Comprehensive Performance Analysis and Comparison of Vehicles Routing Algorithms in Smart Cities

∓Shen Wang, ±Soufiene Djahel, ∓Jennifer McManis, ‡Cormac McKenna and ‡Liam Murphy
∓Lero, RINCE, School of Electronic Engineering, Dublin City University, Ireland
±Lero, School of Computer Science and Informatics, University College Dublin, Ireland
‡IBM Software Group, Industry Solutions Development, Ireland
shen.wang4@mail.dcu.ie, soufiene.djahel@ucd.ie, mcmanisj@eeng.dcu.ie, {CORMCKEN, Liam_J_Murphy}@ie.ibm.com

***Abstract*—Due to the severe impact of road traffic congestion on both economy and environment, several vehicles routing algorithms have been proposed to optimize travelers itinerary based on real-time traffic feeds or historical data. However, their evaluation methodologies are not as compelling as their key design idea because none of them had been tested under both real transportation map and real traffic data. In this paper, we conduct a deep performance analysis and comparison of four typical vehicles routing algorithms under various scalability levels (i.e. trip length and traffic load) based on realistic transportation simulation. The ultimate goal of this work is to suggest the most suitable routing algorithm to use in different transportation scenarios, so that it can provide a valuable reference for both traffic managers and researchers when they deploy or optimize a large scale centralized Traffic Management System (TMS). The obtained simulation results reveal that dynamic A* is the best routing algorithm if the TMS has sufficient memory or storage capacities, otherwise static A* is also a great alternative.**

***Keywords* – ITS, Smart Transportation, Vehicles Routing Algorithms, Comparative Study, Shortest Path, Performance Evaluation, Smart Cities.**

I. Introduction

With the trend of worldwide urbanization, an increasing number of vehicles are swarming into city road networks of limited capacity, leading to excessive increase of traffic congestion, road accidents and air pollution. According to the new released report [1] from Texas Transportation Institute, the economic loss due to traffic congestion, in terms of extra travel delay and fuel consumption, is estimated to $121 billion in the United States in 2011. Therefore, scientists and researchers from both industry and academia have proposed the so called "Intelligent Transportation Systems (ITS)" [3], which have already been successfully deployed in many regions in United States, Japan and Europe to ease the traffic management task.

Smart routing of vehicles is one of the most essential services offered by ITS. This service aims to achieve an optimal load balance of the traffic on the roads by leveraging the collected real time road conditions in an efficient and accurate way that helps the driver to find a shortest or fastest route to a given destination. According to the traffic incidents report released by the U.S. Federal Highway Administration, urban traffic accidents lead to about 50% - 60% of overall congestion delays [4]. Therefore, in order to bypass the blocked roads, due to accidents, a real-time vehicles re-routing algorithm should be efficient enough to provide alternative routes to the drivers in a very short time to reduce the accidents impact on their journey.

In general, vehicles routing algorithms are equivalent to the application of shortest path problem (SPP), which is a classic problem in graph theory, to the transportation domain. When we apply SPP in a transportation scenario, the weight value of each edge (i.e. road segment) can refer to several routing metrics according to the drivers demand such as, travel time and fuel consumption, rather than travel distance only. A vehicle route planning problem is usually classified into two main categories; static (shortest) and dynamic (fastest). The former problem consists in finding a shortest path from origin node to destination node using a map with invariable weight values. The most representative solution in this category is Dijkstras Algorithm (DA) [5]. Based on the static algorithms, if we want to take the future change of road conditions (e.g. average travel time on each road segment) into account, then dynamic vehicle route planning algorithms should be applied. The typical dynamic routing algorithm is an improved version of A* for deterministic discrete-time dynamic networks [6].

Obviously, dynamic vehicles routing algorithms are more useful and practical than static ones because the drivers prefer to keep their planned route optimal throughout the whole journey. Hence, enhancing their efficiency is of vital importance for centralized (i.e. centralized decision-making process) Traffic Management Systems (TMS), which are widely deployed and used across the world (e.g. SCATS [7] and SCOOTS [8]). However, for the best of our knowledge, no work in the literature has conducted a deep performance evaluation and analysis of the different vehicles routing algorithms under varying road scenarios, in order to highlight the advantages and drawbacks of each of them. Indeed, this evaluation is necessary as it will enable better understanding of the missing features in the existing algorithms and thus improved algorithms can be designed to deal with the increasing traffic loads in future smart cities.

In the rest of the paper, we first present an overview of vehicles routing algorithms as well as the related works in section II. Afterwards, we describe the simulation environment, the dataset and the comparison methodology we used in our experiments along with the main performance evaluation metrics applied to the four algorithms, in section III. In section IV, we present and analyze in details the obtained simulation results. Finally, the conclusion and future work are outlined in Section V.

II. Related works

A general classification of the existing vehicles routing algorithms is shown in Figure 1 from which we distinguish two main classes of algorithms, static and dynamic. The static algorithms include, amongst others, Dijkstras Algorithm (DA) [5], and its improved version A* [11] in which the Euclidean Distance is introduced as

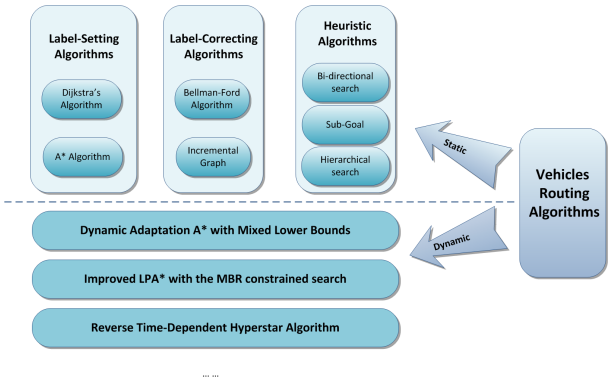


Figure 1: Classification of vehicles routing algorithms

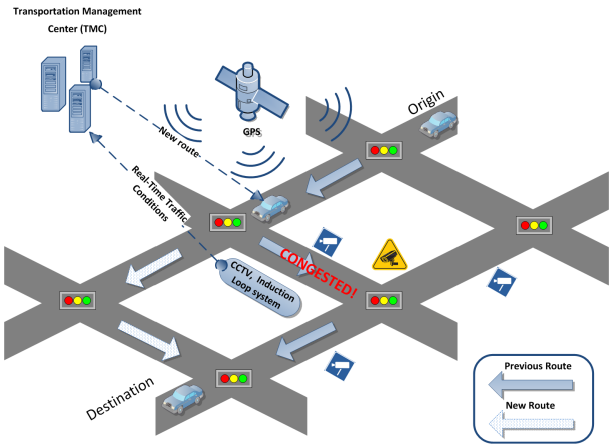


Figure 2: Re-routing scenario in a centralised ITS

the lower bound to ensure it would never overestimate the real travel distance between the origin and the destination nodes. Other algorithms in this class include Label-Correcting (LC) algorithms and some heuristics. The routing algorithms of the second class (i.e. dynamic) consists of Dynamic A*, improved LPA*, etc. These algorithms are more practical in real transportation scenario as the weight of each link in the graph representing the road network is changing over time due to traffic congestion levels variation and random incidents. In what follows, we will briefly present the most significant dynamic vehicles routing algorithms.

In 2002, I. Chabini and S. Lan have proposed a new algorithm [6] called Dynamic Adaptation A* with mixed lower bounds (DAA*_M). DAA*_M is commonly recognized as the cornerstone of dynamic vehicular routing algorithms as it is the first theoretical and experimental contribution to this research field. For the well-known A* algorithm [11], its performance is highly depended on the design of the lower bound, which reflects the quality of the cost estimation function. If this bound is too small, for example approaching to 0, A* will degrade to LS which presents the worst efficiency level. Otherwise, the search scope can be reduced significantly but will not guarantee to find the best solution. The key idea of DAA*_M is to improve the lower bound in the current time interval, according to the shortest path choice in the previous time interval.

In 2007, H. Bo et al. [12] have proposed another dynamic vehicular routing algorithm based on Lifelong planning A* (LPA*) [13]. This algorithm represents a significant progress on shifting the application area of LPA* from the robot grid map to the transportation field. Moreover, the authors have improved it to be applicable to the moving object scenario, and used the minimum bounded rectangle (MBR) to speed up its execution.

The latest noteworthy contribution on dynamic vehicles routing research field, dubbed Hyperstar, has been proposed by M.G.H. Bell et al. in [14], which is an improvement of the authors previous algorithm [15]. Hyperstar is proposed for solving risk adverse vehicle navigation problem based on a well-known transit planning algorithm proposed by Spiess and Florian in [16]. This enhanced algorithm can be considered as a combination of [6] and [16]. The problem it attempts to solve is how to find a fast and reliable route, connecting one pair of origin-destination nodes, that can avoid the road segments in which traffic congestion is too frequent. Instead of finding one path only from origin to destination, this algorithm computes a hyper-path (more than one path from point to point) as the ultimate solution. Therefore, it gives the driver multiple reliable alternative routes when some roads become highly congested or blocked due to random unpredicted on-trip events. Figure 2 illustrates an example of on-trip re-routing based on centralized TMS, equipped with multiple dynamic routing algorithms like Hyperstar, where the TMC updates the drivers initial route upon detection of high congestion level in one of the roads segments belonging to the initial route.

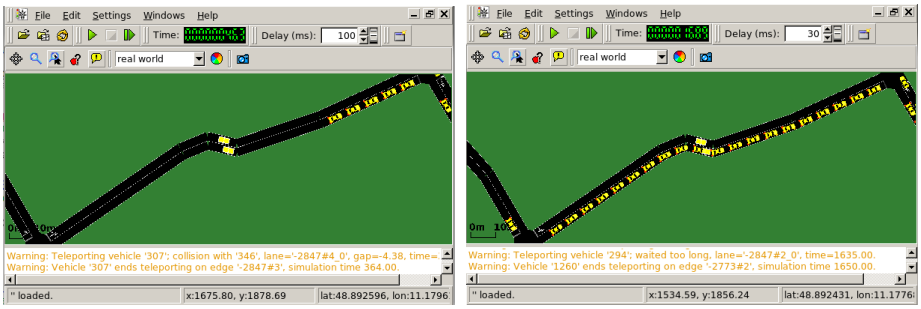
To date, the performance of the above dynamic algorithms has been evaluated under randomly generated road networks only, in which the links are weighted with arbitrary travel time values. This is mainly due to either the limitation of simulation technology in the last decade or real traffic data access issues. Last year, V. T. Ngoc Nha et al. [20] have proposed a general classification of vehicles routing algorithms and described a set of route selection and algorithms evaluation metrics in smart cities, but they didn't provide any testing results. Prior to this work, in 2006, L. Fu et al. [10] have presented a quite comprehensive summary of the heuristic vehicles routing algorithms and their key performance evaluation results, however these results have not been measured using the same testing environment and settings as the authors have collected them from different research papers. Additionally, although the performance of several static vehicles routing algorithms with various data structure implementations had been evaluated in [9] based on different real transportation maps in USA, they measure one metric only, which is the computation time under varying road network sizes and topologies. To conclude, no performance evaluation work has been done so far on both static and dynamic routing algorithms under both various real road transportation maps and traffic flow. Hence, in order to provide valuable reference for engineers and researchers working in the routing field of ITS, we will conduct in the rest of this paper an extensive simulation to assess and compare the performance of four typical algorithms under real city map and various urban roads scenarios.

III. Simulation Setting and Evaluation Methodology

For our comparative study, we have chosen static DA, static A*, dynamic DA and dynamic A* as the four algorithms to be evaluated in the experiments. This choice is justified by the fact that these algorithms have been widely used in several commercial navigation systems as well as some in extensions of ITS. In the sequel, we will present the simulation environment and dataset, explain the different road scenarios to be evaluated as well as our comparison methodology, and finally we present the chosen performance evaluation metrics.

A. Simulation environment setup

We use Simulation of Urban MObility (SUMO) version 0.17.1 [17] to conduct our experiments because it is the most widely used open-



(a) Travel time calculation in SUMO: scenario 1 (b) Travel time calculation in SUMO: scenario 2

Figure 3: Illustration of travel time calculation inaccuracy in SUMO

source microscopic urban transportation simulator. All the algorithms are implemented in Python and then dynamically called to generate routes to be applied to the relevant cars in SUMO via TRACI [18]. Moreover, to ensure a higher probability of realistic testing background, a highly realistic traffic dataset (TAPACologne version 0.17.0), for SUMO, which is made available by the project TAPAS-Cologne [1][19] is used in our experiments. This dataset contains two hours traffic data starts from 6:00am to 8:00am on the weekdays. Besides, all our simulation tests have been conducted using the following configuration: Windows 8 Pro 64-bit, Intel(R) Core(TM) i7-3520M CPU@ 2.9GHz, 8.00GB memory and SSD hard drive.

a) ***Travel time API improvement****:* to test the dynamic algorithms, we have set the update interval of the travel time in each road segment to 30 seconds in order to create a discrete time-dependent network. We have also prohibited the overtaking in order to meet the requirements of first-in-first-out (FIFO) network. Besides, before running our experiments in SUMO, we have checked the correctness and accuracy of travel time calculation API provided by the simulator, as it is mentioned in SUMO website that this API has not been verified yet (see *http://sumo.sourceforge.net/doc/current/docs/userdoc/TraCI/Edge_Value_Retrieval.html*). To this end, we have measured the travel time using SUMO API [2], and found that it is not accurate as the returned value doesn't reflect the real waiting time of a vehicle in different scenarios, thus it may lead to unrealistic travel time estimation. To illustrate this inaccuracy of SUMO API, consider the scenario shown in Figure 3(a) where the road segment is almost empty with few cars only waiting in front of the junction. In this case, a vehicle going straight needs a few minutes of waiting time, however according to SUMO API its travel time would be infinity as its current average speed is zero, which is equivalent to the scenario shown in Figure 3(b). Nevertheless, these two scenarios are totally different, which proves the inaccuracy of SUMO travel time calculation API.

Although, according to the literature, there is a common way defined by the Bureau of Public Roads (BPR) [2] to calculate the travel time for freeways, it cannot, unfortunately, be applied in our simulation because our main focus is on urban roads and not freeways. To overcome this problem, we have proposed and implemented a solution that ensures accurate calculation of travel time in urban scenarios, as described below.

- We divide a road segment according to its occupancy by cars, so for the unoccupied part we use the maximum speed to calculate the travel time while we use average vehicle speed for the occupied part.
- We set minimum vehicle speed equals to 0.1 m/s (inspired from *http://sumo.sourceforge.net/doc/current/docs/userdoc/Simulation/Output/TripInfo.html*, as it defines the "waitSteps" as The number of steps in which the vehicle speed was below 0.1m/s) for the case where all the vehicles on the road are standing still. This is because those vehicles will not stop forever, they are just waiting for the chance (i.e. green traffic light or congestion mitigation in the road ahead) to go.
- Given each simulation step in SUMO lasts 1000ms, we calculate the average travel time every 30 seconds to reduce the statistical error in the further step.

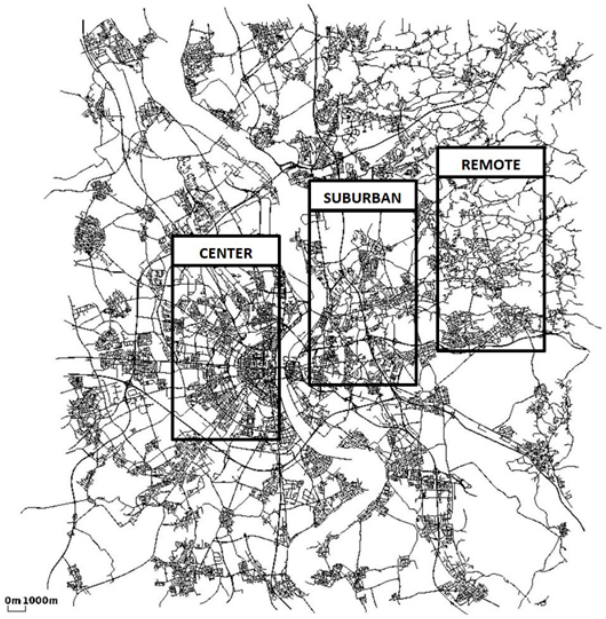


Figure 4: The three sub maps as shown in Cologne city map

The test results of this solution have shown the accuracy over the SUMO API.

B. Simulation scenarios

In our experiments, the principle is to evaluate and compare the algorithms efficiency under varying road networks size and traffic loads, while planning trips of various lengths. Furthermore, from practitioners point of view, if we consider the most widely used ITS, named SCATS, characterized by its 3-tier control architecture,

[1]TAPASCologne is an initiative by the Institute of Transportation Systems at the German Aerospace Center (ITS-DLR), aimed at reproducing, with the highest level of realism possible, car traffic in the greater urban area of the city of Cologne, in Germany." From http://kolntrace.project.citi-lab.fr/

[2]SUMO API calculates the travel time as the ratio of the road segment length and the average speed of all the vehicles running on this segment.

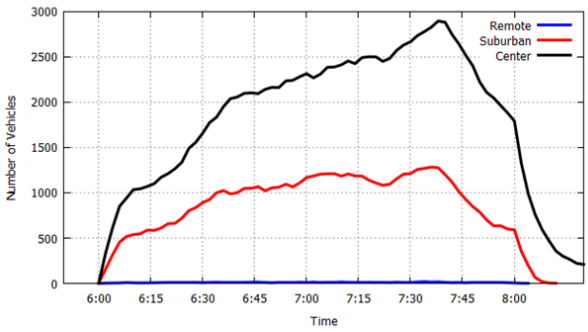


Figure 5: Traffic load in the three sub maps

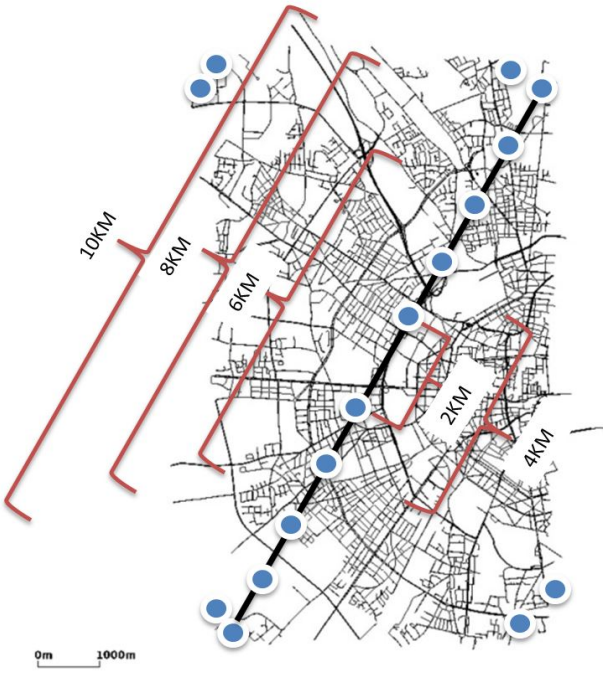


Figure 6: Illustrative example of Origin-Destination (OD) pairs selection

where in the middle tier there are several sub-mainframes in charge of the road monitoring equipment deployed in one specific region. Consequently, if the existing ITSs need to add a routing feature, as IBM Intelligent Operations for Transport for example, the comparative study we are conducting in this paper would be highly valuable, as road network characteristics and topology vary from one region to another.

As shown in Figure 4, we extract three sub-maps from three different areas in the original large Cologne road network, which are city center, suburban and remote area. Thus, we have 3 different simulation scenarios named center, suburban and remote, where each of them represents a different scalability level. Although these 3 sub-maps have the same size: 5.350(width)*9.350(length) = 50.0225 km^2, they can still ensure varying scalability levels in terms of the number of nodes and links in the graph representing the road map as well as the traffic load (i.e. the number of cars in a certain time period), as depicted in Table I and Figure 5 respectively.

Table I: Number of nodes and links in the three different road maps

	Number of nodes (Junctions)	**Number of links (Road segments)**
Center area	4025	8496
Suburban area	2597	5711
Remote area	1810	4170

C. Comparison methodology

As we have already set the test scenarios with different scalability levels, for routing algorithms searching a path for each Origin-Destination (OD) pair, we still want to find out how the performance of these algorithms evolves when the trip length gets longer. In fact, we wouldn't know the exact trip length till the car reaches its destination. In order to use trip length as another scalability parameter in our experiments, we apply the Euclidean Distance between the origin and the destination nodes to measure the trip length. Usually, the longest trip distance in an urban area is around 10km, so if a driver plans a trip longer than 10km, the hierarchical routing algorithm [10] is more suitable in this case. Consequently, in our experiments, the testing groups of OD pair are organized into 5 trip length scales, 2km, 4km, 6km, 8km, 10km as depicted in Figure 6. It is worth noting also that two OD pairs with similar Euclidean trip length may have quite different real trip distance due to the difference in the topology of the area between the origin and destination nodes. To mitigate the potential negative impact caused by this fact, we pick out four different OD pairs for one trip length group in one specific simulation and calculate the average of their results. Hence, we have 60 sets of testing results for each routing algorithm.

D. Evaluation metrics

The main metrics that we choose to measure the performance of the implemented routing algorithms are the number of selected nodes, the computation time, the required data storage space (i.e. the size of data to be loaded into memory during the algorithm execution), travel distance, travel time and travel time reliability. The number of selected nodes is a common metric, widely used in the literature, to show the quality of the theoretical design of a standard shortest path algorithm, the less is the better. The computation time is a different metric from the previous one because sometimes the algorithm can decrease the number of selected nodes, but the way to do so may bring too many time-costly computations such as power and evolution functions. Hence, the computation time is an indicator to assess the practical performance of the algorithm rather than the theoretical one. The third metric we consider in our evaluation is the used memory space. This metric is difficult to be monitored during the algorithm's execution, therefore we just measure the data storage space requirements as it is proportional to the memory cost. Some algorithms show the best performance in terms of computation time but this advantage may cost large memory space usage. Even though memory usage is not a big issue any more due to the recent advances in data storage technology, it is still one of the key indicators from engineering perspective, especially when deploying or optimizing the operations of the existing large scale ITSs.

The following two metrics are travel distance and travel time, which represent the real distance travelled by the vehicle and the time spent during the trip. These two metrics are very important to drivers as the cost of a route, in terms of fuel consumption, is highly dependent on them. The last metric is travel time reliability which can be defined as the degree of how likely encountering an abnormal delay when the vehicle travels on a specific road segment during a given time period. In our simulation, we calculate the travel time reliability based on Poluss study [21], as follows:

$$TT\_Reliability = \frac{StandardDeviation}{AVG\_TT} \quad (1)$$

Notice that the higher value of $TT\_Reliability$ (i.e. Travel Time Reliability) we get from the Equation 1, the lower travel time reliability we have. This equation represents a simple method to compute the reliability because we just need to know which algorithm can provide the most reliable route and which one performs worst with regard to this metric. In our simulation, we have collected 240 samples of average travel time for the period from 6:00am to 8:00am with sampling interval equals to 30 seconds for every link. Subsequently, we used these samples to calculate the standard deviation and AVG_TT (i.e. Average Travel Time), and then calculate the $TT\_Reliability$ for each link in the road network for the three sub-maps. Finally, we compute the $TT\_Reliability$ of each route as a sum of the $TT\_Reliability$ of all the links it consists of.

IV. Simulation results and analysis

In this section, we provide detailed discussion and analysis of the obtained simulation results according to the aforementioned evaluation metrics.

A. Number of selected nodes

The results shown in Figure 7 highlight the theoretical performance for the different algorithms. Usually, for all algorithms, the number of selected nodes [6] decreases gradually with the decrease of scalability level (i.e. the size of the road network that varies from center, suburban and remote areas) as well as the trip length. These results lead to some interesting conclusions. First, the dynamic and static versions of DA exhibit similar performance and are much less effective in this aspect compared to A*, which means that DA confirms its lack of advantage from design point of view. However, due to the ease of its implementation, as shown in the following test, DA is still useful under certain circumstances. Second, due to the advanced design of its lower bound, dynamic A* always performs the best and left the other three algorithms far behind even compared with static A*. The only exception is when the trip is planned in the center area with a length of 2KM, where both dynamic A* and static A* show the same theoretical performance. In this case, we recommend static A* for the sake of implementation simplicity. Third, we find that in the remote area scenario, the theoretical performance of static A* shows clear degradation when the trip length gets longer (i.e. $\geq$ 6KM), especially for 10KM trips.

B. Computation time

The computation time reflects the practical performance of an algorithm based on its execution time and is calculated after the map and lower bounds have been loaded into memory. As depicted in Figure 8, the computation time for all the algorithms is proportional to the scenario scalability level as well as the trip length. In remote area scenario, the performance of static A* decreases sharply when the trip length is equal to or greater than 6 km. These results are mainly in line with the theoretical performance results discussed above. Besides that, there are three observations worth noting. First, dynamic A* outperforms the other algorithms under almost all the tested scenarios. It performs even better than static A* as this latter needs to calculate the lower bound (including involution and evolution operations) during its execution while dynamic A* just loads the lower bound it needs into the memory. Second, dynamic DA show always the worst performance and is much less effective compared with the other three algorithms because it has to check the travel time whenever a new node is selected and it lacks an enhanced design as A*. Last but not least, static A* achieves the best practical performance when the trip length is less than 6km in center area, 4km and 2km in suburban area, and 2km in remote area.

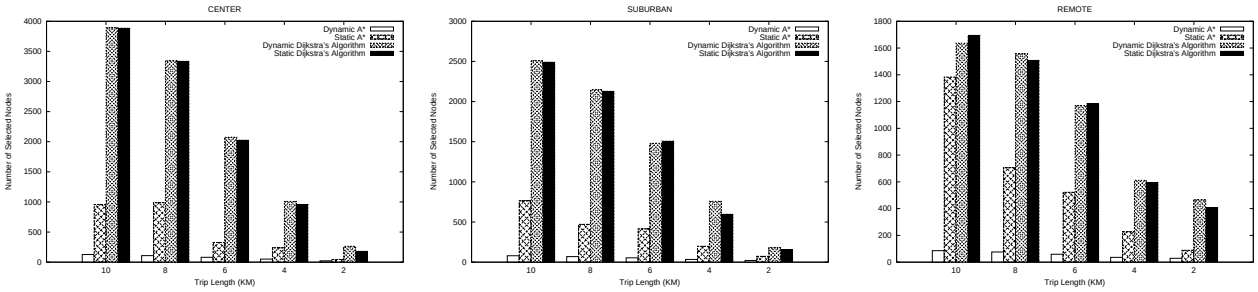
C. Travel time and travel distance

It is acknowledged that A* and DA always give the same best route solution if A* doesn't overestimate the minimum cost from source node to destination node. Therefore, here, we consider both algorithms as a whole and just compare their static and dynamic versions. As shown in the histogram of travel time in Figure 9, the results are clear for the trip lengths 10km, 8km and 6km, from which we can conclude that for the same trip length the static algorithms ensure a faster route in remote scenario compared to suburban and center areas. Notice that in center area scenario the calculated route is the slowest. On the contrary, for shorter trips length (i.e. 2 and 4 KM) the results are unclear for static algorithms, as in this case the quality of the route, in terms of travel time, would be highly dependent on the road topology between the OD pairs. Although the results for dynamic algorithms are more or less in a same pattern for the trip lengths greater than or equal to 4 km, the order is not as normal as we expected because they provide better routes in suburban scenario compared to the center area scenario. Moreover, the calculated route in the remote scenario is faster than that calculated in center scenario for trip lengths of 10km and 4km only, while very similar routes, in terms of travel time, are calculated for trip lengths of 8km and 6km. From these results we can conclude that the dynamic algorithms can provide more stable routes, in terms of travel time, compared to the static counterpart. Finally, we notice that for short trips of 2km and 4 km all the algorithms provide very similar quality of route. Hence, in this case we suggest using the simplest algorithm.

Looking at the graph of travel distance depicted in Figure 10, we see that the static algorithms can always give the shortest route compared with the dynamic ones. However, this advantage is limited to trips of the same lengths in one specific scenario. Consequently, if the travel distance is the only metric considered for vehicles routing then any of the four algorithms can satisfy the drivers requirements. The only exception for this metric is the case of trip length of 2 km where the travel distance planned in remote area is almost 3 times much longer than the other two scenarios. This is mainly due to the characteristics of the road network topology in the remote area. To overcome this issue, we suggest that the vehicle's navigation system might recommend alternative metrics when the computed travel distance exceeds some thresholds.

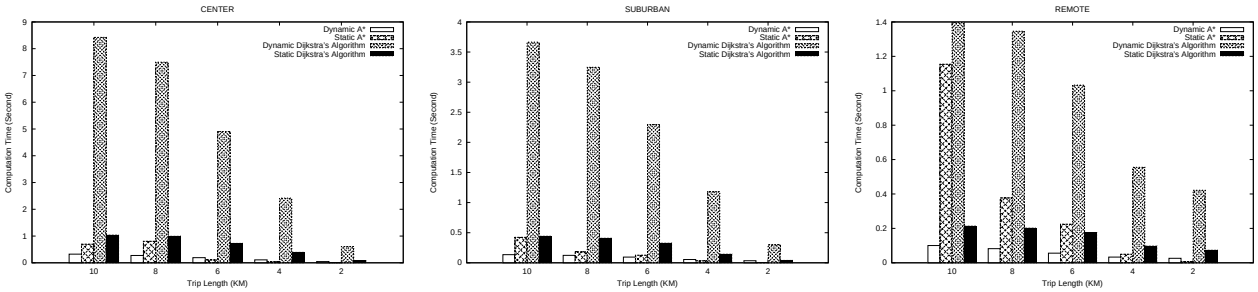
D. Travel Time Reliability

The results plotted in Figure 11 divulge, as expected, that the $TT\_Reliability$ differs significantly in the three scenarios. For the suburban scenario, the travel time reliability of the routes provided by both static and dynamic algorithms is higher (i.e. the lower value in the graph) that of the routes calculated in the center area. However, this supremacy decreases gradually when the trip length gets shorter. When the trip length drops to 2km the four algorithms show roughly the same performance. On the other hand, for the remote scenario, the travel time reliability of the routes calculated by the four algorithms is much higher (lower value, around 2500 times lower) than the previous case. This is due to the fact that during the period from 6:00am to 8:00am there is almost no change for the traffic flow in the remote area, as depicted in Figure 5. Last, for algorithm comparison, in center scenario, static algorithms perform slightly better than the dynamic ones, in suburban case they show roughly the same performance, while in remote area, dynamic algorithms are better. We remark from these results that one abnormal point exists in the remote scenario trip length of 4km. We assume that the reason behind this is the very low change of traffic flow in the remote scenario, so the result of travel time reliability would be very sensitive to the various topologies in the area between the different OD pairs. To conclude, we didn't find any obvious difference between the four algorithms, in terms of travel time reliability, therefore an improvement would require a new algorithm to be devised.



(a) Case of center area (b) Case of suburban area (c) Case of remote area

Figure 7: Number of selected nodes vs. trip length



(a) Case of center area (b) Case of suburban area (c) Case of remote area

Figure 8: Computation time vs. trip length

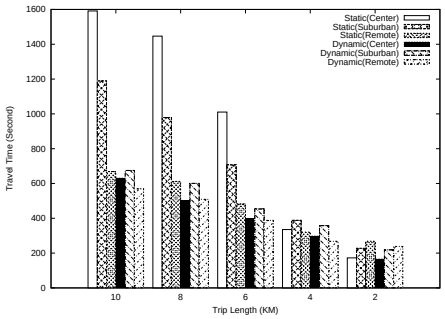


Figure 9: Impact of trip length and road network topology on the efficiency of vehicles routing algorithm in terms of travel time

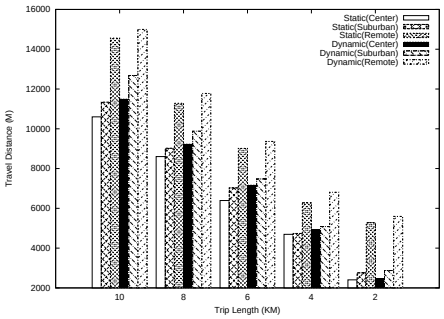


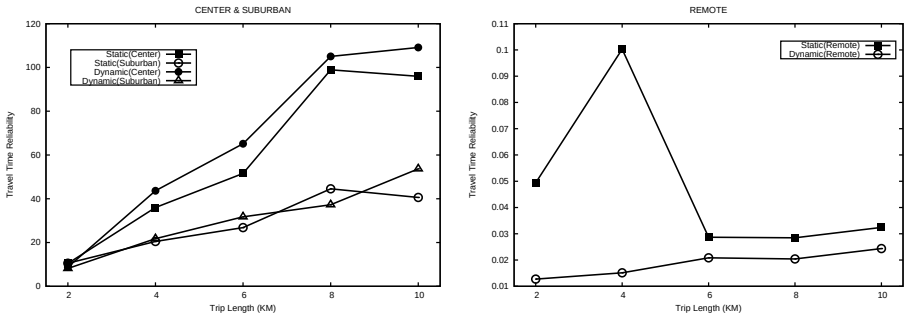
Figure 10: Impact of trip length and road network topology on the efficiency of vehicles routing algorithm in terms of travel distance

E. Data storage space usage

In the Table II, we present the memory space needed by each algorithm to perform the route calculation under different scalability levels. Basically, static algorithms need only to load the map data into the memory, and in our implementation this data consists of a static map "$Static\_Map$" data in SUMO format. In contrast, for the dynamic algorithms more data need to be loaded such as link status data "$Link\_Status$", node data "$Nodes$" and lower bound data "$Lower\_Bounds$". The link data shows the different travel times on different time intervals for every link, in addition to the transportation topology data which includes node data and basic link information. The link status in the dynamic context is thus a huge volume of data with its size is the number of time intervals times larger than the corresponding size in the static context. In our simulation, we set the travel time update frequency to 30 seconds, and the simulation duration to 2 hours, which means that the dynamic link status data is 240 times larger than the static links data. For dynamic A*, its advanced lower bounds needs to be pre-calculated by static all-to-all DA and the results should be stored for each scenario. Afterwards, these results will be loaded to the memory to enable the execution of A* algorithm. We notice from the Table II that dynamic A* needs 55.56 times more memory space than its static counterpart when the execution being performed in center area, and even for the remote area, it still needs 126,713,008 bytes, which is 31.47 times more that static A*. This is the only one obvious disadvantage of dynamic A*.

F. Implementation cost

Besides the five metrics that we have discussed above, the algorithm implementation cost is another important aspect that should be taken into account to ensure more informed decision about what



(a) Case of Center and suburban areas

(b) Case of Remote area

Figure 11: Travel Time Reliability vs. trip length

Table II: Data storage space usage by each algorithm (in Bytes)

	Center area			**Suburban area**			**Remote area**		
	Static	**DDA**	**DA***	**Static**	**DDA**	**DA***	**Static**	**DDA**	**DA***
$Static\_Map$	7,884,103	null	null	5,455,490	null	null	4,039,884	null	null
$Nodes$	null	217,250	217,250	null	144,669	144,669	null	101,628	101,628
$Link\_Status$	null	144,139,906	144,139,906	null	96,389,386	96,389,386	null	69,029,812	69,029,812
$Lower\_Bounds$	null	null	293,667,771	null	null	120,703,021	null	null	57,581,568
$Total$	7,884,103	144,357,156	438,024,927	5,455,490	96,534,055	217,237,076	4,039,884	69,131,440	126,713,008

algorithm to use in a centralized ITS. Since sometimes the algorithms are implemented at the hardware level which is highly dependent on the number and type of statements for the algorithm execution, a simple implementation can not only reduce the computation time but also decrease the energy consumption. Since A* has similar implementation to DA with one more estimation function, we can define the order of the implementation cost of the four algorithms studied in this work as follows:

$$DynamicA^* > DynamicDA > StaticA^* > StaticDA$$

Finally, our suggestions on the best algorithm to apply in different scenarios are presented in Table III. These suggestions are based on the centralized ITS architecture, in which the ITS server receives large number of drivers requests of fastest and shortest routes.

V. Conclusion

Throughout this paper we have provided a thorough performance evaluation of four vehicles' routing algorithms followed by deep analysis and comparison of the obtained results. This evaluation work for both static and dynamic routing algorithms is carried out based on real transportation network and highly realistic traffic load. Such valuable performance assessment under different scalability levels and trip lengths has never been done in the literature, for the best of our knowledge. Moreover, we have discussed the implementation cost of these algorithms and suggested the most suitable algorithm to apply in several scenarios. Dynamic DA has never been suggested for any scenario of practical use due to its exaggerated computation time. If the driver needs the shortest route, static DA is recommended for centralized ITS use in remote area due to its low complexity and good performance in terms of computation time. In the center and suburban scenarios, static A* is a good choice for long trips (i.e. $\geq$ 6km) whereas static DA is a better alternative for short trips (i.e. $\leq$ 4km). For fastest route queries, dynamic A* would be highly recommended due to its low computation time and high quality of the calculated route, especially for long trips. For shorter trips, static A* is preferred as it can also provide routes with good travel time and its memory usage cost is low. In the future, we plan to vary the departure time for the same simulation background in the center area scenario to explore the impact of traffic load variation on the aforementioned evaluation metrics. Besides, by using tools provided by IBM Industry Solutions and Research Labs, we can generate the more complete simulated data for a whole week (24 hours traffic data) to investigate more features of the four studied algorithms.

VI. Acknowledgment

This work was supported, in part, by Science Foundation Ireland grant 10/CE/I1855 to Lero - the Irish Software Engineering Research Centre (www.lero.ie).

References

[1] Schrank, David, Bill Eisele, and Tim Lomax. "2012 Urban Mobility Report." Texas Transportation Institute, Texas A & M University, 2012.

[2] Bureau of Public Roads (1964). Traffic Assignment Manual. U.S. Dept. of Commerce, Urban Planning Division, Washington D.C.

[3] K. Chen and J.C. Miles, ITS Handbook 2000 Recommendations from the World Road Association (PIARC), Artech House, 1999.

[4] Freeway Incident Management Handbook, Federal Highway Ad- ministration, 2000. [Online]. Available: http://ntl.bts.gov/lib/jpodocs/ rept-mis/7243.pdf

[5] Dijkstra, Edsger W. "A note on two problems in connexion with graphs." Numerische mathematik 1.1 (1959): 269-271.

[6] Chabini, Ismail, and Shan Lan. "Adaptations of the A* algorithm for the computation of fastest paths in deterministic discrete-time dynamic networks." Intelligent Transportation Systems, IEEE Transactions on 3.1 (2002): 60-74.

[7] Lowrie, P. R. "The Sydney coordinated adaptive traffic system-principles, methodology, algorithms." International Conference on Road Traffic Signalling, 1982, London, United Kingdom. No. 207. 1982.

[8] Hunt, P. B., et al. SCOOT-a traffic responsive method of coordinating signals. No. LR 1014 Monograph. 1981.

[9] Zhan, F. Benjamin, and Charles E. Noon. "Shortest path algorithms: an evaluation using real road networks." Transportation Science 32.1 (1998): 65-73.

[10] Fu, Liping, D. Sun, and Laurence R. Rilett. "Heuristic shortest path algorithms for transportation applications: state of the art." Computers & Operations Research 33.11 (2006): 3324-3343.

[11] Hart, Peter E., Nils J. Nilsson, and Bertram Raphael. "A formal basis for the heuristic determination of minimum cost paths." Systems Science and Cybernetics, IEEE Transactions on 4.2 (1968): 100-107.

Table III: Summary of our suggestion on the most efficient vehicles routing algorithm in different road scenarios

	Trip length				
Fastest / Shortest	**10 km**	**8 km**	**6 km**	**4 km**	**2 km**
Center area	Dynamic A* / Static A*	Dynamic A*/ Static A*	Dynamic A*/ Static A*	Static A*/ Static DA	Static A*/ Static DA
Suburban area	Dynamic A*/ Static A*	Dynamic A*/ Static A*	Dynamic A*/ Static A*	Static A*/ Static DA	Static A*/ Static DA
Remote area	Dynamic A*/ Static DA	Dynamic A*/ Static DA	Static A*/ Static DA	Static A*/ Static DA	Static A*/ Change the route criterion

[12] Huang, Bo, Q. Wu, and F. B. Zhan. "A shortest path algorithm with novel heuristics for dynamic transportation networks." International Journal of Geographical Information Science 21.6 (2007): 625-644.

[13] Koenig, Sven, Maxim Likhachev, and David Furcy. "Lifelong planning A*."Artificial Intelligence 155.1 (2004): 93-146.

[14] Bell, Michael GH, et al. "Time-dependent Hyperstar algorithm for robust vehicle navigation." Transportation Research Part A: Policy and Practice (2012).

[15] Bell, Michael GH. "Hyperstar: A multi-path Astar algorithm for risk averse vehicle navigation." Transportation Research Part B: Methodological 43.1 (2009): 97-107.

[16] Spiess, Heinz, and Michael Florian. "Optimal strategies: A new assignment model for transit networks." Transportation Research Part B: Methodological 23.2 (1989): 83-102.

[17] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, SUMO - Simulation of Urban MObility - an Overview, in SIMUL 2011, The Third International Conference on Advances in System Simulation,(Barcelona, Spain), 2011.

[18] Wegener, Axel, et al. "TraCI: an interface for coupling road traffic and network simulators." Proceedings of the 11th communications and networking simulation symposium. ACM, 2008.

[19] Uppoor, Sandesh, and Fiore Marco. A large-scale vehicular mobility dataset of the Cologne urban area.14mes Rencontres Francophones sur les Aspects Algorithmiques des Tlcommunications (AlgoTel) (2012): 1-4.

[20] V. T. Ngoc Nha, S. Djahel and J. Murphy. "A Comparative Study of Vehicles' Routing Algorithms for Route Planning in Smart Cities". *VTM 2012, Satellite Workshop of IFIP Wireless Days 2012*, Dublin, Ireland, November 20, 2012.

[21] A. Polus, "A study of travel time and reliability on arterial routes", *Transportation*, Vol. 8, No. 2, pp.141-151, 1979.