The area is centred at the CBD of Melbourne. The speed limit of road segments, the number of lanes on road links and the direction of road links are extracted from OpenStreetMap (https://www.openstreetmap.org). This network has 20400 vertices and 25600 edges. To minimize the impact of variance in road links, this road network only contains freeways and arterial roads in the area. For this network, we vary the number of vehicles between 10000 and 50000. We use Gaussian distribution for generating sources and destinations. The granularity of MIRA's heatmap is set to a relatively low value, 3×3, because the density of roads is not high as we only use arterial roads. We do not include SAINT in this experiment because the algorithm cannot return routes in a manageable time based on this network. We would like to highlight that we do not directly control the flow speed, but we set the speed limit of links based on the type of the links. For arterials, the flow speed may not reach the free-flow speed (the speed limit) when the roads are congested. This can be observed in many peak-hour scenarios. Our simulator shows the fluctuation of flow speeds on all the road links unless the traffic load is very low.

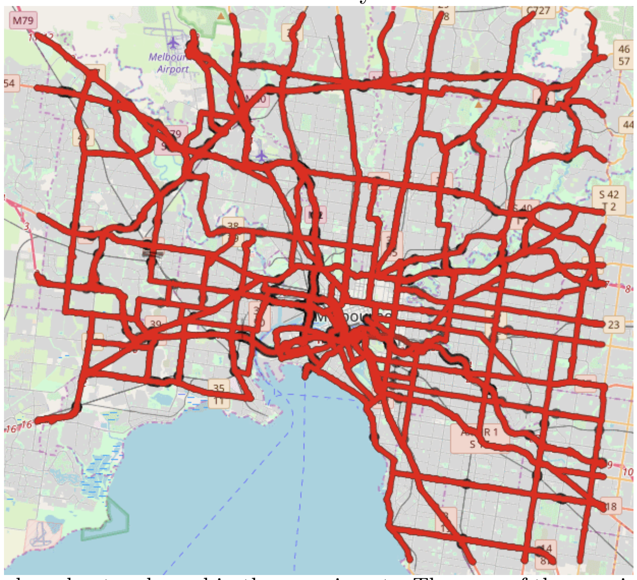

Fig. 6: The real road network used in the experiments. The map of the area is shown in the background

5.4 Results

5.4.1 Changing α for LDA

We use the synthetic network to evaluate the α parameter. The locations of sources and destinations are generated with Gaussian distribution. We test the effect of α on TTRI (Figure 7). In this test, the number of vehicles varies between 1000 and 4000. We vary the value of α between 0 and 2. We should note that LDA behaves in the same way as Dijkstra's algorithm when α is 0 because there is no time penalty for intersecting routes. As shown in Figure 7, there is no significant difference between the α values when there are 1000 or 2000 vehicles. However, when there are 3000 vehicles, the TTRIs are 4.29 for $\alpha = 0$ and 3.69 for $\alpha = 2$. TTRIs are lower than the two values when

α is 0.5, 1, and 1.5. This shows that the quality of the routes increases, i.e., TTRI drops, when α increases from 0. A positive α value can help to avoid intersecting routes but vehicles may have to make a larger number of detours. When α is beyond a certain range, 0.5-1.5 in our case, the time saved by avoiding intersecting routes may be negated by the time spent on the detours. Based on this result, we set α to 0.5 for the rest of the experiments.

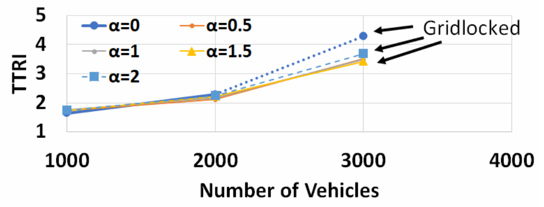

Fig. 7: Results on LDA's α. Lower TTRIs are better. Source-destination distribution is Gaussian. Gridlock threshold for all α values is 3000

5.4.2 Synthetic Network

The following results compare the effects of the number of vehicles and the source-destination distribution on the performance of four algorithms.

Number of Vehicles MIRA outperforms other algorithms except when the number of vehicles is very low (Figure 8). When the number of vehicles increases, TTRI and TTRS increase for all algorithms. This is understandable as the number of intersecting routes generally increases when there are more vehicles, leading to longer delays on the road. MIRA is the best algorithm for avoiding gridlocks. The gridlock threshold of MIRA is 9600 while the gridlock threshold of the second-best algorithm, SAINT, is 6800. As discussed earlier, MIRA is more advanced than LDA as MIRA utilizes traffic information at both the link level and the global level. Same to our expectation, MIRA's routes leads to shorter travel times compared to LDA's. The result shows that although LDA has a better performance compared to FIFA-Fastest, it does not work as well as MIRA and SAINT.

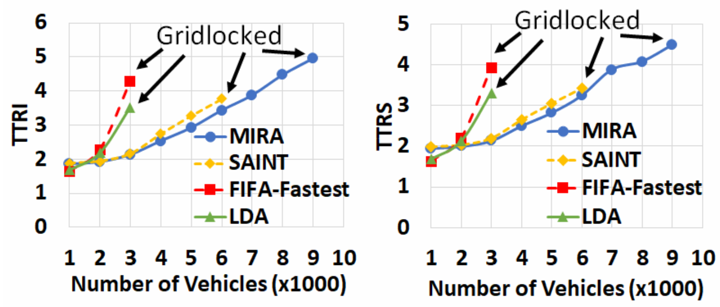

Fig. 8: TTRI and TTRS (lower values are better) achieved with the synthetic road network. LDA's α is 0.5. Source-destination distribution is Gaussian. Gridlock thresholds of MIRA, SAINT, FIFA-Fastest and LDA are 9600, 6800, 3000, and 3000, respectively

The vehicles tend to concentrate to certain sub-areas with SAINT while they are more evenly distributed around the whole road network with MIRA

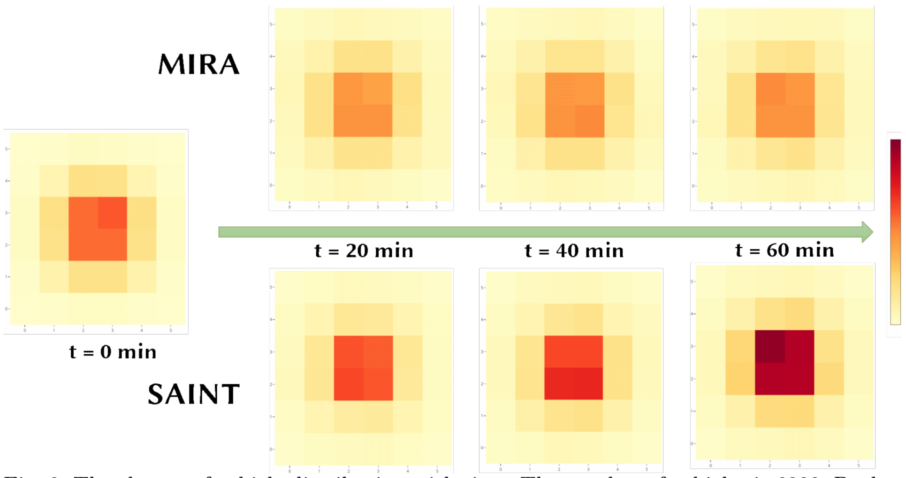

Fig. 9: The change of vehicle distribution with time. The number of vehicles is 6800. Darker areas have a higher density of vehicles. MIRA has less congestions than SAINT. Source-destination distribution is Gaussian.

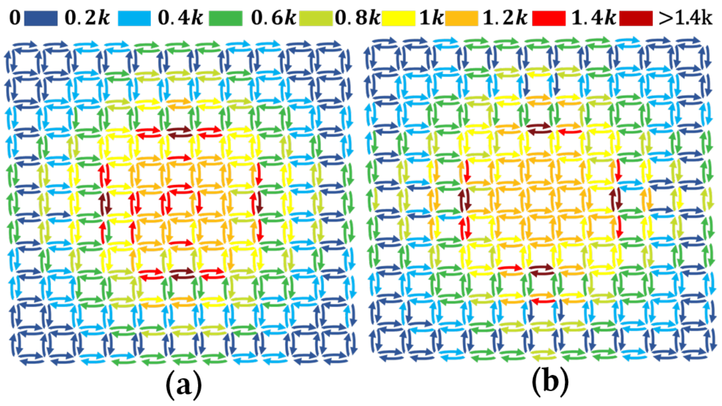

Fig. 10: Road links are coloured based on the number of routes that pass through them when the number of vehicles is 6800. Routes in (a) are created by SAINT. Routes in (b) are created by MIRA. The figure shows that there are less congested roads scheduled by MIRA than SAINT. Source-destination distribution is Gaussian.

(Figure 9). Based on the global view of traffic flow conditions, MIRA can suggest routes that bypass the sub-areas with slow traffic, leading to a better distribution of the vehicles. This can significantly reduce the chance of gridlocks. On the contrary, SAINT follows a local diversifying strategy without considering traffic flow conditions in the large scope as MIRA. This can result in more intersecting routes, leading to the concentration of vehicles. The difference between the two algorithms can also be observed in Figure 10. The figure shows the total number of routes that go through the road links. We can see that many road links in the central part of the road network are used by more than 1000 routes with SAINT. This implies that there can be a high number of intersecting routes in the area. The routes suggested by MIRA, however, do not go through the road links in the central area as much as the routes suggested by SAINT.

Source-Destination Distribution As shown in Figure 11, all algorithms perform well when the sources and destinations follow a uniform distribution. MIRA has a small advantage over other algorithms in this situation. In the morning peak hour (Uniform-Gaussian), the majority of trips are heading towards the city center. Thus, the traffic in the city center is affected by an increasing number of route-intersections. The result shows that MIRA outperforms other methods, while FIFA-Fastest and LDA reach gridlock. The results show that TTRI increases significantly for all algorithms when the distribution changes from Uniform-Uniform to other distributions. This is understandable as there is a higher probability of route intersections when the sources or destinations become clustered. At afternoon peak hour (Gaussian-Uniform), vehicles are heading towards suburbs where no traffic congestion exists. In this situation, the result indicates that MIRA, which uses a city block detour strategy, can lead to extra detours. For the Gaussian-Uniform distribution, SAINT achieves a lower TTI than MIRA. In this scenario, LDA can navigate vehicles poorly but with no gridlock. The result with Gaussian-Gaussian distribution shows that MIRA outperforms SAINT while both FIFA-Fastest and LDA suffers from gridlock.

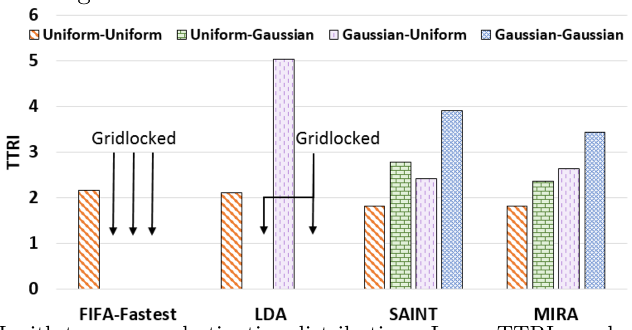

Fig. 11: TTRI with two source-destination distributions. Lower TTRIs are better. LDA's α is 0.5. Number of vehicles is 6000

5.4.3 Real Network

For the real network, MIRA achieves the best performance among all the algorithms (Figure 12). As expected, TTRI and TTRS increase for all algorithms when the number of vehicles increases. The gridlock threshold of both LDA and FIFA-Fastest is lower than that of MIRA by 10000. This shows that the global view of traffic conditions used by MIRA helps to reduce intersecting routes significantly. The gap of TTRI between MIRA and other two algorithms becomes higher when there are more vehicles in the network. For example, the gap of between MIRA and LDA is 0.136 with 10000 vehicles. The gap increases to 1.467 with 30000 vehicles. We observe a similar trend in TTRS. Compared to small road networks, such as the synthetic network shown earlier, large road networks allow vehicles to make relatively longer detours in exchange for bet-

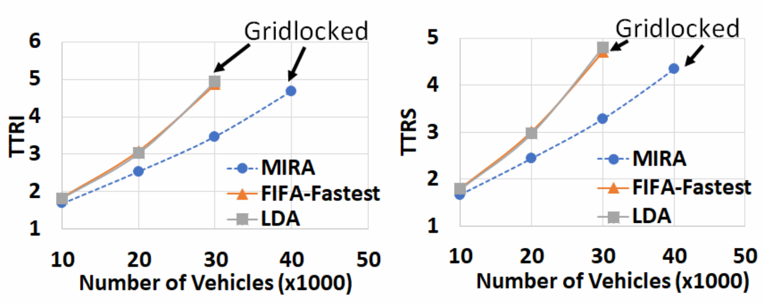

Fig. 12: TTRI and TTRS achieved with the real road network (lower values are better). LDA's α is 0.5. Source-destination distribution is Gaussian. Gridlock thresholds of LDA, FIFA-Fastest, and MIRA are 30000, 30000, and 40000, respectively. We do not include SAINT because the algorithm cannot return routes in a manageable time with this network

ter travel times. MIRA can take this advantage by using the heatmap while other algorithms cannot.

5.4.4 sMIRA versus MIRA

In this experiment, we compare MIRA and its variation, sMIRA, using a more complex traffic scenario based on the synthetic network. The source-destination pairs of 80% of the vehicles are generated with Gaussian distribution. For the rest of the vehicles, their trips start and end at random locations on the border of the area, which means they pass through the area rather than concentrating to the centre. In this experiment, the resolution of the heatmap is set to 6×6 as it works best for both algorithms in this scenario. sMIRA performs better than MIRA as sMIRA not only has a higher gridlock threshold but also achieves a lower TTRI and a lower TTRS (Figure 13). As sMIRA puts more weights on the city blocks that are currently affected by intersecting routes, it can handle this scenario better than MIRA.

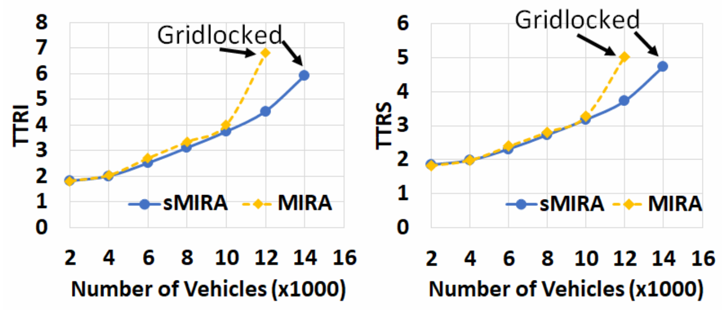

Fig. 13: TTRI and TTRS achieved by MIRA and sMIRA (lower values are better) with the synthetic network. The source-destination distribution for 80% of the vehicles is Gaussian. The source and destination of the rest of the vehicles are uniformly distributed on the border of the area. Gridlock thresholds of MIRA and sMIRA are 12000 and 14000, respectively

5.4.5 Other Considerations

We evaluate other factors that may affect the performance of MIRA. We also measure the computation time of the algorithms.

Ratio of FIFA-Fastest Routes A common problem with traffic optimizations is that vehicles may not follow the suggested routes in the real world. For example, while CAVs can follow the exact routes given by MIRA, a large number of human-driven vehicles may tend to follow the shortest paths as they are not connected with a TMS. In this test, we assume that a portion of the vehicles are not connected vehicles. Their routes are computed with FIFA-Fastest. The routes of the remaining vehicles are given with MIRA or SAINT. As shown in Figure 14, MIRA performs better than SAINT as it achieves a lower TTRI when the ratio of FIFA-Fastest routes is between 0% and 20%. SAINT leads to a gridlock when the ratio is higher than 20%. Differently, gridlock appears with MIRA when more than 40% routes are computed with FIFA-Fastest.

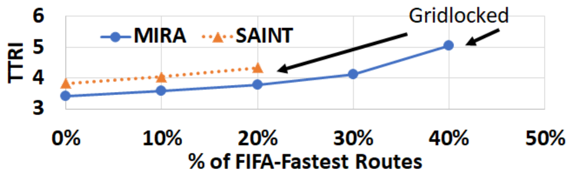

Fig. 14: TTRI achieved when a portion of the routes are computed with FIFA-Fastest. The synthetic network is used for this experiment. Lower TTRIs are better. Source-destination distribution is Gaussian. Number of vehicles is 6000. SAINT leads to gridlocks when more than 20% routes are given by FIFA-Fastest. Gridlocks happen with MIRA when there are more than 40% FIFA-Fastest routes

Query Batch Size of MIRA We also test the impact of the query batch size on the quality of the routes returned by MIRA. As the flowchart (Figure 5) shows, trip queries are generated one by one, and the system computes a route for each query. In this experiment, we do route assignment in batches to see whether optimizing routes in batches can enhance traffic efficiency in the whole network as we have more flexibility in assigning routes. We vary the number of navigation requests that are processed in a batch between 1 and 50. This may change the dynamics of route allocation as it changes the stream window size from 1 to 50. We randomize the order of the requests in each run. Although processing queries in large batches with a large stream window size may lead to certain improvement in system-wide traffic optimization, it is normally unpractical as vehicles have to wait for a long time to get routes. As shown in Figure 15, TTRI does not change significantly with different batch sizes. In other words, processing requests in small random batches does not have a significant impact on the quality of the routes. This shows that MIRA is a ready-deployable algorithm. For example, there are approximately 57,000 people who live in the inner region of Melbourne and commute to work by cars [8]. Assuming each of them makes one navigation request at a random time during the typical peak hour period in Melbourne, 6.30am to 9am, there will be approximately 7 requests per second. Based on our result, the stream window size can be set to a small value, e.g., one second, which leads to a fast response time of the application without lowering the quality of routes.

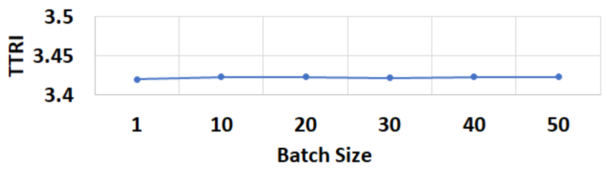

Fig. 15: TTRI achieved by MIRA with different batch size of navigation requests. The synthetic road network is used for this experiment. Source-destination distribution is Gaussian

Stochastic Delays for Route Update Messages The proposed TMS is a centralized system that continuously communicates with CAVs. The TMS needs to receive the route updates from all vehicles, which creates a large number of messages. The messages may arrive at the TMS with a certain level of delays depending on various factors such as congestions in the communication channel. We note that delays occurring before the start of driving are not considered as they do not affect traffic congestions directly. In this experiment, we examine the impact of stochastic delays of route update messages on the outcome of MIRA. We generate delays with a probability p_{delay}. When $p_{delay} = 1$, all updates are delayed, while a probability $p_{delay} = 0$ means no delay exists. If an update is delayed, the time length of the delay is generated based on a uniform distribution in a range of $[d_{min}, d_{max}]$. In this test, we consider two ranges of delay, $[1, 3]$ minute (short delay) and $[3, 20]$ minute (long delay). The probability p_{delay} varies from 0 to 1 with step size 0.25. Figure 16 shows that TTRI is almost stable for short delays and increases by 3% for long delays when p_{delay} increases from 0 to 1. So, we can conclude that MIRA is robust against stochastic delay of route updates.

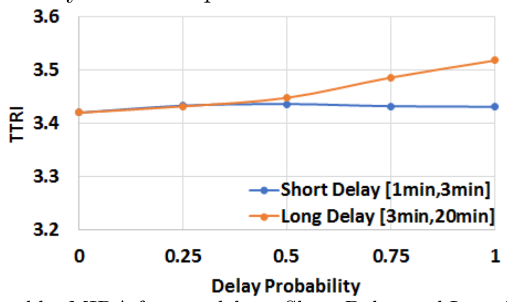

Fig. 16: TTRI achieved by MIRA for two delays, Short Delay and Long Delay, with different delay probabilities when route update messages are delayed. The synthetic road network is used for this experiment. Source-destination distribution is Gaussian

Computation Time We show a comparison between the algorithms in terms of computation time. The computation complexities of FIFA-Fastest, LDA, SAINT, and MIRA are $O(|V|log|V|+|E|)$, $O(|V|log|V|+|E|)$, $O(k|V|(|V|log|V|+E))$, and $O(|V|log|V|+|E|)$, respectively, where $|V|$ is the number of vertices, $|E|$ is the number of edges and k is a hyper parameter for SAINT, which is set to 5 as the value leads to the best performance of SAINT based on our tests. We vary the number of vertices in a synthetic road network. For each setting, we create 5 batches of navigation requests, each of which has 10 requests. The average computation time for creating one route is reported from each batch. We then average the values from the 5 batches and report it in Figure 17. The result shows that MIRA can solve a query within 0.1 second in the worst case. Based on the aforementioned application scenario with Melbourne commuters, the approximate number of navigation requests per second is 7. It will take MIRA less than 0.7 second to process the requests in the worst case, which means the processing speed can cope with the rate of incoming requests. This again shows that MIRA is a ready-deployable algorithm.

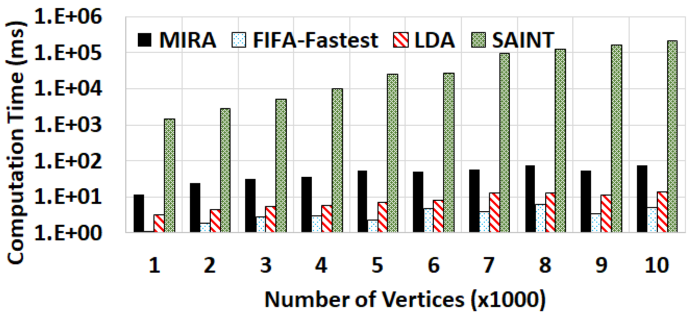

Fig. 17: Average computation time for creating one route in a synthetic road network. LDA's α is 0.5. Source-destination distribution is Gaussian. Time is measured in milliseconds. Logarithmic values of the time are shown

6 Conclusions

We have shown that using certain traffic heuristics in route allocation can help to minimize the impact of intersecting routes while keeping computation costs at a low level. We have also shown that route optimization can be more effective when it considers not only the traffic conditions at individual road links but also the difference in traffic conditions between the sub-areas of a road network. In our future work, we will take time into consideration in order to achieve even better traffic efficiency with MIRA. Also, we will consider other contributing factors to the traffic delays (e.g., queue spillback) to make the delay model more comprehensive. We hope our work can inspire more studies on traffic management with connected autonomous vehicles.

7 Conflict of Interest

The authors declare that they have no conflict of interest.

8 Acknowledgement

We thank the anonymous reviewers whose comments helped improve and clarify this study.

References

1. J. Autey, T. Sayed, and M. El Esawey. Operational performance comparison of four unconventional intersection designs using micro-simulation. *Journal of Advanced Transportation*, 47(5):536–552, 2013.
2. P. Bansal and K. M. Kockelman. Forecasting americans' long-term adoption of connected and autonomous vehicle technologies. *Transportation Research Part A: Policy and Practice*, 95:49–63, 2017.
3. J. G. Bared and E. I. Kaisar. Median U-turn design as an alternative treatment for left turns at signalized intersections. *ITE Journal*, 72(2):50–54, 02 2002.
4. M. Beckmann, C. B. McGuire, and C. B. Winsten. Studies in the economics of transportation. https://trid.trb.org/view/91120, 1956.
5. D. Boyce and Q. Xiong. User-optimal and system-optimal route choices for a large road network. *Review of network Economics*, 3(4), 2004.
6. Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, and J. Hicks. Dynamic traffic assignment: A primer. *Transportation Research Circular*, (E-C153), 2011.
7. I. Dadić, G. Kos, K. Poić, and E. Gašparac. Measures for reducing traffic congestion in cities. *PROMET-Traffic & Transportation*, 11(1):15–19, 1999.
8. H. de Silva and A. Lightfoot. Commuting to work by private vehicle in Melbourne: Trends and policy implications. In *Australasian Transport Research Forum*, 2010.
9. U. Demiryurek, F. Banaei-Kashani, and C. Shahabi. A case for time-dependent shortest path computation in spatial networks. In *SIGSPATIAL*, pages 474–477. ACM, 2010.
10. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.
11. T. L. Friesz and D. Bernstein. Analytical dynamic traffic assignment models. In *Handbook of transport modelling*, pages 181–195. Elsevier, 2000.
12. P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.
13. R. Jacob, M. Marathe, and K. Nagel. A computational study of routing algorithms for realistic transportation networks. *Journal of Experimental Algorithmics*, 4:6, 1999.
14. O. Jahn, R. H. Möhring, A. S. Schulz, and N. E. Stier-Moses. System-optimal routing of traffic flows with user constraints in networks with congestion. *Operations research*, 53(4):600–616, 2005.
15. J. Jeong, H. Jeong, E. Lee, T. Oh, and D. H. C. Du. SAINT: Self-adaptive interactive navigation tool for cloud-based vehicular traffic optimization. *IEEE Transactions on Vehicular Technology*, 65(6):4053–4067, 2016.
16. M. Lujak, S. Giordani, and S. Ossowski. Route guidance: Bridging system and user optimization in traffic assignment. *Neurocomputing*, 151:449–460, 2015.
17. S. Motallebi, H. Xie, E. Tanin, J. Qi, and K. Ramamohanarao. Streaming route assignment for connected autonomous vehicles (systems paper). In *SIGSPATIAL*. ACM, 2019.
18. U. T. Nguyen, S. Karunasekera, L. Kulik, E. Tanin, R. Zhang, H. Zhang, H. Xie, and K. Ramamohanarao. A randomized path routing algorithm for decentralized route allocation in transportation networks. In *SIGSPATIAL*, pages 15–20. ACM, 2015.
19. K. Ramamohanarao, J. Qi, E. Tanin, and S. Motallebi. From how to where: Traffic optimization in the era of automated vehicles. In *SIGSPATIAL*, pages 10:1–10:4. ACM, 2017.
20. K. Ramamohanarao, H. Xie, L. Kulik, S. Karunasekera, E. Tanin, R. Zhang, and E. B. Khunayn. SMARTS: Scalable microscopic adaptive road traffic simulator. *ACM Transactions on Intelligent Systems and Technology*, 8(2):26:1–26:22, 2016.
21. J. Rios-Torres and A. A. Malikopoulos. Energy impact of different penetrations of connected and automated vehicles: a preliminary assessment. In *SIGSPATIAL*, pages 1–6. ACM, 2016.
22. J. Rios-Torres and A. A. Malikopoulos. A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps. *IEEE Transactions on Intelligent Transportation Systems*, 18(5):1066–1077, 2016.

23. W. Szeto and H. K. Lo. Dynamic traffic assignment: properties and extensions. *Transportmetrica*, 2(1):31–52, 2006.
24. N. B. Taylor. The contram dynamic traffic assignment model. *Networks and Spatial Economics*, 3(3):297–322, 2003.
25. M. Terrill, H. Batrouney, S. Etherington, and H. Parsonage. *Stuck in traffic? Road congestion in Sydney and Melbourne*. Number 2017-10. Grattan Institute, 2017.
26. TomTom. Online navigation services. http://developer.tomtom.com, 2015.
27. Transport Systems Catapult. Market forecast for connected and autonomous vehicles. https://ts.catapult.org.uk/intelligent-mobility/im-resources/publications/, 2017.
28. J. G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers*, 1(3):325–362, 1952.
29. C. Wright, D. Jarrett, and G. Appa. Spatial aspects of traffic circulation: II. Routing patterns that exactly minimise path crossings. *Transportation Research Part B: Methodological*, 29(1):33–46, 1995.
30. W. Zhang, N. Aung, S. Dhelim, and Y. Ai. DIFTOS: A distributed infrastructure-free traffic optimization system based on vehicular ad hoc networks for urban environments. *Sensors*, 18(8), 2018.