Optimal Path Selection in a Continuous-Time Route Reservation Architecture

C. Menelaou, S. Timotheou, P. Kolios, C.G. Panayiitou and M.M. Polycarpou

Abstract—In this work, we adopt the route reservation architecture proposed in [1] and adapt the reservation algorithm so that route reservations are made based on continuous-time intervals (instead of fixed discrete-time intervals). Using the continuous-time reservation architecture, we formulate a mixed integer linear programming (MILP) problem to obtain the optimal path of any vehicle, avoiding roads that are at-capacity. Since the MILP problem is hard to solve efficiently, we also propose a heuristic iterative algorithm that is based on Dijkstra's algorithm, is of low-complexity and provides close-to-optimal results, as compared to the MILP obtained solution (at least for small problems where MILP provided a solution in reasonable time). The simulation results further indicate that the Dijkstra-based heuristic can be solved more efficiently in continuous-time rather than discrete-time for high-demand scenarios.

I. INTRODUCTION

Traffic congestion occurs as traffic demand exceeds the infrastructure’s available capacity especially during rush-hours when all drivers seek to travel through main arteries of the road network. Interestingly, even if accurate real-time information on traffic state is available to all drivers, congestion can still appear due to the selfish driving behavior, as discussed in [2].

Recent advances in telecommunication systems have greatly broadened the potential solutions that can be realized, including next generation navigation that allows for informed routing decisions to be made. Today, commercially available vehicles are already equipped with semi-autonomous features such as Adaptive Cruise Control and Collision Avoidance System, with positive impact to vehicle safety and driving behavior. Eventually, these autonomous capabilities have the potential to improve network utilization, to carry significantly more vehicles improving the average travel time [3]. Addressing the latter target is still an open problem (as detailed in [3]).

Utilizing the available information and communication technologies, it is possible to develop a route reservation architecture in order to alleviate traffic congestion as described in our earlier work [1]. In this architecture, a vehicle, communicates its origin-destination pair to a road-side unit (RSU) before setting off for its journey. Then, the RSU taking into consideration all previous requests, schedules the vehicle from its origin (O) to its destination (D) through congestion-free road segments and communicates the schedule back to the vehicle. The RSU may delay start time of the vehicle until all road segments in the shortest path become uncongested or may compute an alternative path, slightly longer than the shortest path, that includes no congested road segments, or a combination of the two. Furthermore, the RSU reserves capacity in the road network along the provided path to avoid congestion for all future route requests. These reservations provide estimates of the future state of each link of the network, thus when reservations for a link during an interval exceed its critical capacity, the specific link becomes unavailable. The quality of the future state estimates depends on the driver adherence to the RSU provided schedule. In the context of connected autonomous vehicles (AV) this may be easy while for human drivers, path enforcing policies are needed. We point out that with the available technology, such policies can be put in place.

The rest of the paper is organized as follows. A brief literature review is included in Section II while Section III mathematically describes the continuous time route reservation problem. Section IV formulates a mixed-integer-linear program that optimally solves the formulated problem. Subsequently, Section V develops a low complexity continuous-time Iterative Dijkstra heuristic algorithm (IDAC). Simulation results included in Section VI examine the performance of the proposed heuristic compared to the MILP formulation and the corresponding discrete-time Dijkstra algorithm. Finally, Section VII concludes the paper.

II. RELATED WORK

Several approaches exist that aim to dynamically reduce congestion such as, variable speed limit control, lane changing control and ramp metering which, are proposed to enable better road management [4] (all have been explored and tested in the past). Other solutions based on Dynamic Traffic Assignment (DTA) advice drivers on changing their routes according to real-time network states. However, it appears that such recommendations do not usually provide solid results on reducing congestion. A state-of-the-art dynamic approach is the perimeter control [5]. Perimeter control utilizes the Macroscopic Fundamental Diagram (MFD) dynamics in order to regu-
late traffic in-flow at the boundaries of a particular region thus to maintain the region’s density under desirable levels. To do so, boundary control techniques (i.e., street closures and traffic signalling control) restrict in-flow in cases when a region’s density exceeds critical capacity. However, these approaches do not take into consideration or apply any control action on endogenous flows (i.e., flows that are generated within the region).

The alternative time-dependent solutions use either static or stochastic models to predict traversal times [6]. Using either online or offline approaches, these travel time predictions guide drivers to follow the shorter-travel-time paths [7]. The majority of the latter solutions identify the shortest-travel-time path based only on state estimation without considering the non-linear effects involved in congested conditions [8].

Route decisions that consider the MFD dynamics can effectively influence and shrink the extent of congestion as detailed in [9]. According to recent literature [10], several routing algorithms indicate that the possibility of spreading the load across a larger area of the network can significantly improve the overall traveling times. Moreover [11] indicates that a first-order macroscopic model can accurately adapt the network mobility behavior using communications in the presence of Autonomous Vehicles (AVs). Therefore, under this setting routing reservations in both space and time should be appropriate and effective control strategy for traffic management.

As already mentioned, this work provides a more formal validation of the previous work by the authors on finding close to optimal paths utilizing the route reservation architecture [1], [12]. Specifically, the formulation of this paper provides an alternative approach of reserving road capacity, based on a continuous-time instead of the time discretization approach. This approach eliminates discretization errors and is more efficiently solved in high traffic scenarios.

the optimal path that a vehicle needs to follow in order to arrive at its destination at the earliest possible time avoiding congested roads. Therefore, the vehicle will arrive at the destination at almost the earliest possible time, as avoids roads that are expected to be at capacity.

III. Problem Formulation

A transport region is considered as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$, $|\mathcal{V}| = |\mathcal{E}|$, representing the road segments. The vehicular mobility characteristics across the investigated (homogeneous) region are defined according to the MFD [13] with parameters $\rho_c$, $\rho_p$, and $v_f$, representing the critical density corresponding to the maximum flow, the jam density and the free-flow speed, respectively. Each road segment $(i, j) \in \mathcal{E}$, $(i, j) \in \mathcal{V}$ is characterized by the parameters $\rho_c^f$, $\rho_p^f$, and $\lambda_{ij}$ indicating the jam density, the critical density and the length of road segment $(i, j)$, respectively. Additionally, the parameter $\rho_{ij}(t)$ defines the instantaneous density of segment $(i, j)$ at time $t$, while the parameter $\rho_c^f$ denotes the maximum density that a road segment can accommodate in order to operate at speed-at-capacity $v_c$, i.e., $\rho_{ij}(t) \leq \rho_c^f$. Because $v_c \approx v_f$, the traversal-time for each road segment $(i, j) \in \mathcal{E}$ is equal to $\bar{t}_{ij} = \lambda_{ij}/v_f$.

Given the past route requests, the proposed route reservation scheme keeps track of the accumulated number of vehicles within each road segment over time. This is achieved by maintaining $n_{ij}(t_0, t)$ which indicates the number of vehicles in road segment $(i, j)$ at some future time $t \geq t_0$, where $t_0$ is the current time. These state variables help the RSU determine the time intervals at which each road link is admissible. A road segment $(i, j)$ is considered admissible at time $t$ if the number of reservations at that time is not larger than the number of vehicles corresponding to the segment’s critical density, i.e.:

$$n_{ij}(t_0, t) \leq \rho_c^f \lambda_{ij} \quad (1)$$

In this way, the RSU constructs the admissible sets $S_{ij}(t_0) = \{(t_{ij1}^1, t_{ij1}^w), ..., (t_{ijK_{ij}(t_0)}, t_{ijK_{ij}(t_0)})\} \in \mathcal{E}$, which define the admissible time intervals $(t_{ijk}, t_{ijk+1}^w)$, $k \in K_{ij}(t_0) = \{1,...,K_{ij}(t_0)\}$, where $K_{ij}(t_0)$ denotes the number of admissible time intervals of segment $(i, j)$ at time $t_0$. Note also that $t_{ijk}^w < t_{ijk+1}^w$ and $t_{ijk}^w$ denote the lower and upper bounds of the $k$-th admissible time interval of the link $(i, j) \in \mathcal{V}$, respectively. These time intervals are determined by the RSU, given the past reservations and the path of a vehicle, under the assumption that the vehicle will travel at a constant speed $v_f$.

Congestion-free routing can be achieved if vehicles traverse the network only through admissible road segments. For this reason, the RSU utilizes the admissible intervals of each link and needs to formulate and solve an optimization problem to determine the shortest path for a vehicle such that the admissibility condition (1) for each link is always satisfied. Two alternative options arise in case the shortest path of vehicles includes a non-admissible link. The first prompts vehicles to wait at their origin until all road segments in the shortest path become admissible. The second chooses an alternative path where all links are admissible. Further, combining both options may yield to a better solution, in the sense that the vehicle will arrive at the destination at the earliest time. For this formulation, the following variables are needed.

Let $t_0$ and $d_i$ denote the routing request time and the vehicle arrival time at node $i$, respectively. The cost of traversing the road segment $(i, j)$, $c_{ij}(d_i, t_0)$ is:

$$c_{ij}(d_i, t_0) = \begin{cases} \bar{c}_{ij}, & \text{if } d_i + t \in S_{ij}(t_0), \forall 0 \leq t \leq \bar{c}_{ij}, \ i \neq O \\ c_{ij} + w, & \text{if } t_0 + w + t \in S_{ij}(t_0), \\ \infty, & \text{otherwise}, \end{cases} \quad (2)$$

The alternative time-dependent solutions use either static or stochastic models to predict traversal times [6]. Using either online or offline approaches, these travel time predictions guide drivers to follow the shorter-travel-time paths [7]. The majority of the latter solutions identify the shortest-travel-time path based only on state estimation without considering the non-linear effects involved in congested conditions [8].
where, \( w \) denotes the time interval for which the vehicle will have to wait at its origin before starting its trip (\( i = O \)).

This work formulates and solves an optimization problem for determining the path that will allow a vehicle to arrive at its destination node \( D \) at the earliest possible arrival time while avoiding non-admissible links. This is referred to as the Earliest Arrival Time at Destination (EATD) problem presented next.

**EATD problem:** Given an origin-destination (\( O - D \)) pair let \( p_h \) denote the \( h \)-th path from source \( O \) to destination \( D \) denoted as \( p_h = (v_i^0, v_j^1, (v_i^1, v_j^2), (v_i^2, v_j^3), \ldots (v_{kn}^1, v_{kn}^h), v_{kn}^h) \), where \( v_i^0 \in V \) is the \( j \)-th visited node in the \( h \)-th path, with \( v_i^0 = O \) and \( v_{kn}^h = D \). Additionally, let \( d_{ij}^h \) denote the arrival time at junction \( v_j \) then, the arrival time to each node of the path can be expressed as:

\[
\begin{align*}
    d_{ij}^0 &= t_0, \quad w \geq 0 \\
    d_{ij}^h &= d_{ij}^{h-1} + c_{ij} \quad \text{if} \quad x_{ij}^h = 1 \\
    d_{ijk}^h &= d_{ijk}^{h-1} + c_{ijk} \quad \text{if} \quad x_{ijk}^h = 1 \\
\end{align*}
\]

Given the vehicle routing request time \( t_0 \) and the admissible sets \( S_j(t_0) \), \( \forall (i, j) \in \mathcal{E} \), then the EATD problem requests the earliest-destination-arrival-time (from \( O \) to \( D \)) subject to avoiding links that are at capacity. Hence, the EATD problem can be expressed in compact form as:

\[
    d_{ij}^D = \min_{d_{ij}} d_{ij}^D \quad \text{s.t.} \quad \text{Constraints (2) - (5)}.
\]

Next, we present a MILP formulation for solving the EATD problem. This is an NP-complete problem [1] thus the MILP can only solve small instances of the problem. Thus in the subsequent section we also present a heuristic for obtaining a solution in a reasonable time.

**IV. MILP FORMULATION OF EATD PROBLEM**

In this section, we develop a Mixed Linear Programming Formulation (MILP) that optimally solves this problem. The formulation aims to select appropriate road segments that connect the origin \( O \) with the destination \( D \) and minimize the arrival time at \( D \), while ensuring the admissibility condition. For this reason, we introduce binary variables \( x_{ij} \) and \( \psi_{ijk} \), \( k \in \mathcal{K}_{ij}(t_0) \) which indicate that the road segment \( (i, j) \) is part of the optimal path \( \pi^* \) (i.e., \( x_{ij} = 1 \)), and the admissibility condition is satisfied only for the \( \mathcal{K}_{ij} \) time interval \( (\psi_{ijk} = 1) \).

The mathematical formulation for the EATD problem is as follows:

\[
\begin{align}
\text{(P0)} \quad & \min_{x_{ij}, \psi_{ijk}, d_{ij}, \forall i, j, k} d_D \\
\text{s.t.} \quad & \sum_{(i,j) \in \mathcal{E}} x_{ij} - \sum_{(j,i) \in \mathcal{E}} x_{ji} = \begin{cases} 
1, & \text{if} \ i = O, \\
-1, & \text{if} \ i = D, \\
0, & \text{otherwise}, 
\end{cases} \quad \forall i \in V \\
\end{align}
\]

\[
\sum_{k \in \mathcal{K}_{ij}(t_0)} \psi_{ijk} = x_{ij}, \quad (i, j) \in \mathcal{E}, \quad (7c)
\]

\[
\text{if} \ x_{ij} = 1 \quad \text{then} \quad d_j = d_i + \bar{c}_{ij}, \quad (i, j) \in \mathcal{E}, \quad (7d)
\]

\[
\text{if} \ \psi_{ijk} = 1 \quad \text{then} \quad d_j \geq t_{ijk}^j \quad \text{and} \quad d_j \leq t_{ijk}^u, \quad (i, j) \in \mathcal{E}, \quad k \in \mathcal{K}_{ij}(t_0), \quad (7e)
\]

\[
d_i \geq t_{0}, \quad i \in \mathcal{V} - \{O\}, \quad (7f)
\]

\[
x_{ij} \in \{0, 1\}, \quad (i, j) \in \mathcal{E}, \quad (7g)
\]

\[
\psi_{ijk} \in \{0, 1\}, \quad (i, j) \in \mathcal{E}, \quad k \in \mathcal{K}_{ij}(t_0). \quad (7h)
\]

In the above formulation, equalities (7b) describe the flow constraints that ensure connectivity of the optimal path from source to destination, while (7c) ensure that exactly one time-interval satisfies the admissibility condition. Conditions (7d) and (7e) describe the increase in cost incurred by traversing road segment \( (i, j) \) and ensure that a link is traversed during an admissible time interval, respectively. Constraints (7f)-(7h) simply denote the nature (e.g., continuous, binary) and range of each set of variables. Note that the waiting time at the origin \( w \), is implicitly imposed by letting \( d_0 \geq t_0 \), so that \( w = d_0 - t_0 \).

<table>
<thead>
<tr>
<th>ID</th>
<th>Logical Expression</th>
<th>Equivalent MILP Exp.</th>
</tr>
</thead>
<tbody>
<tr>
<td>LC1</td>
<td>if ( \delta ) = 1 then ( \sum_j a_j x_j \leq b )</td>
<td>( \sum_j a_j x_j - b \leq U(1 - \delta) )</td>
</tr>
<tr>
<td>LC2</td>
<td>if ( \delta ) = 1 then ( \sum_j a_j x_j \geq b )</td>
<td>( \sum_j a_j x_j - b \geq L(1 - \delta) )</td>
</tr>
</tbody>
</table>

Table 1. Equivalent MILP expressions to specific logical one.

Although formulation (7) fully describes EATD, it cannot be solved with mathematical optimization solvers as it does not adhere to any standard form, because of the logical constraints in (7d) and (7e). For this reason, the two logical constraints are transformed into equivalent MILP expressions according to Table 1 [14]. In the table, it is true that \( U \) and \( L \) denote upper and lower bounds to the expression \( \sum_j a_j x_j - b \), i.e. \( L \leq \sum_j a_j x_j - b \leq U \). Appropriate exploitation of the table yields the following MILP formulation:

\[
\begin{align}
\text{(P1)} \quad & \min_{x_{ij}, \psi_{ijk}, d_{ij}, \forall i, j, k} d_D \\
\text{s.t.} \quad & \text{Constraints (7b), (7c), (7f) - (7h)}, \quad (8a) \\
& \quad d_i - d_j + \bar{c}_{ij} \leq M_1(1 - x_{ij}), \quad (i, j) \in \mathcal{E}, \quad (8c) \\
& \quad d_i - d_j + \bar{c}_{ij} \geq M_2(1 - x_{ij}), \quad (i, j) \in \mathcal{E}, \quad (8d) \\
& \quad d_i \geq t_{ijk}^{l}, \psi_{ijk}, \quad (i, j) \in \mathcal{E}, \quad k \in \mathcal{K}_{ij}(t_0), \quad (8e) \\
& \quad d_j \leq t_{ijk}^{u} + M_3(1 - \psi_{ijk}), \quad (i, j) \in \mathcal{E}, \quad k \in \mathcal{K}_{ij}(t_0), \quad (8f)
\end{align}
\]

where \( M_1, M_2 \) and \( M_3 \) are constants.

In formulation (8), constraints (8c) and (8d) are equivalent to (7d). This is true because the condition "if \( x_{ij} = 1 \) then \( d_j = d_i + \bar{c}_{ij} \)" is equivalent to the condition "if \( x_{ij} = 1 \) then \( d_j \leq d_i + \bar{c}_{ij} \) and \( d_j \geq d_i + \bar{c}_{ij} \); hence, using LC1 and LC2 from Table 1 with \( U = M_1 \) and \( L = M_2 \), yields the result. Let \( D^* \) denote an upper bound to the solution of the problem (e.g., obtained through a heuristic algorithm); then, it is true that
The customized Dijkstra algorithm for solving the Relaxed-EATD problem is described in Alg. 2. First, the algorithm initializes the arrival times \(d_i\) to each junction to infinity except from the arrival time of the origin node which is set equal to the request time plus the preliminary waiting incurred so far from previous iterations of IDAC. Initially all junctions are non-labelled, and hence the set of non-labelled nodes, \(Q\), is set to \(V\), while the predecessor list \(P\) is set to null (lines 2-3). Then, an iterative procedure is followed until the travel times of all junctions are finalized, i.e. all nodes are labelled. In each iteration, the junction \(l\) that has the earliest arrival time is labelled (i.e. its travel time is finalized) (line 5) and then it is examined whether the travel time of \(l\)'s neighbours can be improved (lines 6-12). To do so, the smallest waiting time \(w_{ij}\) is computed which is needed to go from junction \(l\) at time \(d_l\) to junction \(j\), based on the admissibility of the particular road segment (line 7). If the examined labelled junction \(l\) improves the arrival time at its neighbour \((j)\) then the arrival time at \(j\), \(d_j\) is updated and \(l\) is noted as the predecessor of \(j\) (lines 8-11). In this way, the algorithm calculates and updates the earliest arrival times of non-labelled neighbour road junctions in each iteration. The above procedure repeats until all road junctions are characterized as labelled. Finally, the predecessor list which holds the best path from \(O\) to \(D\) is exploited to calculate the total waiting time at all junctions, \(w^T\) (lines 14-17). By allowing waiting at all nodes, Alg.2 returns a better solution than the proposed solution; nonetheless, it is not applicable to real traffic networks since vehicles cannot stop and wait along arbitrary road junctions.

As emphasized above, IDAC is a continuous-time adaptation of the discrete-time heuristic proposed in [1], referred to in the sequel as IDAD. The complexity of the algorithm in [1] is \(O(LEV)\), with \(L\) denoting the number of reiterations of the Relaxed-EATD problem. The complexity of IDAC (i.e., the proposed continuous-time variant) is improved to \(O(LEV \log (E))\) since the waiting intervals that may be required at any intermediate road junction can be calculated more ef-
V. SIMULATION RESULTS

In order to evaluate the performance of the developed algorithms, a Manhattan-style network consisting of 36 two-way, single-lane road segments and 28 junctions. Vehicles arrive probabilistically, according to Poisson distribution, while for each trip a random O – D pair is selected. All vehicles are assumed to follow the route assigned to them by the RSU. A critical density of \( \rho_C = 30 \text{veh}/\text{km}/\text{lane} \) and a free-flow speed of 15m/s are used. Simulations are performed for varying total flow rates in the range of 1000 to 8000 veh/h. Further both algorithms are compared with the results obtained by the Iterative Dijkstra Algorithm Discrete-Time (IDAD) as proposed in [1].

Performance evaluation is conducted in both ideal and realistic environments. In an ideal environment, the topology is modeled as a graph with edge costs equal to the travel time of the corresponding road segments, assuming free-flow speed. In this ideal environment, vehicles do not form queues whereas in a realistic environment vehicles form queues and follow each other, have acceleration and deceleration times, and experience delays at intersections due to the passing of higher priority vehicles. To capture these effects, the SUMO micro-simulator [15] is used, which employs the Krauss car following model [16] for vehicle mobility. In our simulations the car-following model parameters are set as follows: vehicle length 5m, maximum speed 15m/s, acceleration 2.5m/s², deceleration 4.5m/s², and minimum gap distance of 2.5m. To account for stochastic effects, ten (10) Monte Carlo simulations are performed for each considered scenario. In addition to comparing the performance of IDAC and MILP, comparisons are also conducted against the uncontrolled scenario (referred to as TB in the plots) where each vehicle travels from the origin to destination along the shortest distance path without any waiting at the origin.

A. Ideal simulation environment

Figures 1 (a), (b) and (c) depict respectively the average travel time (origin waiting not taken into consideration), origin waiting time and total time (origin waiting plus travel time) that vehicles experience within the network for different flow rates.

Fig. 1(a) shows that as the flow rate increases, the paths produced from MILP are longer than those of IDAC and IDAD, resulting in higher average travel times. In fact, both IDAC and IDAD paths appear to maintain constant travel times irrespective of the demand flows.

Figures 1 (b) and 1 (c) illustrate that at low flow-demands (e.g., 1000 to 5000, veh/h) the performance of the all algorithms is almost identical, while at high flow-demands (greater than 5000 veh/h) the MILP approach outperforms the IDAC and IDAD algorithms by up to 20% and 30% respectively. The reason is that at low demands there are no significant restrictions in terms of road segment admissibility so that all algorithms yields similar results. However, as demand increases, the admissibility sets become more fragmented and require the examination of a large number of time-interval combinations to find the best path. Hence, despite the fact that MILP routes vehicles through longer paths, it imposes less waiting time at the origin leading to solutions with better total time. Furthermore, 1 (b) illustrates that as the flow rate increases, the average waiting time increases exponentially and becomes more than one order of magnitude larger than the average trip time. Further, as 1 (b) demonstrates, the waiting time at the origin is increased up to 10% for IDAD compared to IDAC, since IDAD may reserve more time-slots than it actually needs. Despite this effect, it is illustrated in the realistic simulation environment that all the proposed approaches lead to significantly better results compared to the TB approach where every vehicle enters immediately into the network and follows the shortest path.

The combined results of 1 (a), (b) and (c) indicate that the MILP solutions impose virtual waiting within network (involve cycles inside the network), which leads to less total time but longer paths and hence higher fuel consumption. On the contrary, IDAC and IDAD approaches are based on Dijkstra’s algorithm which do not produce cycles; hence, all required waiting introduced only at the origin.

Fig. 1 (d) demonstrates the average execution time of all algorithms for different flow rates. It can be seen that as the flow rates increase, the execution time of all algorithms also increases because the admissibility sets are more fragmented and require the examination of a larger number of time-intervals. Comparing the execution time of the IDAC and IDAD algorithms to the MILP approach, they are significantly faster, outperforming the MILP by two to three orders of magnitude. Comparing IDAD and IDAC algorithms in low flow rates the IDAD seems more efficient than IDAC but in high flow rates IDAD becomes slower despite IDAD’s complexity being lower. The increase in execution time is due to unnecessary reservations which in turn increase the road segments unavailability times and thus more re-iterations are required as the problem gets harder to be solved.

B. Realistic simulation environment

Figures 2 (a) and (b) illustrate respectively, the mean number of vehicles that reach their destination and the average travel time as a function of the different flow rates. Specifically, the dashed lines in Fig. 2 (a) represent the average number of vehicles that have finished their journey within the simulation time and the scattered plots are the realizations obtained by each simulation run. Similarly, the dashed lines in Fig. 2 (b) represents
only a portion of vehicles that adhere to the specific route of having multiple homogeneous regions and the e include the investigation of multi-reservoir networks with provides close-to-optimal results. Future work directions in terms of road utilization and travel time against that both algorithms achieve substantial improvements in the traditional behavior of taking the shortest path. In addition, the results indicate that the developed heuristic provides close-to-optimal results. Future work directions include the investigation of multi-reservoir networks with multiple homogeneous regions and the effect of having only a portion of vehicles that adhere to the specific route.

VII. Conclusions
This work has adapted the route reservation scheme introduced in [1] which aims to achieve congestion-free routing. In this suggested adaptation, the reservations are maintained based on a continuous-time framework compared to the discrete-time quantized versions of [1]. An optimal solution of the problem is obtained by constructing an appropriate MILP formulation that can be solved with standard solvers. An iterative heuristic algorithm based on a customized version of Dijkstra’s shortest path algorithm is also developed. Simulation results in both ideal and realistic conditions indicate that both algorithms achieve substantial improvements in terms of road utilization and travel time against the traditional behavior of taking the shortest path. In addition, the results indicate that the developed heuristic provides close-to-optimal results. Future work directions include the investigation of multi-reservoir networks with multiple homogeneous regions and the effect of having only a portion of vehicles that adhere to the specific route.

REFERENCES