BNART: A Novel Centralized Traffic Management Approach for Autonomous Vehicles

Anonymous

***Abstract*—Traffic management poses a critical challenge in modern metropolitan areas, where efficient management can significantly reduce travel time and vehicle emissions. With the vision of smart cities in mind, the increasing prominence of autonomous cars is anticipated to revolutionize road networks under the control of a central supercomputer known as the Central Traffic Control Unit (CTCU), which maintains real-time location information for each vehicle within its jurisdiction and assumes responsibility for path planning. This paper investigates the problem of traffic management for self-driving vehicles, focusing on the utilization of essential information, such as source and destination locations, to guide vehicles along the most efficient routes, mitigating road congestion and ensuring prompt arrivals. To address this challenge, we formulate the problem as a mixed-integer linear programming model. Furthermore, due to its complexity, we propose a heuristic method called the Best Neighbor Algorithm for Routing Traffic (BNART). We extensively evaluate the performance of our proposed heuristic approach against optimal solutions on small-scale instances and other state-of-the-art methods such as the shortest path algorithm on large-scale instances. Through simulation, we demonstrate that our proposed method outperforms other methods in terms of average travel time, exhibiting a negligible gap to the optimal solution.**

***Index Terms*—Traffic management, autonomous vehicles, central traffic control, path planning, congestion avoidance.**

I. Introduction

As the world witnesses remarkable advancements in autonomous driving technology, it becomes increasingly crucial to examine how we can effectively manage the flow of these vehicles within our existing transportation infrastructure. The rise of autonomous vehicles promises unprecedented benefits, including enhanced safety, reduced congestion, optimized fuel consumption, and improved accessibility. However, to fully realize these benefits, a Central Traffic Control Unit (CTCU) must be in place to optimize the flow of autonomous vehicles within the existing infrastructure. The CTCU represents a paradigm shift in traffic management for autonomous vehicles. By employing a centralized system that collects real-time information, including source and destination points, the CTCU leverages advanced algorithms and predictive analytics to determine the most efficient routes for autonomous vehicles. Its primary objective is to minimize travel time by dynamically avoiding congestion-prone areas and directing vehicles along the fastest available paths. This visionary approach to traffic management has the potential to reshape urban mobility. By synchronizing the movements of autonomous vehicles, the CTCU optimizes traffic flow, reducing delays and enhancing overall system capacity. Moreover, it enables seamless integration with existing transportation infrastructure and human-driven vehicles, fostering harmonious coexistence on our roads.

In this paper, we delve into the complex realm of traffic management for self-driving vehicles, with a focus on the dynamic road environment where vehicles enter the system randomly. The CTCU must possess the capability to adapt to this ever-changing landscape as new vehicles join the system. When a vehicle joins the system, important data such as the source and destination of the vehicle is sent to the CTCU for processing. The CTCU then sends the calculated routes and speed to the vehicle to join the road and traffic flow, continuously updating the vehicle's route in real time based on the current road conditions. The objective here is not merely to find the shortest path for each vehicle, but rather to minimize the overall traveling time for all vehicles in the system by choosing routes that result in less traffic congestion. To achieve this, the CTCU continuously scans roads in real-time to detect any potential congestion.

While the vehicle Routing Problem has been an area of research for some time, various methods have been proposed to solve this problem. Some work focuses on accurate data measurement using the Internet of Things (IoT) for better data accuracy [1]–[3]. Others address the dynamicity of the problem [4]–[6] by employing nature-inspired algorithms like Genetic Algorithm and Ant Colony Optimization. Some researchers have combined big data and genetic algorithms to consider the dynamicity of route planning and traffic management [7]. Others have solved the problem using a genetic algorithm and combinational optimization [8]. Additionally, a predictive control algorithm applied to a Markov decision process has been proposed to tackle dynamicity [9]. However, these approaches often require extensive learning periods or large amounts of training data to perform well in real-world scenarios.

In this paper, we contribute to the field of traffic management for autonomous vehicles by presenting a novel approach. To address the dynamic road environment and adapt to changes without prediction, we divide the roads into multiple smaller segments, allowing for the mathematical calculation of traffic in each segment (i.e., the number of vehicles sharing the same segment) as new vehicles enter the environment. We also consider the location, speed, and interactions of each vehicle as it enters and exits road segments and reaches intersections.

To address this challenge, we first present a mathematical formulation of the problem as a mixed-integer linear programming (MILP) model. However, due to the inherent complexity of the optimization model, we propose a heuristic method

known as the Best Neighbor Algorithm for Routing Traffic (BNART). By proposing this scalable algorithm, we aim to enhance traffic flow, minimize congestion, and reduce travel times in dynamic road environments. We then thoroughly evaluate the performance of our proposed heuristic method in comparison to optimal solutions for small instances, as well as state-of-the-art methods such as the Breadth First Search (BFS) shortest path algorithm for large instances. By focusing on metrics such as average traveled distance and average traveling time, we assess the effectiveness of BNART in achieving efficient traffic management for self-driving vehicles.

Through extensive simulation results, we showcase the capabilities of our proposed heuristic method. While it may not surpass the BFS shortest path algorithm in terms of average traveled distance, our method demonstrates superior performance in terms of average traveling time, with a remarkably close gap to the optimal solution. These findings underline the potential of the BNART heuristic method to effectively navigate autonomous vehicles through the road network, minimizing delays, and improving overall travel efficiency.

The remainder of this paper is structured as follows. Section II summarizes the related work. Section III describes the system model and the problem. Section IV presents a detailed overview of the problem formulation, introducing the mathematical foundation of our approach. In Section V, we describe the BNART heuristic method, highlighting its key components and operational aspects. Section VI presents the experimental evaluation of our proposed method, comparing its performance against optimal solutions and existing state-of-the-art algorithms. Finally, Section VII concludes the paper, summarizing the findings and discussing potential avenues for future research.

II. Related Work

The management of traffic for self-driving vehicles has been the focus of extensive research efforts in recent years. In this section, we review the literature that addresses various aspects of traffic management, including routing, optimization, and dynamic road environments.

A. Routing Algorithms for Self-Driving Vehicles

Efficient routing is a fundamental component of traffic management for self-driving vehicles. Numerous routing algorithms have been proposed to address this challenge. For instance, Zhang et al. [10] introduced a genetic algorithm-based approach for the path planning of autonomous vehicles. Their algorithm considers factors such as road conditions, traffic congestion, and vehicle priorities, resulting in improved travel times and reduced congestion compared to traditional methods. In the realm of real-time traffic management, Wang et al. [11] developed a cooperative traffic control system for autonomous vehicles. This system utilizes vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication to exchange information and optimize traffic flow. Through simulation experiments, they demonstrated enhanced traffic efficiency and reduced travel times. Integration with existing infrastructure has also been a focus of research in traffic management. Li et al. [12] proposed a centralized traffic management system that incorporates dynamic traffic signal control and route optimization for autonomous vehicles. By utilizing real-time data and predictive models, their system effectively reduces congestion and improves overall traffic performance. Machine learning techniques have also been leveraged in traffic management for self-driving vehicles. Zhou et al. [13] proposed a deep reinforcement learning-based approach for traffic signal control and routing optimization. The system learns optimal control policies through interactions with the environment, resulting in improved traffic flow and reduced travel times. Furthermore, the concept of a centralized traffic control unit aligns with the work of Smith et al. [14], who proposed a centralized traffic management architecture for autonomous vehicles. Their system collects real-time data from vehicles and infrastructure, enabling efficient route planning and congestion management. The results demonstrated enhanced traffic flow and reduced travel times in simulated scenarios.

B. Dynamic Road Environments and Traffic Adaptability

Addressing the dynamic nature of road environments and adapting to changes in traffic patterns is crucial for effective traffic management. Yang et al. [15] addressed this challenge by proposing a multi-agent traffic management framework, where autonomous vehicles act as agents to collectively optimize traffic flow. Through simulation experiments, they demonstrated improved traffic efficiency and reduced travel times compared to traditional methods. In another study, Liu et al. [16] presented a dynamic traffic management system using predictive control applied to a Markov decision process. Their approach considers real-time traffic conditions and predicts future traffic states to optimize traffic signal timings. However, their method requires a long learning period and a significant amount of training data to perform well in real-world scenarios.

C. Contributions of Our Work

In this paper, we contribute to the field of traffic management for autonomous vehicles by presenting a novel approach that addresses the challenges of dynamic road environments and traffic adaptability. By dividing roads into smaller segments and considering location, speed, and interactions of vehicles, we aim to enhance traffic flow and minimize congestion in real time. Overall, our work contributes to the advancement of traffic management for self-driving vehicles by offering insights into efficient routing algorithms, addressing dynamic road environments, and proposing a scalable heuristic method. The subsequent sections of this paper provide a detailed analysis and evaluation of our proposed approach, further validating its effectiveness and applicability in real-world scenarios.

III. System Model and Problem Description

A. System Model

We consider a road structure as shown in Fig. 1 with a directed graph $G = (P, L)$, where P is a set of nodes and L is a set of links (edges) connecting any two nodes. To easily calculate the speed of vehicles and traffic on a road for our mathematical formulation and modeling of the problem, we divide each road (from one intersection to another) into fixed equal segments. Hence, the node set as shown in the figure is divided into two subsets: a set of intersection nodes $I_{Nodes} \in P$ (shown in the figure with red nodes), and a set of non-intersection nodes (shown with orange color) for road segments. We consider each vehicle $v \in V$ in the network is registered and reports to the CTCU whenever it plans to enter the network at time T_s^v and traverse a road structure from a source S_v to a destination D_v; an example in the figure is shown with a blue arrow, where vehicle 1 (the blue vehicle) starts its journey from S_1 (located on link $< 17, 16 >$) traversing several intersection and non-intersection nodes until it reaches destination D_1 located on link $< 40, 43 >$. Note that any link $< x, y >$ represents a direction of a vehicle from node x to y, and it is not equal to link $< y, x >$. We let T_{Road} be the time taken to traverse any segment (link) when a vehicle is not sharing the link with any other vehicles and it is going at its maximum speed. We add a delay T_{Delay} to any vehicle when it arrives/departs any intersection node (that is one of the nodes on the traversing link is I_{Node}), or when it starts/arrives from/to source/destination node. We also add a multiple traversing delay time percentage of $C_{\%}$ to a vehicle for a number of vehicles that are sharing the same directed link. The vehicles' arrival into the road network is considered to follow a Poisson distribution with density ρ Vehicle/Km [17].

B. Problem Definition

The problem that we are trying to solve here is to find the best routing path for each autonomous vehicle v in the network (system) from its source S_v to its destination D_v by considering the arrival of several vehicles at any time that minimizes the traffic congestion and traveling time of all vehicles. It is to be observed that we are not looking for the shortest path since our objective is to minimize the traveling time to reduce gas emissions, not the traveled distance. Hence, finding the shortest path for each vehicle might cause congestion on the road and thus a longer traveling time. For example in fig. 1, if we consider the shortest paths for the three vehicles (v =1, 2, and 3) as drawn in the figure with three arrows (blue, green, and red), the blue vehicle might cause congestion with the red vehicle if they share both links $< 4, 23 >$ and $< 23, 5 >$ at the same time. Therefore, as shown in the figure, if the blue vehicle changes its course to follow the dotted line via intersection node 6, traffic will be avoided. Also, we should be careful of other vehicles that are sharing the same link, and the chosen path should avoid congested links as possible. It can also be noted that the green

Fig. 1. Illustration of the system model; all the whereabouts of cars in the network are kept track of in the CTCU. The red and yellow nodes respectively represent the intersection and non-intersection nodes. S_1, S_2, and S_3 are respectively the sources of vehicles v =1, 2, and 3. Similarly, D_1, D_2, and D_3 represent the destinations of the three vehicles, and the blue, green, and red arrows show the routes of the vehicles respectively. The blue dotted line illustrates an alternative path for vehicle 1 to avoid congestion with vehicle 3.

vehicle since it is going in the opposite direction of other vehicles, will not cause any congestion. In this problem, the traffic and/or congestion is measured based on the number of vehicles that are sharing the same link or road segment and can be calculated by adding more delays (i.e., $C_{\%}$) as more vehicles are sharing the same link (i.e., $C_{\%} * N^v_{<i,j>} * T_{road}$), where $N^v_{<i,j>}$ is the number of vehicles that are sharing the same link $< i, j >$ with vehicle v. We can find the location of each vehicle by calculating its speed and the time of its arrival in the network (starting time at the source). Since the length of each segment is known, therefore it is easy to calculate the location and traveling time of each vehicle in the network. For simplicity, we assume all vehicles use the maximum road speed (i.e., the time to traverse a link with maximum speed is T_{Road}), and when they reach or depart any intersection, all vehicles have a fixed speed where their link traversing time is calculated by $T_{Delay} + T_{Road}$. The delay T_{Delay} is added because of the acceleration, deceleration, and traffic lights or stop signs. When a new vehicle enters the network at any time, the problem is solved again considering the current locations of other vehicles in the road network.

Problem Definition 1: *Given a Graph G which consists of P nodes connected through L links. The problem is, for each vehicle $v \in V$ that enters the network randomly, to find a path from its source node S_v to its destination node D_v, such that the total traveling time (i.e., $\sum_{v \in V} T_D^v$) is minimum.*

IV. Problem Formulation

In this section, we mathematically formulate the problem as a mixed integer linear programming (MILP). The used notations are listed in Table I. Let T_D^v indicate the arrival time

of vehicle v to its destination. The objective of the optimization model is to minimize the arrival time for all vehicles in set V to their destinations. It can be mathematically written as follows:

$$Min \qquad \sum_{v \in V} T_D^v \tag{1}$$

subject to: (2) - (15), where these constraints are derived in detail in Sections IV-A to IV-C.

A. Routing Constraints

In this subsection, the route for each vehicle is constructed from its source (starting point) to its destination. Let $R^v_{<i,j>} \in \{0,1\}$ indicate whether link $< i,j >$ is set on the route of vehicle v when it traversed from its source to its destination (i.e., $R^v_{<i,j>} = 1$, if link $< i,j >$ is on the route of vehicle v and zero otherwise). Let the source and destination of vehicle v be S_v and D_v respectively. Also, let us assume that the sources and destinations of different vehicles are located somewhere on links. Now, $R^v_{<i,j>} = 1$ if the source or the destination of vehicle v is located on link $< i,j >$. The mathematical formulation for this constraint is written as follows:

$$R^v_{<i,j>} = 1 \qquad \forall v \in V, < i,j >= S_v || D_v \tag{2}$$

Constraints (3) and (4) respectively assure that for the route construction, there is always one adjacent link from the source link and one adjacent link to the destination link that belongs to the route of vehicle v.

$$\sum_{i:<i,j>\in L} R^v_{<i,j>} = 1 \qquad \forall v \in V, < h,i >= S_v \tag{3}$$

$$\sum_{j:<i,j>\in L} R^v_{<i,j>} = 1 \qquad \forall v \in V, < j,k >= D_v \tag{4}$$

To construct a route (path) from a source to a destination of a vehicle, constraint (5) makes sure that links are chosen adjacent to each other. In other words, if a link is chosen on the route of a vehicle (for example, $R^v_{<i,j>} = 1$), then an adjacent link (such as $< j,k >$) will be chosen (i.e., $R^v_{<j,k>} = 1$) to construct connected links for the route. Consequently, if a link is not on the route (i.e., $R^v_{<i,j>} = 0$), then none of its adjacent links should be on the route except if an adjacent link is a source or destination link.

$$\sum_{j:<i,j>\in L} R^v_{<i,j>} - \sum_{j:<j,k>\in L} R^v_{<j,k>} = 0 \qquad \forall v \in V, \forall j \in P :< i,j > || < j,k >\neq S_v || D_v \tag{5}$$

B. Traversing Time Constraints

This subsection determines the time when a vehicle traverses a link, if this link is on the route of the vehicle. Let

TABLE I
NOTATIONS USED IN PROBLEM FORMULATION

Parameters		
P		Set of nodes.
L		Set of links (edges).
V		Set of vehicles.
S_v		Starting point (source) of vehicle v.
D_v		Destination of vehicle v.
B		Large constant bigger than end of system time.
T_s^v		Time when vehicle v enters the network (system).
T_{Road}		Time to traverse any link with maximum road speed.
T_{Delay}		Delay added to the system time.
I_{Node}		Set of intersection nodes.
$C_{\%}$		Percentage of added time delay based on the number of vehicles sharing the same link.
Decision Variables		
$R^v_{<i,j>}$	$\in \{0,1\}$	Indicate whether link $< i,j >$ is set on the route of vehicle v.
$X^v_{<i,j>}$	≥ 0	System time when vehicle v traversed link $< i,j >$.
T_D^v	≥ 0	System time when vehicle v arrives to destination.
$N^v_{<i,j>}$	$\in \mathbb{N}$	Number of vehicles sharing link $< i,j >$ with vehicle v.
$\aleph^{v,v'}_{<i,j>}$	$\in \{0,1\}$	Indicate vehicle v' is sharing link $< i,j >$ with vehicle v.
$\aleph l^{v,v'}_{<i,j>}$	$\in \{0,1\}$	Indicate if vehicle v' is passing through link $< i,j >$ [To linearize the multi-vehicle constraint]
$\aleph g^{v,v'}_{<i,j>}$	$\in \{0,1\}$	Indicate if vehicle v' is arriving to link $< i,j >$ [Require to linearize the multi-vehicle constraint]

$X^v_{<i,j>}$ indicate the system time when vehicle v traversed (gets to the end of) link $< i,j >$. We set $X^v_{<i,j>} = 0$ if link $< i,j >$ is not on the route of vehicle v as shown in constraint (6). To be noted, B is a big constant larger than any elapsed system time.

$$X^v_{<i,j>} \leq B * R^v_{<i,j>} \qquad \forall v \in V, \forall < i,j >\in L \tag{6}$$

Let T_s^v be the time when vehicle v enters the system, and let T_{Road} be the time when a vehicle traverses a link with the maximum road speed assuming that all the roads have a fixed maximum speed. Also, let T_{Delay} be a delay added to the system time due to different reasons such as acceleration/deceleration to/from intersections or destination/source. Constraints (7), (8), and (9) obtain the time when vehicle v arrives at the end of the link $< i,j >$ respectively when the link is the starting point (source), destination, and adjacent to an intersection. In any situation, a vehicle needs T_{Road} time to traverse a link from one end to another (i.e., from vertex i to j on link $< i,j >$). Constraint (7) adds delay T_{Delay} to the system time due to the time a vehicle requires to get into the road and starts accelerating from the source. Similarly, constraint (8) adds the same delay to decelerate and get off the road to the destination. Whereas, constraint (9) adds delay T_{Delay} when vehicle v arrives at an intersection or moves away from an intersection due to acceleration/deceleration, and stop signs/traffic lights. To be noted, I_{Node} indicates the set of intersection nodes.

$$X^v_{<i,j>} = T_{Delay} + T_{Road} + T^v_s \qquad \forall v \in V, < i,j >= S_v \tag{7}$$

$$T^v_D \geq T_{Delay} + T_{Road} + X^v_{<i,j>} \qquad \forall v \in V, < i,j >= D_v \tag{8}$$

$$\begin{aligned} X^v_{<i,j>} + B(1 - R^v_{<h,i>}) + B(1 - R^v_{<i,j>}) \\ \geq T_{Delay} + T_{Road} + X^v_{<h,i>} \\ \forall v \in V, \forall < i,j > \& < h,i >: i || j \in I_{Node} || < i,j >= D_v \end{aligned} \tag{9}$$

In constraint (9), if one of the links (either $< h,i >$ or $< i,j >$) is not on the route of vehicle v, then the above inequality is always satisfied. Otherwise, the inequality reduces to $X^v_{<i,j>} \geq T_{Delay} + T_{Road} + X^v_{<h,i>}$.

Finally, constraint (10) considers delaying when multiple vehicles share the same link. Let $N^v_{<i,j>}$ indicate the number of vehicles that share the link $< i,j >$ with vehicle v at the same time. As more vehicles share the same link, the speed of vehicles on the shared link is slower, and hence more time is required to get to the end of the link. Now, let $C_\%$ be the percentage of added time delay based on the number of vehicles sharing the same link at the same time. Then the constraint can be mathematically written as:

$$\begin{aligned} X^v_{<i,j>} + B(1 - R^v_{<h,i>}) + B(1 - R^v_{<i,j>}) \\ \geq X^v_{<h,i>} + T_{Road} + C_\% * (N^v_{<i,j>} * T_{Road}) \qquad \forall v \in V, \\ \forall < i,j > \& < h,i >: i \& j \notin I_{Node} \& < i,j > \neq S_v || D_v \end{aligned} \tag{10}$$

The number of vehicles that share link $< i,j >$ at the same time with vehicle v (i.e., $N^v_{<i,j>}$) is derived in subsection IV-C.

C. Multi-Vehicle Constraints

Let $\aleph^{v,v'}_{<i,j>} \in \{0,1\}$ indicate whether vehicle v' shares link $< i,j >$ at the same time with vehicle v. Then the number of vehicles that share the same link at the same time with vehicle v is given by:

$$\begin{aligned} N^v_{<i,j>} = \sum_{v' \in V: v' \neq v} \aleph^{v,v'}_{<i,j>} \qquad \forall v \in V, \\ \forall < i,j >: i \& j \notin I_{Node} \& < i,j > \neq S_v || D_v \end{aligned} \tag{11}$$

Note that in the above equation, we are not interested in counting the number of shared vehicles on links that are either the source or destination of vehicle v or adjacent to an intersection. Consequently, we set $N^v_{<i,j>} = 0$ for such links as follows:

$$\begin{aligned} N^v_{<i,j>} = 0 \qquad \forall v \in V \\ \forall < i,j >: i || j \in I_{Node} || < i,j >= S_v || D_v \end{aligned} \tag{12}$$

For all other links, $\aleph^{v,v'}_{<i,j>} = 1$ when vehicle v' is located on link $< i,j >$ at the same time when vehicle v is traversing the link. In mathematical form, it can be written as:
$\aleph^{v,v'}_{<i,j>} = 1$, if

$$\sum_{<h,i> \in L} X^v_{<h,i>} \leq \frac{\sum_{<h,i> \in L} X^{v'}_{<h,i>} + X^{v'}_{<i,j>}}{2} \leq X^v_{<i,j>},$$

$$\forall v \& v' \in V, \forall < i,j >: i \& j \notin I_{Node} \& < i,j > \neq S_v || D_v.$$

The above inequalities indicate that vehicle v and v' via one of the links in $\sum_{<h,i> \in L}$ arrive at node i and pass through the common link $< i,j >$, which can be linearized as follows:

$$\begin{cases} \aleph l^{v,v'}_{<i,j>} \geq \frac{X^v_{<i,j>} - \frac{\sum_{<h,i> \in L} X^{v'}_{<h,i>} + X^{v'}_{<i,j>}}{2}}{B} \\ \aleph l^{v,v'}_{<i,j>} \leq \frac{X^v_{<i,j>} - \frac{\sum_{<h,i> \in L} X^{v'}_{<h,i>} + X^{v'}_{<i,j>}}{2}}{B} + 1 \end{cases}$$
$$\forall v \& v' \in V, \forall < i,j >: i \& j \notin I_{Node} \& < i,j > \neq S_v || D_v \tag{13}$$

$$\begin{cases} \aleph g^{v,v'}_{<i,j>} \geq \frac{\frac{\sum_{<h,i> \in L} X^{v'}_{<h,i>} + X^{v'}_{<i,j>}}{2} - \sum_{<h,i> \in L} X^v_{<h,i>}}{B} \\ \aleph g^{v,v'}_{<i,j>} \leq \frac{\frac{\sum_{<h,i> \in L} X^{v'}_{<h,i>} + X^{v'}_{<i,j>}}{2} - \sum_{<h,i> \in L} X^v_{<h,i>}}{B} + 1 \end{cases}$$
$$\forall v \& v' \in V, \forall < i,j >: i \& j \notin I_{Node} \& < i,j > \neq S_v || D_v \tag{14}$$

$$\begin{cases} \aleph^{v,v'}_{<i,j>} \leq \frac{\aleph l^{v,v'}_{<i,j>} + \aleph g^{v,v'}_{<i,j>}}{2} \\ \aleph^{v,v'}_{<i,j>} \geq \aleph l^{v,v'}_{<i,j>} + \aleph g^{v,v'}_{<i,j>} - 1 \end{cases}$$
$$\forall v \& v' \in V, \forall < i,j >: i \& j \notin I_{Node} \& < i,j > \neq S_v || D_v \tag{15}$$

V. BNART: The Proposed Method

Because the optimization model is very complex and the problem of minimizing traffic congestion thus the traveling time of all autonomous vehicles in the network has to be solved every time a new vehicle enters the system, a scalable heuristic method should be introduced that can solve the problem in real-time. Hence, in this section, we propose our method called BNART (Best Neighbor Algorithm for Routing Traffic), which finds the best neighbor node for any vehicle that reaches an intersection node to find a time-efficient path. It is very crucial for this algorithm to find the best neighbor in no time because at any second a vehicle or multiple vehicles might reach an intersection node, where the CTCU has to run the algorithm and finds the best neighbor to construct a path for each vehicle.

The BNART method, for any vehicle v that reaches an intersection node n, is described in Algorithm 1. As input, the algorithm takes the network graph $G = (P, L)$, a list of the current location of other vehicles in the network (i.e., W: list of directed edges where vehicles are located), the traversed node list of vehicle v until the last node n (i.e., list A), the destination node of vehicle v, and the dead-end node list

Algorithm 1: BNART: Finding the best neighbor node for vehicle v at current intersection node n.

1 **Input:** Graph $G = (P, L)$, dead-end node list E, current vehicles link location list W, destination node D_v, and the traversed node list A;
2 **Result:** Best neighbor node b;
3 $Neighbor[] \leftarrow$ FindNeighbors(n);
4 $Best \leftarrow L$;
5 $b \leftarrow \phi$;
6 **for** *i ← 1 : size(Neighbor)* **do**
7 **if** $Neighbor[i] \notin A \bigcup E$ **then**
8 $T_{traffic} \leftarrow$ GetTrafficTime($Neighbor[i]$, n, W);
9 $Distance \leftarrow$ GetDistance($Neighbor[i]$, D_v);
10 $NormValue \leftarrow$ Normalized($T_{traffic} + Distance$);
11 **if** $NormValue < Best$ **then**
12 $Best \leftarrow NormValue$;
13 $b \leftarrow i$;
14 W.insert($v, < n, b >$);
15 A.append(b);
16 **return** b;

E. The list E contains a list of nodes that are located at a dead-end path, which if followed, there will be no exit route unless returning to the same traversed nodes. This list for some vehicles is updated if their destination node is within this list.

The algorithm in line 3 starts by finding the neighbor list of node n and stores it in $Neighbor$. Also, the algorithm sets the best neighbor and the calculated value to help in finding the best neighbor as null and a large value respectively. For each neighbor node, if it is not in the traversed node list A and the dead-end node list E (line 7), the algorithm finds the traveling time from node n to the neighbor node based on the criteria explained in Section III-B (line 8). The reason for this *if* condition is to make sure that we do not stack at a dead-end and that a node is not traversed multiple times for loop avoidance. If this condition is met, in line 9, the algorithm also finds the shortest distance from the neighbor node to the destination of the vehicle (D_v) using the breath-first-search algorithm to ensure that we get closer to the destination than choosing a neighbor node that is farther than the current node n. The selection criteria of BNART are based on two factors; distance and link traffic for time efficiency. Consequently, in line 10 we normalize the summation of the distance and the traffic time on the chosen link to give equal weight to both values, and in lines 11-13, the algorithm chooses a neighbor node with the lowest normalized value. Finally, both lists A and W get updated after choosing the best neighbor node.

VI. Performance Evaluation

To evaluate the performance of the BNART approach, in this section, we compare them first with the optimal solutions obtained from the optimization model (optimal) on small instances, and then we evaluate its performance on big instances by comparing it with other heuristic methods such as shortest path obtained using breath-first-search (BFS) and no-loop random path (random) methods. For comparison metrics, we choose the average traveling time and distance by varying different parameters such as the number of vehicles, graph size (number of nodes), and graph density (number of links). For evaluation, we let the size of each road segment be one kilometer with a maximum speed of 72 kilometers per hour. The number of intersection nodes has been chosen as 20% of the total nodes in the graph. The time delay T_{Delay} is set to 3-time units, and the added delay time $C\%$ for sharing a link with other vehicles is assigned as 1% for each vehicle sharing the link. For example, if 3 vehicles are sharing a link, then $C\% = 3\%$. The results are averaged over 100 runs, where in each run, the source and destination of each vehicle have been chosen randomly but equally for different methods for fair comparison.

A. Evaluation over small instances

In this subsection, we compare the proposed method (BNART) and other state-of-the-art solutions such as the breath-first-search (BFS) method and the no-loop random path (random) method with the optimization model (optimal) and analyze their performance gaps to the optimal solution. We start by showing the performance of different methods in terms of average traveling time and distance by varying the number of vehicles in Fig. 2. As shown in the figure, increasing the number of vehicles in the graph causes an expected growth in road congestion and hence an increase in the average vehicle traveling time. However, the number of vehicles increase does not affect the traveling distance for both BFS and random methods, because no routing paths have been changed, whereas in the case of the BNART and the optimal solution, due to finding alternative paths to avoid road congestion, an increase in the traveling distance is observed. The figure also shows that the BNART method, in contrast to the other heuristic methods, performs close to the optimal solution with a performance gap range of 2.5% to 11.6%, versus the BFS and random methods with the performance gap ranges of 14.7% to 20.5, and 37.5% to 44.3%, respectively. As for the average travel distance, since the objective of the optimization model is not to minimize the travel distance, but rather to minimize the travel time, the BFS method outperforms the other methods since it does not concern with road congestion and finds the shortest path for each vehicle from its source to destination. Whereas, the random method performs the worst among others in both performance metrics: travel time and distance.

In Fig. 3, the performance of all methods is compared by varying the size of the graph (number of nodes). As illustrated in the figure, as expected the optimization model results in the lowest average travel time, followed very close by the BNART method, and the random method comes at last with the highest travel time. Whereas, the BFS method as expected results in the shortest distance compared to other methods due to the reason explained above. It is to be noted that the BNART

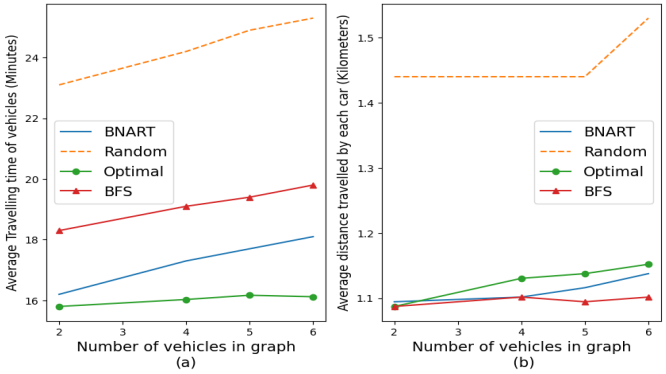

Fig. 2. Performance evaluation of different heuristic methods (BNART, Random, BFS) and the optimal solution by varying the number of vehicles with respect to (a) average traveling time, and (b) average traveled distance.

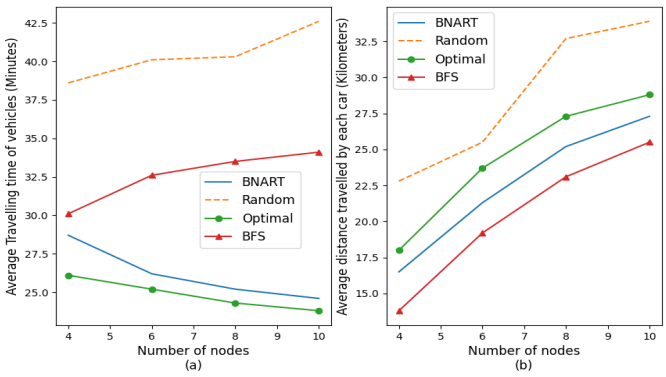

Fig. 3. Performance evaluation of different heuristic methods (BNART, Random, BFS) and the optimal solution by varying the number of nodes with respect to (a) average traveling time, and (b) average traveled distance

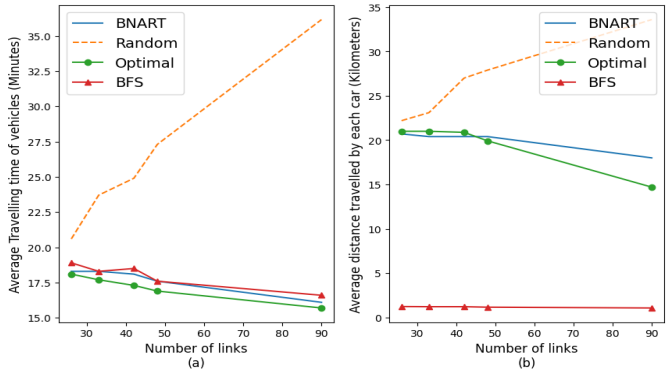

Fig. 4. Performance evaluation of different heuristic methods (BNART, Random, BFS) and the optimal solution by varying the number of links with respect to (a) average traveling time, and (b) average traveled distance

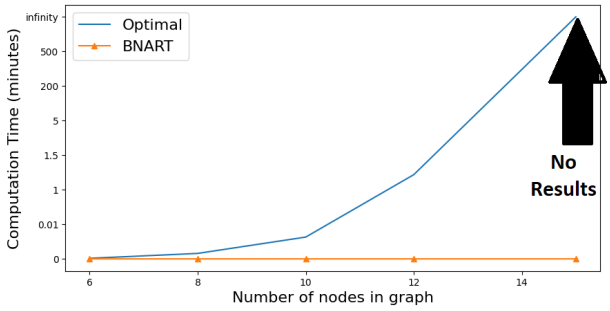

Fig. 5. Computation time of optimization model to obtain the optimal solution versus the BNART approach by varying the graph size (number of nodes in the graph).

method not only performs well in the vehicle traveling time, it also performs well in the vehicle travel distance. That is because the BNART algorithm considers minimizing traveling distance and time together.

In Fig. 4, the performance of different methods is compared with the density of the graph where the total number of links in the graph is varied from 30 to 90 links. As shown in the figure, all the methods except the random method result in lower traveling distance and time as the number of links increases due to the availability of more routes for each vehicle in the graph to reach their destination. However, in the random method, due to the randomness and having more links to traverse in a random path, this method performs very badly and results in increasing the traveling time. As can be noted from the figure, the BNART method performs very close to the optimal solution in the average traveling time with the worst performance gap of 4.5%, and in the best case, it results in an optimal solution when having few links in the graph.

Fig. 5 illustrates the computation time of the optimization model (optimal) versus the BNART method. The graph shows that the computation time of the optimization model increases exponentially, starting with a few milliseconds for a very small graph with 6 nodes, increasing to around 1.5 minutes and 5.5 hours respectively for graphs of size 12 nodes and 14 nodes, and eventually no results (failed to obtain results) for a slightly larger graph of 16 nodes after running for more than 2 days due to complexity of the optimization model and lack of memory. In contrast, the BNART method obtains results in very few milliseconds for the same size graph. To compare the performance of our BNART method on large graphs, in the next subsection, we will exclude the optimization model and compare the performance of the heuristic methods in terms of average traveling time and distance.

B. Evaluation over large instances

In this subsection, to evaluate the performance of different heuristic methods, we use the same performance metrics and parameters as explained in the previous subsection, however using larger graphs. Fig. 6 illustrates the performance of all heuristic methods in terms of average traveling time and distance by varying the number of vehicles in the network. As shown in the figure, the BNART method outperforms the other methods in routing the vehicles to their destinations in the lowest time possible. Whereas, the BFS method results in better performance in terms of the average travel distance. As explained earlier in the above subsection, the reason is that the latter method finds the shortest path from the source to

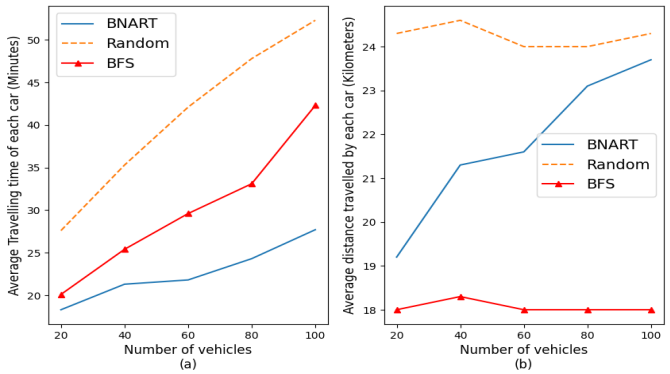

Fig. 6. Performance evaluation of different heuristic methods (BNART, random, and BFS) by varying the number of vehicles concerning (a) average traveling time, and (b) average traveled distance.

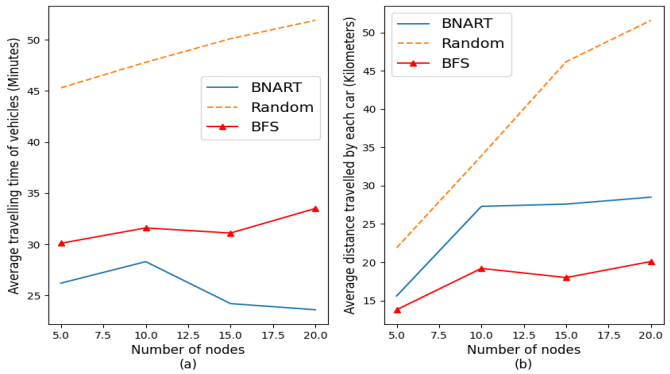

Fig. 7. Performance evaluation of different heuristic methods (BNART, random, and BFS) by varying the number of nodes in the graph concerning (a) average traveling time, and (b) average traveled distance.

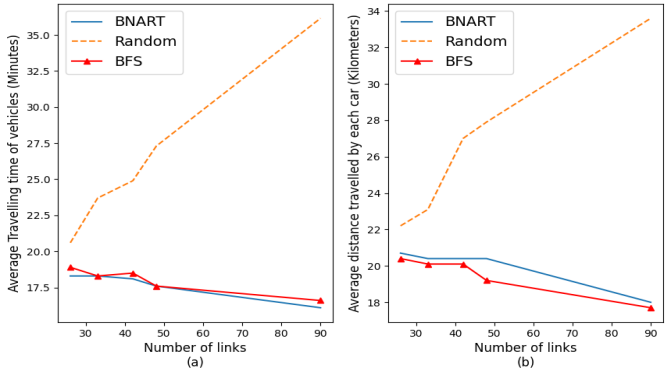

Fig. 8. Performance evaluation of different heuristic methods (BNART, random, and BFS) by varying the number of links in the graph concerning (a) average traveling time, and (b) average traveled distance.

the destination for each vehicle in the network neglecting the traffic congestion, whereas the former method, apart from the shortest distance, considers the road traffic in finding the best route. On the other hand, the random method performs worst that the other two methods in terms of the traveling distance and time due to choosing a random route without considering the shortest path or/and the traveling time although it does not stack in a loop.

In Fig. 7, the BNART method is compared to other heuristic methods for different graph sizes. As depicted in the figure, in terms of average traveling time, the BNART method surpasses the BFS (shortest path) method with 11% in the worst case and 34.7% in the best case. Similarly, the BNART method outperforms the random method in terms of average traveling time with the worst case of 51.2% and best case of 75%. However, in terms of average traveled distance, the BNART method outperforms the random method by taking on average around 13.88 kilometers less distance, and false behind the BFS method by taking on average around 7 kilometers more distance.

Fig. 8 illustrates the performance of different algorithms to the number of links in the graph (density). It can be noted that both BNART and shortest path methods result in better average traveling time and distance with a denser graph, except for the random method where its performance worsens because of increasing the number of alternative paths and choosing a longer path. From the figure, it can also be observed that as the number of links increases, the average traveling time difference between the BNART and the shortest path methods increases and the difference of the average traveled distance decreases. This shows that the performance of the BNART method is enhanced with the graph density.

VII. Conclusion

This study has addressed the critical challenge of traffic management for self-driving vehicles in modern metropolitan areas, aiming to guide them along the most efficient routes. With the increasing prominence of autonomous cars and the vision of smart cities, the role of the Central Traffic Control Unit (CTCU) in revolutionizing road networks has been highlighted. Equipped with real-time location information and responsible for path planning, the CTCU plays a pivotal role in mitigating road congestion and ensuring prompt arrivals. To address the dynamic nature of the road environment and the random arrival of vehicles, we formulated the problem as a mixed-integer linear programming (MILP) model. Recognizing the complexity of the optimization model, we introduced the Best Neighbor Algorithm for Routing Traffic (BNART) as a heuristic method. Extensive evaluations of our proposed heuristic approach against optimal solutions on small-scale instances, as well as comparisons with other state-of-the-art methods such as the Breadth First Search (BFS) shortest path algorithm on large-scale instances, have demonstrated the effectiveness of our approach. While our proposed method may not surpass the BFS algorithm in terms of the average traveled distance, it outperforms the BFS method in terms of average travel time, exhibiting a negligible gap to the optimal solution. This finding underscores the practical viability of the BNART heuristic method in achieving efficient traffic management for self-driving vehicles. By prioritizing average travel time and considering the dynamic road environment, our approach contributes to enhancing overall travel efficiency and minimizing delays. The results obtained from extensive simulation

experiments provide valuable insights into the performance of our proposed method, validating its effectiveness in real-world scenarios. By optimizing the utilization of self-driving vehicles and considering factors such as road congestion and prompt arrivals, our approach contributes to the broader field of traffic management in metropolitan areas. Furthermore, the combination of the CTCU and the proposed BNART heuristic method offers a promising direction for future research and practical implementation. For future work, we plan to extend our research by modeling the problem as a Markov Decision Process (MDP) and exploring the use of reinforcement learning and/or deep reinforcement learning techniques to further enhance the efficiency of traffic management for self-driving vehicles. These approaches can potentially offer adaptive and dynamic solutions that adapt to changing road conditions and vehicle demands.

REFERENCES

[1] S. Javaid, A. Sufian, S. Pervaiz, and M. Tanveer, "Smart traffic management system using internet of things," in *2018 20th international conference on advanced communication technology (ICACT)*. IEEE, 2018, pp. 393–398.

[2] A. M. Miyim and M. A. Muhammed, "Smart traffic management system," in *2019 15th International Conference on Electronics, Computer and Computation (ICECCO)*. IEEE, 2019, pp. 1–6.

[3] P. Manikonda, A. K. Yerrapragada, and S. S. Annasamudram, "Intelligent traffic management system," in *2011 IEEE Conference on Sustainable Utilization and Development in Engineering and Technology (STUDENT)*. IEEE, 2011, pp. 119–122.

[4] Y. Fan, Q. Zhang, and S. Quan, "A new approach to solve the vehicle routing problem: The perspective of the genetic algorithm combined with adaptive large neighbour search," in *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)*. IEEE, 2021, pp. 331–337.

[5] H. Zhang, Q. Zhang, L. Ma, Z. Zhang, and Y. Liu, "A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows," *Information Sciences*, vol. 490, pp. 166–190, 2019.

[6] G. Jie, "Model and algorithm of vehicle routing problem with time windows in stochastic traffic network," in *2010 International Conference on Logistics Systems and Intelligent Management (ICLSIM)*, vol. 2. IEEE, 2010, pp. 848–851.

[7] S. Zheng, "Solving vehicle routing problem: A big data analytic approach," *IEEE Access*, vol. 7, pp. 169 565–169 570, 2019.

[8] Fuce and Wanghui, "The solving strategy for the real-world vehicle routing problem," in *2010 3rd international congress on image and signal processing*, vol. 7. IEEE, 2010, pp. 3182–3185.

[9] K. I. G. A. Liu K, Li N, "A vehicle routing problem with dynamic demands and restricted failures solved using stochastic predictive control." in *2019 American Control Conference (ACC)*. IEEE, 2019, pp. 1885–1890.

[10] Y. Zhang, L. Hu, and Q. Peng, "A genetic algorithm-based path planning approach for autonomous vehicles," *Journal of Advanced Transportation*, pp. 1–12, 2020.

[11] S. Wang, C. Yang, C. Wang, and W. Wang, "Cooperative traffic control system for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 1, pp. 385–396, 2019.

[12] J. Li, H. Guo, and D. Cao, "Centralized traffic management system for autonomous vehicles with dynamic traffic signal control and route optimization," *Transportation Research Part C: Emerging Technologies*, vol. 122, p. 102951, 2021.

[13] H. Zhou, C. Cai, and S. Wang, "Deep reinforcement learning for traffic signal control and routing optimization in intelligent transportation systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 2, pp. 688–698, 2018.

[14] A. Smith, M. Johnson, and J. Anderson, "Centralized traffic management architecture for autonomous vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 6, pp. 2478–2488, 2020.

[15] L. Yang, Y. Ma, and H. Zheng, "A multi-agent framework for traffic management in autonomous vehicle systems," *Transportation Research Part C: Emerging Technologies*, vol. 105, pp. 630–651, 2019.

[16] J. Liu, S. Yang, and J. Wang, "Routing optimization for electric vehicles with energy constraints in a dynamic transportation network," *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 132–148, 2019.

[17] A. Abdrabou and W. Zhuang, "Probabilistic delay control and road side unit placement for vehicular ad hoc networks with disrupted connectivity," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 1, pp. 129–139, 2010.