Smart Driving Methodology for Connected Cars

Ioan Stan, Vasile Suciu, Rodica Potolea
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
ioan.stan@cs.utcluj.ro, vasile.suciu@student.utcluj.ro, rodica.potolea@cs.utcluj.ro

***Abstract*—Time is one of the life's treasure we get and can't be bought or recover. Nowadays, many of us spend a lot of time in the cars and sometime lose it in traffic congestion because of several reasons that can or can't be controlled. Most of the existing navigation solutions try to improve the driving time of each individual car but doesn't approach the improvement of the time spent in traffic for the entire navigation ecosystem.**

In this paper we define and apply a methodology that systematically approaches car navigation challenges by using the benefit of connected cars information sharing in order to improve the global amount of time spent in traffic without focusing on individual route finding algorithms efficiency.

The proposed methodology is validated by an experiment that simulates traffic congestion on real map data corresponding to an urban area. The results proved that the global amount of time spent in traffic is decreased when connected cars information is used and shared.

***Index Terms*—methodology, connected cars, smart driving, traffic, route planning, data structure**

I. Introduction to Connected Cars Ecosystem

Existing navigation systems use map data and route search algorithms to provide optimal routes for individual cars. The evolution and proof of concept solutions for the Internet of Things draw attention of researchers and companies from several domains. Intelligent Transportation System is such a domain that have a systematic description in [1] as a combination of infrastructure, computing, telecommunications, wireless and transportation technologies. State of the art work on the Intelligent Transportation Systems is reviewed by the authors in [2] and [3]. Moreover, the work in [4] describes and analyses a hierarchical architecture of the Intelligent Transportation System named Vehicle-to-Vehicle-to-Infrastructure (V2V2I).

V2V2I architecture proposes a complete solution for Intelligent Transportation System, a solution that fulfills all the quality factors. One component of the Intelligent Transportation System is the Navigation System that tries to provide optimal routes for specific route requests. Nowadays, most of the existing navigation systems provide optimal routes for singular drivers. This can be optimal when the traffic flow is smooth but, can be a bottleneck in case of traffic congestion. This happens because optimal solutions for singular cases sometimes will introduce time penalization to other cases and moreover, can affect the entire transportation system. From this perspective, communication and data sharing between entities is a key requirement of an Intelligent Transportation System.

The objective of this work is to propose a smart driving methodology based on communication and data sharing between connected cars so that traffic flow in urban areas is improved. The purpose of the methodology is to improve the overall traffic experience of an urban area without focusing on individual route finding efficiency.

In the next section we present several traffic challenges that we want to approach with the proposed smart driving methodology. The third and fourth sections contains the description and design of the proposed smart driving methodology in terms of concepts, route search algorithms and testing scenarios infrastructure. The fifth section shows experimental results as proof of concept for the proposed smart driving methodology. In the last section are discussed the conclusions and future work.

II. Connected Cars Traffic Challenges

A. Traffic Congestion

Every year the total number of worldwide vehicles in use increases linearly. Figure 1 (from [5]) shows the linear evolution of the worldwide number of vehicles in use during 10 years.

The urbanization process highly increases the agglomeration and therefore, traffic congestion in urban areas is expected to increase even more in future [6]. In general, the existing roads and infrastructure of urban areas can't be changed because of spacial limitations. A direct consequence of this is the increasing of time spent in the car for each driver. Table I shows the most congested areas in the world where drivers spend from 6 full days to 11 full days in traffic congestion (in some countries this represent the full vacation period for one year).

B. Lack of Valuable Information Usage and Sharing

Each single car normally follows its own route, the route decided by the driver or by the navigation system. In this way

TABLE I
Hours Spent in Traffic Congestion During a Year in Most Congested Urban Areas from the World

Place	Hours/Year/Car
Europe	*over 200*
North America	*over 150*
South America	*over 250*
Oceania	*over 150*

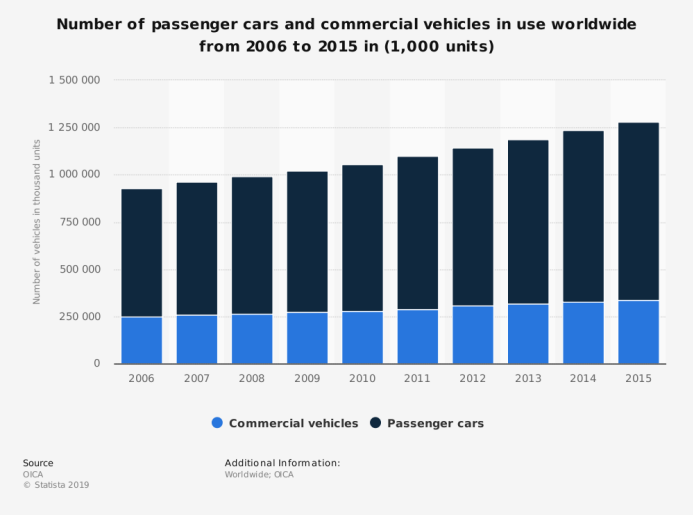

Fig. 1. Statistics of Vehicle In Use Between 2006 to 2015 [5]

some roads and intersections of an urban area can become more busy than roads from other drive-able areas in the same urban zone.

This issue is partially a limitation of the navigation systems/knowledge of the driver and is happening because of lack of valuable information usage and sharing. Most of the existing navigation systems are aware only of the position of their navigating car but not of the whole navigation ecosystem.

C. Solution Execution Cost

Supposing that the cars' positions and the global mechanism for coordination is implemented, it is very important to reduce the risk of deploying a solution that uses such information. Therefore, a simulated environment would help to estimate the benefits and limitations. Currently, to the best of our knowledge, there is no tool that simulates tens of thousands of cars that navigate on a real map data in an urban area. The lack of such tools increases the associated risks of deploying a solution that is not reliable for a specific zone (in [7] is presented an example where more than 300 millions pounds were used to implement and deploy a solution that made the traffic worse).

III. Smart Driving Design

Most of the existing navigation systems use non-cooperative algorithms to generate routes between start and destination points. In this way for each route request the algorithms try to find the best individual route without taking into account the effect of that route on the entire navigation ecosystem.

Considering the challenges presented above, in this section is described a methodology proposal that tries to improve the navigation ecosystem through car data sharing (position) in order to obtain an efficient route planning solution that balances the requested routes distribution on the roads so that traffic congestion is predicted and avoided as much as possible.

A. Route Planning Model

Route computation algorithms are using several parameters used to efficiently describe the input and output of the context. Below are described the main parameters and concepts we considered fore route computation in the context of connected cars:

- $\mathbf{p_s}$ - starting point of a route given as GPS coordinates
- $\mathbf{p_d}$ - destination point of a route given as GPS coordinates
- $\mathbf{segmentID}$ - segment on the navigation map represented by segment ID
- $\mathbf{C_{map}}(\mathbf{segmentID})$ - map cost value of a segment from the map (e.g. euclidean distance, turn costs, speed limit on the segment, etc.)
- $\mathbf{R}(\mathbf{p_s}, \mathbf{p_d})$ - fastest generated route between starting point p_s and destination point p_d
- $\mathbf{N}(\mathbf{segmentID})$ - set of neighbour segments of segmentID
- $\mathbf{visited}(\mathbf{segmentID})$ - returns TRUE if segmentID was visited in the routing algorithm search graph and FALSE otherwise
- $\mathbf{C_{max}}$ - constant representing the maximum cost value possible on the map (used to limit the forward search exploration algorithm)
- $\mathbf{carID}$ - car represented by an ID. It corresponds to a route $R(p_s, p_d)$
- $\mathbf{R}(\mathbf{carID})$ - route on which a car is supposed to follow during navigation
- $\mathbf{Routes}(\mathbf{t})$ - routes set generated until a specific time
- $\mathbf{Segments}(\mathbf{R}(\mathbf{carID})$ - set of the segments of a route
- $\mathbf{lanesCount}(\mathbf{segmentID})$ - number of lanes on a segment used for car density computation
- $[\mathbf{t_s}(\mathbf{carID}, \mathbf{segmentID}), \mathbf{t_e}(\mathbf{carID}, \mathbf{segmentID})]$ - predicted time interval when a car is on a segment represented by segment ID
- $\mathbf{carsCount}(\mathbf{seggmentID}, \mathbf{t})$ - predicted number of cars on a segment at a specific time used for car density computation
- $\mathbf{S_{avg}}(\mathbf{segmentID}, \mathbf{t})$ - average speed on segments at a specific time
- $\mathbf{C_{cars}}(\rho(\mathbf{segment}, \mathbf{t}))$ - cost factor corresponding to predicted car density on a segment at a specific time
- θ - threshold value that indicates traffic congestion
- $\mathbf{length}(\mathbf{segmentID})$ - length of a segment from a planned route
- $\mathbf{carsCount}(\mathbf{segmentID})$ - total number of connected cars that navigates through a specific segment on the map

- $\mathbf{carLength}$ - the average length of a car. In this work we considered to be **7 meters**
- predicted car density on a segment at a specific time $\rho(\mathbf{segmentID}, \mathbf{t})$ defined as

$$\frac{\mathbf{carsCount}(\mathbf{segmentID}, \mathbf{t}) \cdot \mathbf{carLength}}{\mathbf{length}(\mathbf{segmentID}) \cdot \mathbf{lanesCount}(\mathbf{segmentID})}$$

B. Traffic Data Representation

Car information representation is one key element that is used during route planning mechanism. In our case car information means traffic data and is represented by the position of the car on a real map data segment during a period of time. The route planning mechanism tries to compute a route based on several parameters. Some of the parameters are estimated by applying different queries on traffic data.

Below are the queries we considered relevant to achieve our objective of efficiently using and sharing real time traffic information:

- *How many cars pass a road during any specific time interval ?*
- *What is the maximum number of cars on a road during any specific time interval ?*
- *How many cars are on a road at any specific time ?*

The answers and efficiency of responding to queries depends on the data structure that is used to represent the traffic data.

The above queries are of two types: interval queries and specific moment query. Both query types can be considered range queries (the specific moment query is a range query with an interval that has both bounds equal). There are several data structures that support range queries. In this work we considered efficient for our purpose the Segment Tree and K-ary Tree data structures to achieve our main objective of efficiently using and sharing connected cars data.

Segment Tree is a height-balanced binary tree that can be used to perform range queries and range updates. It works like a tree data structure where each node represents a segment of the data set that is composed recursively by the segments of the data set contained in its children. The leaves contain atomic values from the data set [8].

K-ary Tree is a rooted tree data structure where each node has no more than k indexed children [9].

The basic operations on both Segment Tree and K-ary Tree data structures are:

- **Query** - queries cars presence on a segment in time
- **Update** - updates cars presence on a segment in time

Based on the above queries and basic operations, in order to achieve our objective we need to define specialized operations on Segment Tree and K-ary Tree:

- **QueryCarsPassing** - queries how many cars navigate on a segment in a time interval [t1, t2]
- **UpdateCarsPassing** - each time a new car is navigating on a segment during time interval [t1, t2], increases by 1 the number of navigating cars on that segment for the time interval [t1, t2].
- **QueryCarsMaxCount** - queries maximum number of cars that navigate on a segment during time interval [t1, t2]
- **UpdateCarsMaxCount** - each time a new car is navigating on segment during time interval [t1, t2], increases with 1 maximum number of cars that navigate on a segment during the given interval [t1, t2]
- **QueryCarsCount** - queries how many cars are on a segment at a specific moment t (when a new car is entering on a segment)
- **UpdateCarsCount** - when a new car is entering on a segment at a specific moment t, increases with 1 the cars count and when the car is leaving the segment decreases with 1 the cars count

With all these operations we can answer the required traffic queries and therefore, they are necessary and sufficient for our objective of getting and updating efficiently real time traffic information.

Each data structure operations takes $O(\log n)$ but, each implementation have different multiplicative complexity constants that impact considerable the runtime performance. Moreover, each implementation have different impact on the memory footprint.

C. Route Search Algorithms

Route search algorithms use in general the same concepts and can have different flows in order to fulfill the requirements of their running context. There are several route finding algorithms used in the navigation systems. Some of them are well known in the literature: Dijkstra, Bidirectional Search A* Algorithm, Forward Oriented Search A* Algorithm.

It is worth to mention that in [10] are described specific and efficient route finding algorithms. Our purpose is to use existing route finding algorithms and apply them in order improve the overall traffic efficiency in urban areas.

The algorithm presented in [11] is an elegant combination of the reach-based algorithm described in [12] and A* graph search algorithms.

One important aspect of the routing algorithms represent the highway hierarchies concept that is used to improve their computation time. In [13] is presented and analyzed the effect of using highway hierarchies for many-to-many shortest route calculation.

The Route Search Algorithm has a main role in achieving the objective of reducing the global driving time in a context of connected cars. In terms of Route Planning this is translated into reducing the average of the Estimated Time of Arrival (ETA) for connected cars.

In this section we present forward oriented approach of a connected cars route search algorithm based on A*. The core element of the algorithm is the usage and sharing of the real time cars' positions during route computation. In this

Algorithm 1 Forward Oriented Search A* Algorithm

Data: $p_s, p_d, map\_data, connected\_car\_data, \theta$

Result: $R(p_s, p_d)$

▷ initialize forward graph search cost queue

$init(forwardQueue, p_s)$

▷ initialize backward graph search cost queue

$init(backwardQueue, p_d)$

$backwardSteps \leftarrow 10$

while *forward search unmet all backward processed segments* **do**

  $forwardHead \leftarrow forwardQueue.head()$

  $backwardHead \leftarrow backwardQueue.head()$

  if $backwardSteps == 0$ ***OR*** $forwardHead < backwardHead$ **then**

    $segmentID \leftarrow forwardQueue.pop()$

  end

  else

    $segmentID \leftarrow backward.pop()$

    $backwardSteps \leftarrow backwardSteps - 1$

  end

  $processSegment(segmentID)$

end

way, when a route is computed it takes into consideration the presence of the active navigating cars on the roads in order to avoid traffic congestion by smartly choosing alternative routes if necessary.

In the first step we initialize 2 priority queues: for forward and backward search. The backward priority queue is used for at most 10 steps during the search algorithm in order to ensure that the destination point is reached. Most of the steps are running on forward search and therefore, we name the flow Forward Oriented Search A* Algorithm.

While forward search unmet all backward processed segments (in at most 10 steps) the algorithm considers both priority queues to choose the segment with the smallest cost and then processes it using the real time predicted traffic on that segment at a specific time (see Algorithm 2). The key elements in the Algorithm 2 are the real time predicted traffic on a segment at a specific time and the condition that predicts the congestion on a segment and tries to force the navigation on another segment in order to avoid the congestion. The traffic prediction and congestion avoidance is based on the K-ary Tree data structure mentioned above.

IV. Scenarios Simulation

In most engineering domains, before deploying a solution it is important to be tested and validated on different scenarios in order to avoid risks of deployment costs and issues fixing in production(see [7]). This requirement is also necessary in this work and therefore, we propose different testing scenarios that ensures fulfillment of quality factors.

Algorithm 2 Process Segment

processSegment(segment)

Data: $segment$

Result: $updatedcost$

foreach *segmentID in* $N(segment)$ **do**

  if [***not*** $visited(segmentID)$ **then**

    $C_{map} \leftarrow C_{map}(segmentID)$

    $t \leftarrow$ *predicted time when the car arrives on segment*

    if $\theta < \rho(segmentID, t)$ **then**

      $C_{cars} \leftarrow C_{max}$ {force to try another segment}

    end

    else

      $C_{cars} \leftarrow C_{cars}(\rho(segmentID, t))$

    end

    $costValue \leftarrow C_{map} \cdot C_{cars}$

    $updateCost(segmentID, costValue)$

  end

end

A. Testing Scenarios Classification

In [14] are classified and analyzed different traffic scenarios as shown below:

- **Navigation area**: urban and non-urban
- **Length of the requested route**: short, medium and long
- **Size of the city where navigation happens**: small, medium and big

Based on this classification a solvable **Core Scenario** is found, representing medium and long routes requests in medium and big cities.

For the urban areas we consider that is valuable to run tests on cities that have different types of road topology as follows:

- **Cities in the form of a grid**: New York, Barcelona
- **Historical cities with irregular road topology**: Cluj-Napoca, Rome
- **Historical cities with irregular road topology and modern infrastructure combined**: Paris, Vienna

B. Testing Scenarios Setup

For testing urban traffic we propose to have areas of diameters between **10 kilometers to 20 kilometers**.

Testing scenarios should use routes of different lengths that can have between **1 kilometer to 20 kilometers** and have random start and destination points situated in different zones defined as rectangles that doesn't overlap (e.g. the 6 main districts in Cluj-Napoca).

The route requests must be done in a short period of time in order to simulate peak traffic hours when happens traffic congestion (e.g. 10.000 route requests in 150 seconds).

In [15] the authors use traffic classification based on different *simulated road topology* in order to predict and reduce the traffic congestion. Our proposal is based on *real map data road topology*.

With such intensive testing scenarios it is very important to have an efficient (in terms of runtime and memory usage) simulation tool that efficiently generates route requests that simulates real routes on real map data.

V. Proof of Concept Experimental Results

In order to validate the approach and objective achievement of the proposed methodology we run one experiment by simulation traffic in a real map data topology (e.g. Cluj-Napoca city map).

A. Measurement Infrastructure and Metrics

The proposed approach in this paper was validated and measured by using OSMAnd open source application adapted for our objective. It is an open source navigation solution implemented in Java that can run on Android, iOS and desktop platforms [16]. OSMAnd is based on Open Street Map (OSM) data. We changed the application's Route Search Algorithm in order to consider traffic data represented by connected cars.

For our objective we found valuable to use the following metrics for testing measurements:

- **Histogram for Lengths of the Navigated Segments**
- **Estimated Time of Arrival (ETA)**
- **Average Speed of a Car**

B. Measurement Results on Cluj-Napoca

Our measurement was done in the city of Cluj-Napoca in Romania, by requesting 10.000 random routes inside the city. In order to simulate traffic congestion the time that passes between first car route request and last car route request was 150 seconds. The start and destination points are inside the 6 main districts of Cluj-Napoca and the length of the requested routes are between 4 kilometers to 20 kilometers.

We considered that all the cars follow the route that the algorithm provides to them. The chart in figure 2 represents the histogram for the lengths of the navigated segments in both cases: Basic Routing Algorithm and Connected Cars Routing Algorithm (this names were firstly used in [17]). It can be observed that the Connected Cars Routing Algorithm line is almost all the time above the line representing Basic Routing Algorithm. This shows that comparing with Basic Routing Algorithm, Connected Cars Routing Algorithm tries more alternative segments in order to balance the traffic flow and to avoid traffic congestion.

Because the number of segments with length greater than 800 meters is insignificant, the chart 2 contains only the segments that have length up to 800 meters. The total number of navigated segments in case of Basic Routing Algorithm is **2381** which is lower than the total number of navigated segments in case of Connected Cars Routing Algorithm that is **2461**.

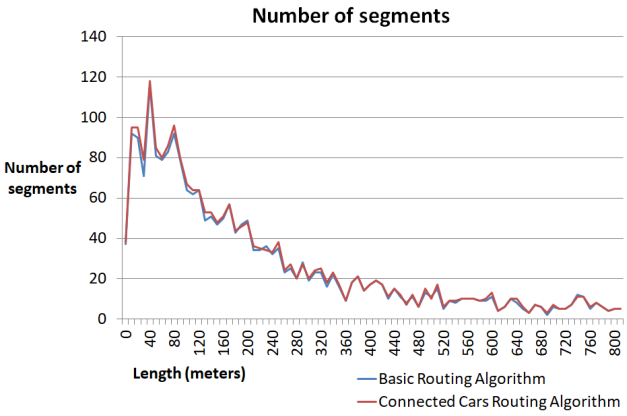

Fig. 2. Segments' Lengths Histogram

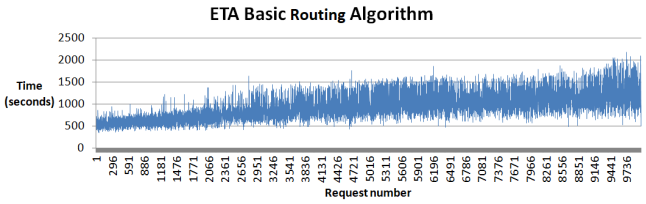

Fig. 3. Estimated Time of Arrival Basic Routing Algorithm

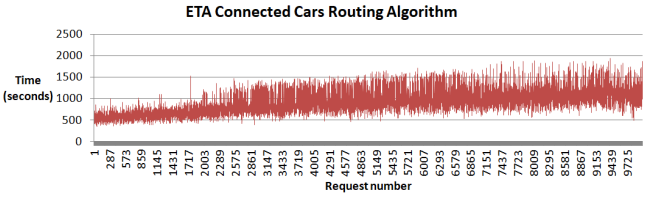

Fig. 4. Estimated Time of Arrival Connected Cars Routing Algorithm

Figure 3 shows ETAs computed for all the routes by using the Forward Oriented Search A* Algorithm that doesn't try to predict and avoid traffic congestion but, tries to provide fastest singular routes. We can observe that ETAs starts below 500 seconds and can reach values greater than 2000 seconds for some routes. The obtained average ETA for this case is **931 seconds**.

The graph from figure 4 shows ETAs computed for all the routes by using the Forward Oriented Search A* Algorithm that tries to predict and avoid traffic congestion in order to improve the total driving time for the entire connected cars ecosystem (for all navigating cars). In this case the ETAs starts below 500 seconds and have the up limit below 2000 seconds proving an improvement on the driving time. The obtained average ETA for this case is **908 seconds**.

In both figure 3 and figure 4, the increasing of the signal value proves that the ETA increases with the number of navigating cars.

For our testing scenario the connected cars shared data (position) helps to predict and avoid traffic congestion and improves the global driving time with **64 hours** in total and

the average driving for each car with **23 seconds**.

The average speed of a car in case of Basic Routing Algorithm is about 37 km/h and in case of Connected Cars Routing Algorithm is about 39 km/h.

In figure 4 can be observed that there is no group of values with ETA greater than 2000 seconds comparing with figure 3. This shows that besides optimizing the route average ETA with 23 seconds, the optimization becomes more obvious when the traffic congestion increases and longer driving time routes appears (improvement of more that 350 seconds). Therefore, besides improving the ETA of the entire ecosystem the proposed methodology helps on improving the ETA for routes that are part of traffic congestion and have long ETAs.

VI. Conclusion and Future Work

In this paper we defined a smart driving methodology that considers the information usage and sharing between connected cars (car position) in order to improve the global driving time and to avoid traffic congestion by smartly following alternative routes.

In the first part we presented an introduction on the connected cars ecosystem as part of the Internet of Things concept and discussed the related work in this domain.

We analyzed the existing connected cars challenges from the traffic congestion perspective and we found several reasons of the traffic issues that drivers encounter nowadays:

- Traffic Congestion
- Number Vehicles in Use and Infrastructure
- Lack of Valuable Information Usage and Sharing
- Solution Execution Cost

Due to such traffic issues, some people spend an amount of time in traffic congestion equal with their vacation period for one year.

Our solution was validated based on an existing open source application OSMAnd [16] that we changed in order to consider connected cars information (traffic data) during route search algorithm. The key element of the proposed solution is the ability to answer efficiently range queries regarding car positioning in time. For this purpose we considered Segment Tree and K-Tree data structures for car positioning representation. To ensure that all quality factors are respected and the deployment cost risks of the solution are reduced, we proposed a set of testing scenarios and an efficient simulation approach.

The proof of concept validation for the proposed methodology was done by running a test that simulates traffic congestion by generating in 150 seconds 10.000 car routes on a real map data in the city of Cluj-Napoca. The results showed that car sharing information (car position) can improve the total driving time of the ecosystem with 64 hours and therefore, driving experience can be improved when a systematic methodology is used in the connected cars context. In this way we achieved our objective of having a validated smart driving methodology that improves traffic flow in connected cars context through real time traffic data usage and sharing.

The work in this paper can be continued in several ways in future. For Route Search Algorithms we can try different approaches and compare the results in terms of results quality and performance (CPU and memory). Connected cars data structures representation is one of the main topics that can be deeply analyzed and evaluated in terms of runtime and memory usage (e.g. different variants of K-ary tree implementation). Testing on different scenarios will help us to better calibrate the Routing Search Algorithm for providing more qualitative results, discovering new traffic behaviours and to be able to use machine learning for Route Search Algorithm parameters calibration.

References

[1] F. Martinez, C.-K. Toh, J.-C. Cano, C. Calafate, and P. Manzoni, "Emergency services in future intelligent transportation systems based on vehicular communication networks," vol. 2, pp. 6–20, 06 2010.

[2] L. Figueiredo, I. Jesus, J. A. T. Machado, J. R. Ferreira, and J. L. M. de Carvalho, "Towards the development of intelligent transportation systems," in *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No.01TH8585)*, 2001, pp. 1206–1211.

[3] P. Papadimitratos, A. D. L. Fortelle, K. Evenssen, R. Brignolo, and S. Cosenza, "Vehicular communication systems: Enabling technologies, applications, and future outlook on intelligent transportation," *IEEE Communications Magazine*, vol. 47, no. 11, pp. 84–95, November 2009.

[4] J. Miller, "Vehicle-to-vehicle-to-infrastructure (v2v2i) intelligent transportation system architecture," *2008 IEEE Intelligent Vehicles Symposium*, pp. 715–720, 2008.

[5] "Statistics of vehicle in use between 2006 to 2015," accessed: 2019-05-12. [Online]. Available: https://www.statista.com/statistics/281134/number-of-vehicles-in-use-worldwide/

[6] "Urbanization," accessed: 2019-05-12. [Online]. Available: https://ourworldindata.org/urbanization

[7] "Report: 300m traffic jam-busting scheme made some journeys longer," accessed: 2019-05-13. [Online]. Available: https://www.aol.co.uk/news/2019/04/07/a-300m-traffic-jam-busting-scheme-made-some-journeys-longer-a/

[8] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, *Computational Geometry: Algorithms and Applications*, 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Third Edition*, 3rd ed. The MIT Press, 2009.

[10] P. Sanders and D. Schultes, "Engineering fast route planning algorithms," in *Experimental Algorithms*, C. Demetrescu, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 23–36.

[11] A. V. Goldberg, H. Kaplan, and R. F. Werneck, "Reach for A*: Efficient point-to-point shortest path algorithms," in *IN WORKSHOP ON ALGORITHM ENGINEERING EXPERIMENTS*, 2006, pp. 129–143.

[12] R. Gutman, "Reach-based routing: A new approach to shortest path algorithms optimized for road networks." 01 2004, pp. 100–111.

[13] S. Knopp, P. Sanders, D. Schultes, F. Schulz, and D. Wagner, "Computing many-to-many shortest paths using highway hierarchies," 01 2007.

[14] I. Stan and R. Potolea, "Connected cars traffic flow balancing based on classification and calibration," in *Mining Intelligence and Knowledge Exploration - 6th International Conference, MIKE 2018, Cluj-Napoca, Romania, December 20-22, 2018, Proceedings*, 2018, pp. 177–188. [Online]. Available: https://doi.org/10.1007/978-3-030-05918-7_16

[15] T. Yamashita, K. Izumi, K. Kurumatani, and H. Nakashima, "Smooth traffic flow with a cooperative car navigation system," in *Proceedings of the International Conference on Autonomous Agents*, 01 2005, pp. 478–485.

[16] V. Shcherb, "Osmand," accessed: 2019-05-14. [Online]. Available: https://osmand.net/

[17] I. Stan, D. Toderici, and R. Potolea, "Segment trees based traffic congestion avoidance in connected cars context," *2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP)*, pp. 137–143, 2018.