Anytime Route Planning with Constrained Devices

Marcus de L. Braga[a,c,*], Alyson de J. dos Santos[a,c], Aloysio C. P. Pedroza[a], Luís Henrique M. K. Costa[a], Marcelo Dias de Amorim[b], Yacine Ghamri-Doudane[c]

[a] *Universidade Federal do Rio de Janeiro – PEE/COPPE/GTA – Rio de Janeiro, Brazil*
[b] *LIP6/CNRS – UPMC Sorbonne Universités – Paris, France*
[c] *Université de La Rochelle – L3i – La Rochelle, France*

Abstract

Urban mobility became a major challenge around the world, with frequent congestion and ever growing travel time. Albeit recent advances in the area of Intelligent Transportation Systems (ITS), it is still difficult to predict and manage the road infrastructure due to dynamics and instability of the traffic. One key issue is how, given some traffic monitoring information, a vehicle decides to dynamically change its route. In this paper, we analyze algorithms of the anytime class to make the route planning considering GPS traces of buses in Rio de Janeiro, as a measurement of traffic flows. Anytime algorithms inform, in a timely fashion, a sub-optimal response and progressively improves it as time goes by. We evaluate time and memory consumption, route length, arrival time, average velocity, distance traveled, and pathways on an experimental platform composed of Raspberry Pi nodes. For different time windows, the results show that ARA* allows finding alternative routes that, if used, help reduce traffic congestion.

Keywords: Route planning, ARA*, vehicular networks, anytime algorithms.

1. Introduction

Urban mobility is a major challenge for managers of metropolises worldwide, significantly impaired by the constant traffic jams that cause economic, social, and environmental impacts. From the physical point of view, road traffic is an unbalanced system of particles (vehicles) that interact and influence each other, leading to instability caused by floating state valuations of these particles when they exceed a critical value [1]. Mechanical or electrical faults, works on roads,

*Corresponding author.
Email addresses: marcus@gta.ufrj.br (Marcus de L. Braga), alyson@gta.ufrj.br (Alyson de J. dos Santos), aloysio@gta.ufrj.br (Aloysio C. P. Pedroza), luish@gta.ufrj.br (Luís Henrique M. K. Costa), marcelo.amorim@lip6.fr (Marcelo Dias de Amorim), yacine.ghamri@univ-lr.fr (Yacine Ghamri-Doudane)

and abrupt lane changes are some factors that contribute to the imbalance of the road system. Driver and pedestrians, with their unpredictable behaviors, are susceptible to fatigue, distraction, and substances that cause the decrease of attention and reflexes. These variables are inherent to the system and cannot be ignored [2, 3, 4].

The extension of the road network and traffic interventions are widely used for increasing the efficiency of the flow of vehicles. However, such measures are time-limited, being efficient only until the next growth in demand [5]. The Traffic info and recommended itinerary, vehicle signage (electronic board) and, limited access warning are ITS applications that assist traffic management [6]. These initiatives have multiple goals, such as minimizing delays perceived by vehicle drivers, and improving the control and management of the road network reducing environmental impacts (e.g., air pollution).

The use of alternative routes is a solution adopted by many drivers to escape traffic jams. However, this solution has no guarantee of success, if there is no information about current conditions of the chosen route. It is clear that despite the technology employed in traffic monitoring systems, there are no self-management techniques capable of acting in unexpected situations and providing proactive traffic management [7].

To help drivers find appropriate alternative routes, we propose to rely on advanced route planning techniques. Route planning consists in finding the shortest path based on the states of the edges with associated costs. The path is optimal if the sum of weights in the edges is minimal in all possible paths from the initial vertex to the goal vertex [8]. Path planning is also important in the mobile robotics area (for navigation and arm movement) and is a support system to decision making in autonomous vehicles, where the localization methods are based on GPS or DGPS[1] [9, 10]. This work investigates the use of the search algorithm Anytime Repairing A* (ARA*) for route planning in a real road network, using a Raspberry Pi prototype as a proof of concept of the execution in resource-constrained devices.

We consider the definition of a resource-constrained device as devices with limited CPU, memory, and energy capability. Those devices range from embedded sensing systems such as the Arduino to expensive smartphones, where the limitation is not the CPU, but energy starvation still is. Therefore, the motivation for investigating route planning in resource-constrained devices is twofold. First, for almost every mobile device energy is an issue and saving CPU clocks preserves energy. Second, one may argue that offloading the program execution to the cloud is an alternative solution to reducing the program execution complexity. The problem is, even if computation offloading is an elegant solution, it is not always feasible. First, certain tasks can be offloaded to the cloud, others not, such as tasks which depend on reading variables that are local to the device, such as a sensor or the GPS. Those are not available in the remote system in the cloud. Second, the device can find itself in an area without Internet access.

[1]The Differential Global Positioning System has an accuracy of 10 cm.

In that case offloading to the cloud is not possible.

We have implemented anytime ARA* algorithm and a proposed and implemented a code-optimized version of ARA*, therefore decreasing the execution time. We use the implementation of classical search algorithms, A* and Dijkstra, in order to perform a comparative study among them.

In this paper, we make a comparative study of algorithms to choose the short path, given the points of source and destination selected randomly. We simulate real traffic of vehicles, using real GPS data of public transportation buses and the Rio de Janeiro city map. The Raspberry Pi platform was used in the tests and the short path is found in terms of distance and travel time. On the other hand, runtime and the memory consumed were measured, as well the quantities of nodes composing the paths found, the distance traveled, the average velocity, the arrival time and pathway's frequency used by algorithms. In summary, our main contributions in this work are:

- We need, besides short paths, also efficient alternative routes. We analyze alternative routes generated from robotic algorithm (ARA*), using real road map and traffic flow from a Brazilian metropolis, Rio de Janeiro.
- We propose the use of path planning algorithms in a constrained device.
- We propose an improvement in ARA* implementation, where the gain computation on average was 3% in the results.
- We describe and evaluate the algorithms, in relation to distance and travel time, using three performance metrics in the three time windows: a) from the algorithm; b) from the routes; and c) from the graph.

Nowadays, the Rio de Janeiro city has third worst traffic of 146 cities in the world, with 51% in the congestion level [11]. Rio de Janeiro city has recently hosted major events such as Soccer World Cup (2014), Rock in Rio festival (2015), as well as the Olympics (2016), suffering with many work sites and traffic jams. Thus, it was the chosen the scenario for testing the route planning algorithms. Raspberry Pi was the computing platform chosen for tests, due to its portability and limited CPU resources. We used two metrics to find the short path: distance and travel time. The execution time and memory consumption, the number in the path, the distance traveled, the average velocity, the travel time are evaluated in the results.

This paper is organized as follows. Section 2 presents related work. Section 3 presents the ARA* algorithm and proposes an optimization called ITS-ARA*. Section 4 describes the experiments, results and analyses with a real implementation on a resource-constrained device. Section 5 concludes the paper and discusses future work directions.

2. Finding paths: Shortest and near-shortest approaches

A major problem in graph theory is the shortest-path problem, i.e., how to find the shortest route, if any, between two nodes of a graph. Formally

speaking, a graph is a pair $G = (V, E)$, where V is a non-empty set of nodes and E a set of edges. Although the shortest-path problem is generally studied in static structures, many problems involving graphs are dynamic by nature. The vehicular scenarios that we consider in this paper are typical examples of dynamic graphs; thus, algorithms such as Dijkstra, Bellman-Ford, and Breadth-first, which were conceived for static topologies, do not work well.

In Faez and Khanjary [6] Dijkstra's algorithm is used to find the shortest path based on traffic conditions collected by Wireless Sensor Networks (WSN) and ITS (optical, inductive, magnetic and video cameras). The optimized routes are simply sent to vehicles by messages in the basic mode. In advanced mode, vehicles are with transponders to provide direct and interactive communication.

The *MobEyes* system proposed by Lee *et al.* [12] uses a bio-inspired algorithm for determining the best path, using information from sensors which capture, classify and periodically generate summary information extracted from the context of vehicular traffic. These summaries (car position or route taken at a certain time) are disseminated in the network using mobile agents and opportunistic relaying. The algorithm is bio-inspired with three behaviors: the bacterium *E.coli*, which shows modes of movement that vary according to the concentration level of nutrients; the ants (other social insects), which use pheromone to signal to fellows the nest with potential conflicts and the *Lévy flights*, which are inspired by the flight behavior of some groups of birds such as the albatross when seeking food.

Li *et al.* [13] propose to use the A* search algorithm with real time adaptation based on the traffic conditions of the period of the day. The choice is made based on the lowest cost for each different time of the day. The proposal uses two fuzzy rules, one for automatic adaptation to changes in the weights of traffic based on vehicle density, and the other makes use of a function to adjust the weights according to criteria set by the driver.

Li *et al.* [14], propose the ARA*+ algorithm (for robotics), a variation of ARA* where the strategy is the re-expansion of the states (visited nodes) after the coefficient ϵ is decreased. The results of the experiences show that the number of expanded states and the search time to get the optimal path is smaller in ARA*+ than ARA*. The first search of ARA*+ is the same of ARA*, found quickly a suboptimal path with ϵ_0 (initial value) and without expansion of the states. If there is time to continue searching then ϵ is decreased and a new result is found based on the previous results. The re-expansion of the results depends on the value of ϵ. In the experiments, for the value 1.2 and 1.0, ARA*+ has less expanded states than ARA* and consequently has lowest search time.

For the selection of the optimal route, Mustafa *et al.* incorporate points of interest (POIs) that users intend to traverse [15]. Route planning considers the division of the geographic area into clusters, as well as the driver's comfort level and the vehicles fuel consumption. Although a real road network has been used in the experiment, the methodology is hard to generalize, considering that driver's comfort is difficult to measure.

The proposals of Lee *et al.* [12] and Li *et al.* [13] use time-based information monitoring the road network. The first proposal uses digests of the behavior

of vehicles (timestamp with position or route taken) at certain times and the second proposal uses information of a given period of one day. However, both proposals do not reflect the current conditions of the pathways because the mode of dissemination of information can cause traffic jams, due to many vehicles receiving the same information and concentrating on the same routes.

The proposal of Faez and Khanjary [6] is quite sophisticated requiring large investment and presenting challenges regarding interoperability of equipment, due to the existence of many manufacturers, each of which with its own communication protocol. Li *et al.* [14] proposed the ARA*+ algorithm, a variation of the ARA* for path planning of robots. The strategy for accelerating the optimal path search is a re-expansion of the states for a certain value of ϵ, a parameter of ARA* algorithm explained in the next section. In this work our proposal for accelerating the algorithm is a optimization in the code, where in the basis case had a computational gain.

3. Anytime Algorithms: From ARA* to ITS-ARA*

Route planning is a key technique in proactive management. We say that a route planning algorithm is complete if it finds a path (if any) in finite time. Otherwise, the algorithm returns the non-existence of a path also in finite time. A planning algorithm is optimal if it finds an optimal path [16, 17].

Although many navigation systems provide the shortest path or the shortest travel time, this is not a guarantee of success in getting to the destination faster, because these systems are based on historical averages of traffic, and could transfer traffic jams from one route to another because the information is shared. Anytime algorithms for route planning quickly find sub-optimal paths and gradually improves the result with time available [18]. The choice of the ARA* algorithm in this paper is due to two main features. Firstly, it adapts well to dynamic traffic by presenting first a sub-optimal route and improving over time. Secondly, the production of sub-optimal routes favors traffic balancing between different routes.

We compare ARA* with Dijkstra's shortest-path algorithm, and A*, the original version of ARA* without anytime route repair. The pseudo code and the descriptions of ARA* algorithm are presented next.

3.1. Anytime Repairing: A and ARA**

ARA* (Algorithm 1) was proposed by Likhachev *et al.* as a faster alternative to the A* algorithm [19]. A* finds sub-optimal solutions based on a heuristic function. If this heuristic function is multiplied by a factor ϵ, the sub-optimality of the solutions found can be tuned. This is called an inflated heuristic. If one runs A* multiple times, each with different values of ϵ, the algorithm provides solutions *at any time*, with different sub-optimality levels. By using different ϵ and running enough time, the algorithm tends to the optimal solution. The basic idea of ARA* is the same, running A* several times, with different values

of ϵ, but reusing previously found results. That way, ARA* tends to find sub-optimal results faster. The same way as A*, the refinement of searches of ARA* tends to the optimal search with increasing execution time.

ARA* relies on the function $f(x) = g(x)+\epsilon * h(x)$, where $g(x)$ is the distance from the vertex x to the source, $h(x)$ is an estimate (based on a heuristic) of the distance to the destination vertex, and ϵ is the heuristic coefficient. The higher the value of ϵ, the less nodes are visited and sub-optimal results will be calculated (much faster). On the other hand, if $\epsilon = 1$, the operation is equivalent to the standard A* with a standard heuristic (not inflated), which is guaranteed to be optimal [13, 19]. The optimality in ARA* is achieved through successive refinements of the results and by decreasing the coefficient ϵ until it reaches 1. At each iteration the values of $f(x)$, $g(x)$, and $h(x)$ are updated for each vertex previously found. For the experiments we used the Euclidian distance heuristic function.

Algorithm 1 ARA*

```
 1: function fvalues(x))
 2:     return g(x) + ε * h(x)
 3: end function
 4:
 5: procedure improvePath())
 6:     while (key(x_goal) > min_{x∈OPEN}(key(x)) do
 7:         remove x with the smallest fvalues(x)from OPEN;
 8:         CLOSED = CLOSED ∪ x;
 9:         for (each successor x' of x) do
10:             if (x' was never visited by ARA* before) then
11:                 v(x') = g(x') = ∞
12:             end if
13:             if (g(x') > g(x) + c(x, x')) then
14:                 g(x') = g(x) + c(x, x')
15:                 if (x' ∉ CLOSED) then
16:                     insert/update x' in OPEN with fvalues(x');
17:                 else
18:                     insert x' into INCONS;
19:                 end if
20:             end if
21:         end for
22:     end while
23: end procedure
24:
25: procedure Main())
26:     g(x_goal = v(x_goal) = ∞; v(x_start) = ∞;
27:     g(x_start = 0; OPEN = CLOSED = INCONS = ∅;
28:     insert x_start into OPEN with key(x_start);
29:     improvePath();
30:     publish current ε − suboptimal solution;
31:     while ε > 1 do
32:         decrease ε;
33:         Move states from INCONS into OPEN
34:         Update the priorities for all x ∈ OPEN according to fvalues(x);
35:         CLOSED = ∅;
36:         improvePath();
37:         publish current ε − suboptimal solution;
38:     end while
39: end procedure
```

ARA* (Algorithm 1) works with three lists: *OPEN*, *CLOSED* and *INCONS* (line 27). The *OPEN* list contains the nodes that have been used in the heuristic

function $h(x)$, but which have not been examined and expanded yet. This list is characterized by the priority of its elements: the smaller the value of the heuristic function, the more promising the vertex. The *CLOSED* list contains the nodes already examined. The *INCONS* (inconsistent) list is responsible for preventing nodes that have function $f(x)$ reduced and are not present in the *OPEN* list. This mechanism avoids that a node is visited more than once. A temporary list is maintained and that content is added to the *OPEN* list, making the nodes already visited have their $f(x)$ functions reduced and re-evaluated. It means that it is not necessary to recalculate the path again, but only to update the weights, so that a "new route" is determined. The ARA* algorithm is basically composed of three procedures: *fvalue*, which is in charge of calculating $f(x) = g(x) + \epsilon * h(x)$, *improvePath*, which performs the refinement of searches, and the *Main* procedure that manages the search refinements by repetitions of *improvePath* calls with decrease of the heuristic coefficient ϵ.

In essence, the ARA* algorithm works like executing A* multiple times, making use of the heuristic coefficient until $\epsilon = 1$, to ensure that the solution is optimal [16]. ARA*, differently from A*, does not guarantee to visit each vertex only once during the search, due to the super estimation of the heuristic function caused by the coefficient ϵ. The *improvePath* procedure (lines 5-23 of Algorithm 1) is responsible for the reuse of the results and for the analyzing nodes that are part of the *INCONS* list. This list consists of nodes that have been visited (i.e., belong to the *CLOSED* list), but that had their heuristic function reduced, implicating in a new evaluation to verify if the node is part of the solution. This operation is performed in the *Main* procedure when the nodes of the *INCONS* list are moved to the *OPEN* list. The function *fvalue* is in charge of computing the heuristic function $f(x)$ for each vertex.

The *Main* (lines 25-39 of Algorithm 1) procedure defines the values of the ϵ for calculating the sub-optimal solution and initializes. The refinement of the solution is realized by a loop, where the operations of heuristic update in the nodes are realized by each decrease of the heuristic coefficient ϵ until it reaches 1 (optimal solution).

3.2. ITS-ARA: Faster ARA**

The basic idea of our modification of ARA* is to reorganize the calls to the following procedures: *improvePath* (lines 29 and 36 in Algorithm 1) and *publish current* ϵ *− suboptimal solution* (lines 30 and 37 in Algorithm 1). Our modification, that we call ITS-ARA* is detailed in Algorithm 2. It preserves the original ARA*'s operation and properties, but makes it faster due to computational economy obtained by avoiding to call lines 10, 11, and 12 when $\epsilon = 1$. We will show in our experiments that such a simple modification leads to satisfactory performance.

4. Experiments

For the experiments we used a real map of the Rio de Janeiro city, the area of Governador Island where the Antonio Carlos Jobim International Airport is lo-

Algorithm 2 ITS-ARA*

```
1: procedure MAIN())
2:     g(x_goal = v(x_goal) = ∞; v(x_start) = ∞;
3:     g(x_start = 0; OPEN = CLOSED = INCONS = ∅;
4:     insert x_start into OPEN with fvalues(x_start);
5:     while ϵ >= 1 do
6:         improvePath();
7:         publish current ϵ − suboptimal solution;
8:         decrease ϵ;
9:         if (ϵ > 1) then
10:            Move states from INCONS into OPEN
11:            Update the priorities for all x ∈ OPEN according to fvalues(x);
12:            CLOSED = ∅;
13:        end if
14:    end while
15: end procedure
```

cated, which has a population of 212,574 inhabitants (3.37 percent of the Rio de Janeiro city population), and an area of 40.81 km^2. Figure 1 shows the methodology used in this work: a) Road Network; b) Traffic Flow; c) GTAPlanning; and d) Data Analysis.

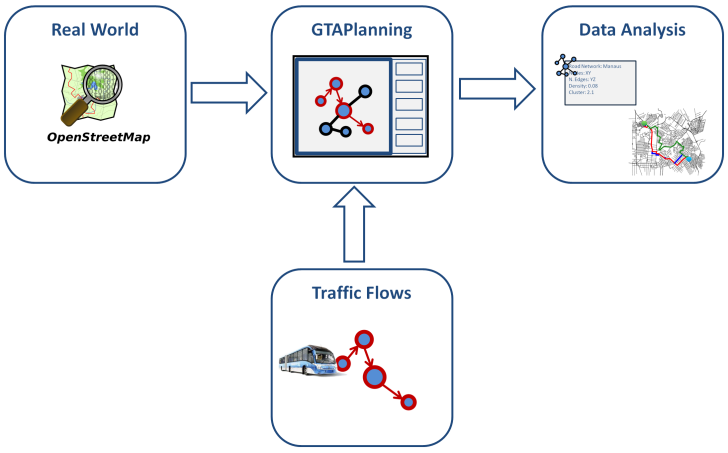
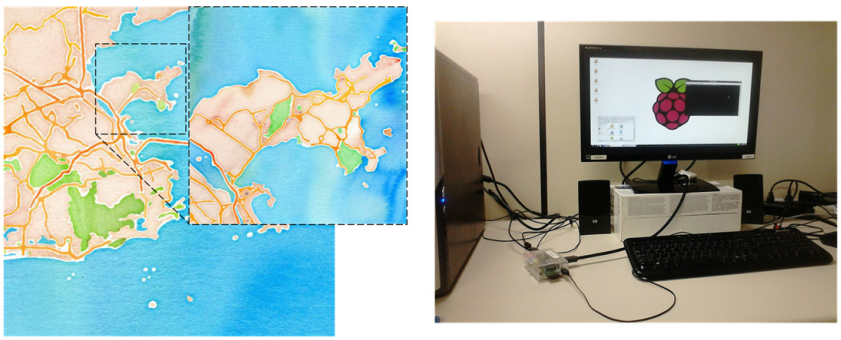


Figure 1: Methodology of the experiments.

Road Network. The map of Rio was downloaded from the OpenStreetMap [20, 21] site. The osm files are expressed in the form of XML formatted, providing basically data primitives such as nodes (points in space), paths (linear features and area boundaries) and relations (explaining how elements are connected). The road network of the Governador Island (Figure 2(a)) contains 7,146 nodes and 7,968 edges.



(a) Governador Island map. (b) Experiments on the Raspberry Pi B+.

Figure 2: Map and Raspberry Pi platform used in the experiments.

Traffic Flow. The vehicular trace we have used for Rio de Janeiro consists of GPS data collected from 6,775 buses, over a period of 31 days from October 1st, 2014 to November 1st, 2014. In this work we use just a one day trace with 972,755 GPS data with ten time windows from 6 am to 8 pm. In this paper, only the three most significant time windows (rush time) are represented: 1) from 7 am to 8 am; 2) from 11 am to 12 am; and 3) from 16 pm to 17 pm. Each entry in the data trace file has the format: timestamp, bus identification, bus line, latitude, longitude and velocity. Times are expressed in hours, minutes and seconds; latitude and longitude in degree (according to position records in GPS), and velocity in kilometers/hours.

GTAPlanning. Tool for route planning and road network analysis. The software is able to load maps in *DGS, OSM* formats and to inform the graph properties, as well as to provide the route planning based in the maps loaded. The DGS (Dynamic GraphStream) format refers to the GraphStream library used in the tool, and OSM format is defined by OpenStreetMap. The software was developed in Java and uses a *GraphStream* dynamic graph library. The route planning module is part of the Route Recommendation System, which is based on the collection traffic of as proposed by Ribeiro Júnior *et al.* [22], using IEEE 802.11 networks.

Data Analysis. In order to demonstrate the computational feasibility of the investigated algorithms, we measured the time of execution, memory consumed, number of nodes (that are part of the shortest path found), average velocity, traveled distance and arrival time. We calculated the shortest path in relation to distance and to travel time. The experiment consisted of randomly choosing 200 pairs (source and destination) for each measure, for a total of 400 results per algorithm. For the informed search algorithms (A*, ARA* and ITS-ARA*), we use Euclidean distance as heuristic function and the heuristic coefficient $\varepsilon = 5.0$ for ARA* and the ITS-ARA*. Raspberry Pi Model B+ [23] (Figure 2(b)) was used in the experiments. It is equipped, with an ARM processor at 700 MHz,

512 MB RAM and an SD card 4 GB, Raspbian operating system and Java Embedded 8.

4.1. Distance metric results

The algorithms are evaluated using different performance metrics and three time windows: a) from the algorithm, execution time and memory consumed are registered; b) from the routes, the traveled distance, the average velocity and arrival time are estimated based on the information extracted of the real data; and c) from the graph, the number of nodes on average in the shortest paths found, as well the frequency of the pathways used for each of the algorithms. From the last measure we can infer the different behavior of the algorithms studied. The results are presented with 95% confidence intervals. In this section the shortest path was calculated using the distance in meters as weight of the edges. This measure was extracted from OpenStreetMap, that provides latitude and longitude data. We use the Haversine formula to calculate great-circle distance between the two points on the surface of a sphere from the coordinates informed.

Figure 3: Governador Island Plot - Distance metric.

Figure 3 shows examples of the route planning generated by algorithms examined. The blue pin map represents the origin point and green pin map the destination point. In this example the result of the Dijkstra and A* was the same, however the ITS-ARA* and ARA* had a different behavior. Then we represent the Dijkstra/A* with blue path and, consequently, the ITS-ARA*/ARA* with green path. In Figure 4, we take the same origin and destination but use the algorithms with travel time as metric. We can notice that the difference between the routes in relation to the distance weight was a little bit smaller with

distance than travel time weight (Figure 4). In the following we present more data supporting these observations.

Figure 4: Governador Island Plot - Travel time weight.

Table 1: Distance weight results – a) Execution Time – execution time in average, given in milliseconds; b) Memory Consumed – memory consumed in average expressed in MB; and c) Nodes – average of the number of nodes in the path found.

Time Window	**Distance**			
	DJK	**A***	**ARA***	**ITS-ARA***
	Execution Time (ms)			
07–08 am	366.85	125.04	22.63	21.70
11–12 am	353.62	118.98	23.05	22.71
04–05 pm	353.59	108.43	23.71	22.88
	Consumed Memory (MB)			
07–08 am	159.68	159.77	159.34	159.64
11–12 am	157.39	156.66	156.11	156.22
04–05 pm	157.06	155.90	156.46	156.77
	Average Number of Nodes			
07–08 am	84.60	84.69	102.00	102.00
11–12 am	83.24	83.60	101.23	101.23
04–05 pm	82.68	82.71	100.87	100.87

Table 1 shows the results of the algorithms for each time window. We can notice that the informed search (A*, ARA* and ITS-ARA*) algorithms have been faster than Dijkstra, where the ITS-ARA* was the fastest in all time windows.

The ITS-ARA* gain was in average 93.73% in relation to Dijkstra, 80.82% in relation to A* and 3.02% in relation to ARA*. For the memory consumed, the four algorithms had the same performance, considering the confidence interval of 95%. In relation to the number of nodes, the results shows that Dijkstra and A* were more efficient, in average by around 17%.

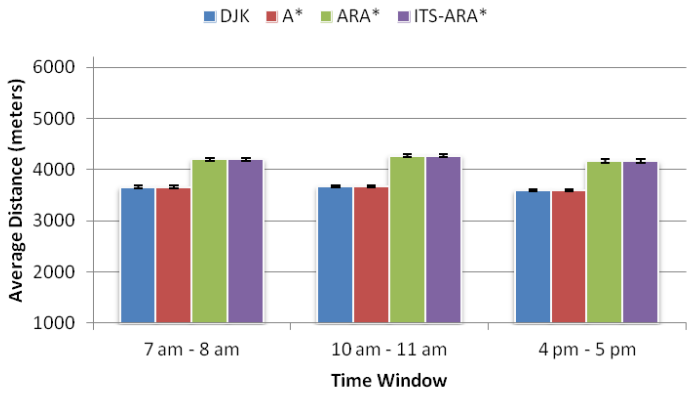


Figure 5: Average Distance (meters) – Distance weight.

Figure 5 shows the average traveled distance, the velocity and the arrival time using distance costs. The results for the average traveled distance confirm the best performance of Dijkstra and A* algorithms: 12.87% (541.09 meters) on the 07 am to 8 am time window; 14.06% (600.83 meters) on the 11 am to 12 am window; 13.71% (571.35 meters) on the 4 pm to 5 pm window.

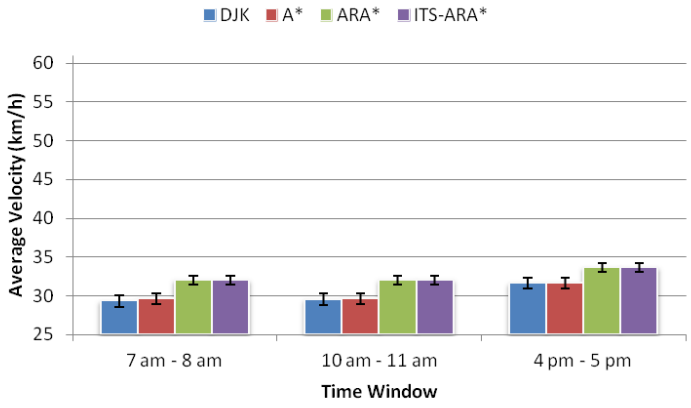


Figure 6: Average Velocity (km/h) – Distance weight.

In the velocity performance metric, the ITS-ARA* and ARA*, were superior,

on average, over to the others (Figure 6). The differences over A* and Dijkstra were: 8.39% (2.68 km/h) on the 7 am to 8 am window; 7.82% (2.51 km/h) on the 11 am to 12 am window; 6.09% (2.05 km/h) on the 04 pm to 5 pm window.

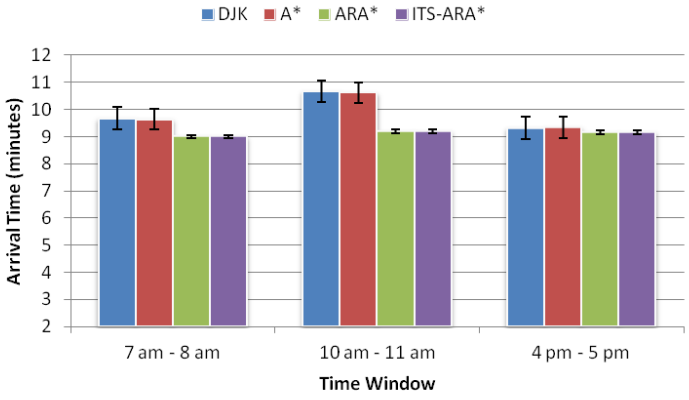


Figure 7: Average Arrival Time (minutes) – Distance weight.

The results in terms of arrival time (Figure 7) show that, on average, the ITS-ARA*/ARA* found better results only in the time window of 10 am to 11 am with 13.8% advantage or, in other words, 1.47 minutes. We can observe that the other results are in the confidence interval. The next results present the analyses developed about the pathway behavior, where the difference between the algorithms in relation to the pathway chosen was show.

Pathway numbers are identifiers for each edge in the graph. The frequencies were calculated, accounting the apparition total number for each algorithm in the time window. However, as expected, the Dijkstra and A* had the same performance, as well ITS-ARA* and ARA*. These results indicate which are the pathway (street or avenue) that are more used and its schedule. This information is useful and helps prevent the traffic jam, since we can observe if alternative routes are generated. We can note the difference between Dijkstra/A* and ITS-ARA*/ARA* in the pathway frequencies. Rarely they had the same result in the three window times, this suggest that the alternative routes, with the high frequency value was 18 (Figures 9) for 86 pathway by ITS-ARA*/ARA*.

Figures 8, 9 and 10 show the pathway behavior in the different time windows (from 7 am to 8 am, 11 am to 12 am and 4 pm to 5 pm, respectively). We can observe that pathway 6 is the most used over the three time windows by Dijkstra/A*, with 9, 11 and 15 times respectively. Pathway 6 corresponds to the stretch of the Cacuia Road to the Galeão Road, a very important stretch in the Governador Island. ITS-ARA*/ARA*, for the same pathway 6, had different utilization frequencies: 4, 5 and 4 times for the three time windows. Pathway 19 has a similar behavior, starting with 9, after decrease to 5 and finishing with 9 times for Dijkstra/A*. ITS-ARA*/ARA*, pathway 19, start with 3

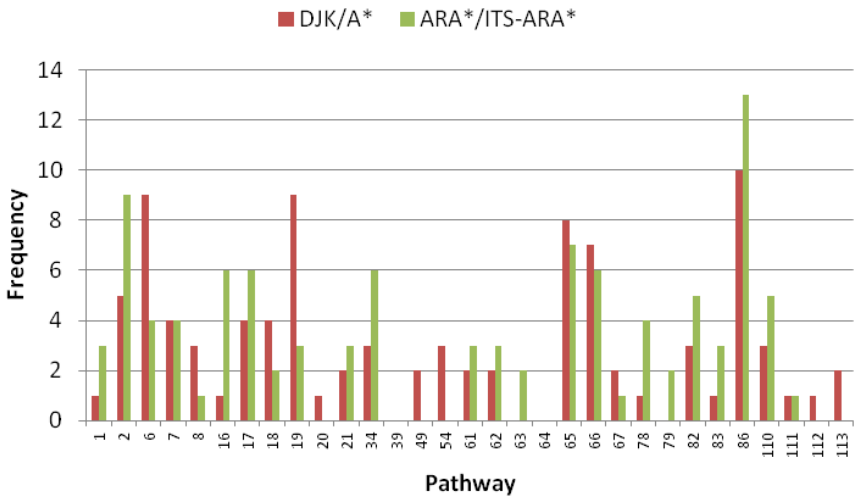


Figure 8: Pathway – 7 am to 8 am – Distance weight.

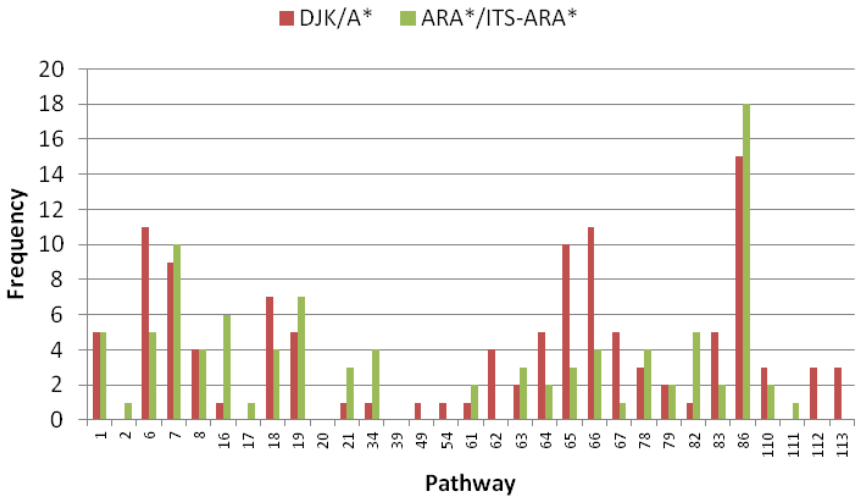


Figure 9: Pathway – 11 am to 12 am – Distance weight.

times and stabilize with 7. The pathway 19 corresponds to the stretch of the Paranapuã Avenue, other important avenue that interconnects four residential districts (Bancários, Cocotá, Moneró and Jardim Carioca). There are pathways

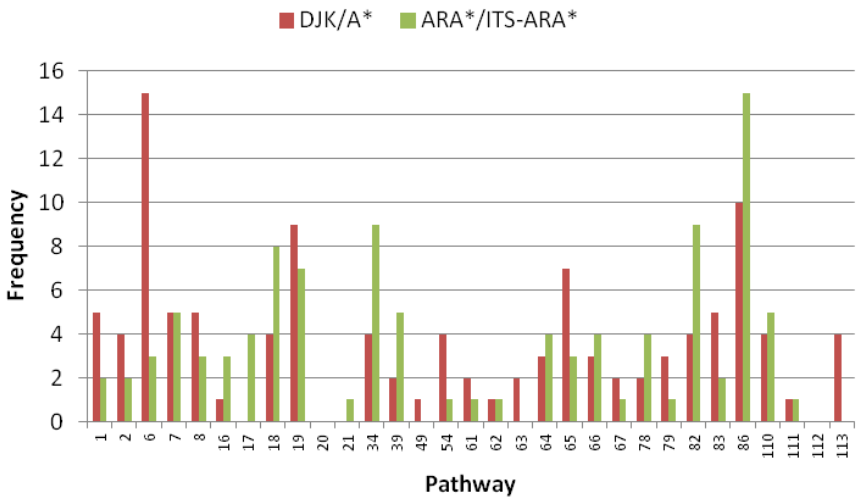


Figure 10: Pathway – 4 pm to 5 pm – Distance weight.

used by only one pair of algorithms, pathway 49 and 113 used by Dijkstra/A*, usage varies from 2 and 1 times and from 2, 3 and 4, respectively. The pathway 49 corresponds to Castorina Faria de Lima Street in the Portuguesa residential district, as well as the pathway 113, where no info record that the traffic jam. Other yet, the pathway 86 corresponds to Galeão Road, very important stretch in the Governador Island.

4.2. Travel Time Weight Results

In this set of experiments, we use the travel time as edge metric for the path computation. Table 2 shows the average path results in relation to the execution time, the memory consumed and the number of the nodes, in the same way as the previous results (Table 1). The optimized ITS-ARA* has been the more fastest of the algorithms, where the gain was in average 93.99% in relation to the Dijkstra, 83.77% in relation to the A* and 3.11% in relation to ARA*. For the memory consumed and the number of nodes, the results are the same, inside the confidence interval. Therefore, we can say that the only difference in performance was the execution time.

Figure 11 shows the average distance in the time windows. We can note that in the three time intervals, there was no difference between the algorithms in each range listed. We can note that the smaller distance traveled occurred in the first time window (7 am to 8 am), around 5,400 meters. We can also see that the average distance results were higher than distance weight cost for the same measure. On the other hand, the algorithm results were the same considering

Table 2: Travel Time weight results – a) Execution Time – execution time in average, given in milliseconds; b) Memory Consumed – memory consumed in average expressed in MB; and c) Nodes – average number of nodes in the path found.

Time Window	Distance			
	DJK	A*	ARA*	ITS-ARA*
	Execution Time (ms)			
07–08 am	377.92	157.01	23.53	22.53
11–12 am	420.33	146.12	23.95	23.61
04–05 pm	367.85	130.79	24.67	23.77
	Consumed Memory (MB)			
07–08 am	159.64	160.97	158.99	160.01
11–12 am	155.60	157.39	155.90	155.92
04–05 pm	157.29	155.73	156.98	156.42
	Average Number of Nodes			
07–08 am	118.49	118.49	116.15	116.15
11–12 am	124.35	124.35	123.69	123.69
04–05 pm	121.29	121.29	122.48	122.48

the confidence interval in the three time windows, differently from the results obtained with the distance metric.

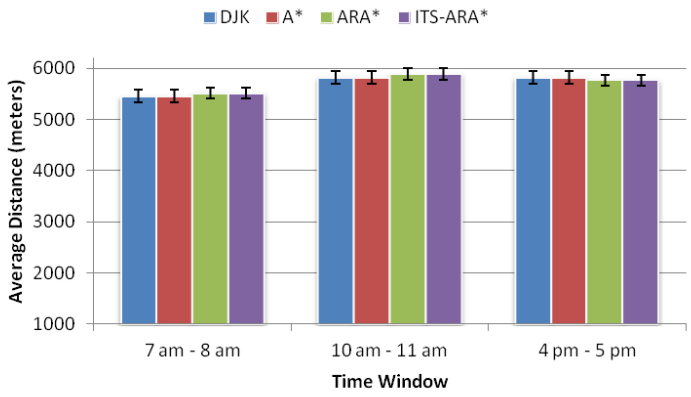


Figure 11: Average Distance (meters) – Travel Time weight.

Average velocity is another measure that confirms the trend. The three algorithms had a rather similar behavior (Figure 12), where the highest average was in the third time window (4 pm to 5 pm, around 60 km/h). We can observe, that when compared to the same distance cost result, the algorithms had the same behavior.

Figure 13 shows the average of the arrival time, where highest average was

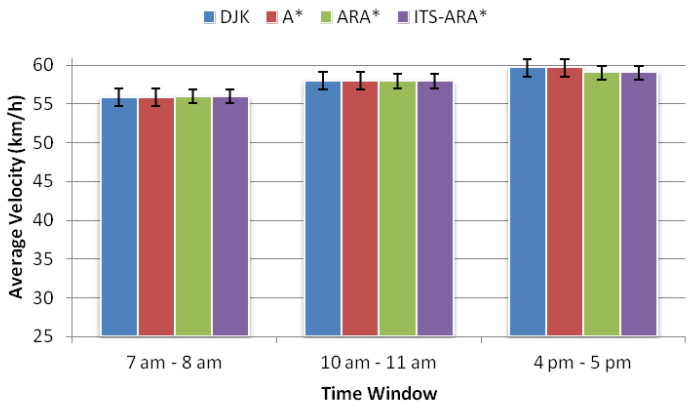


Figure 12: Average Velocity (km/h) – Travel Time weight.

around 6 minutes. However, the results show that there no was no difference in the algorithm values, the third time window represents well this behavior. We can note that, for the arrival time, the results are the same, inside the confidence interval. In this way, taking into consideration the results of the distance, velocity and arrival time, we can conclude that in the travel time metrics, there are no differences between algorithms performance. That way the use of the alternative routes is applicable to the context of the big cities, helping vehicular traffic balancing.

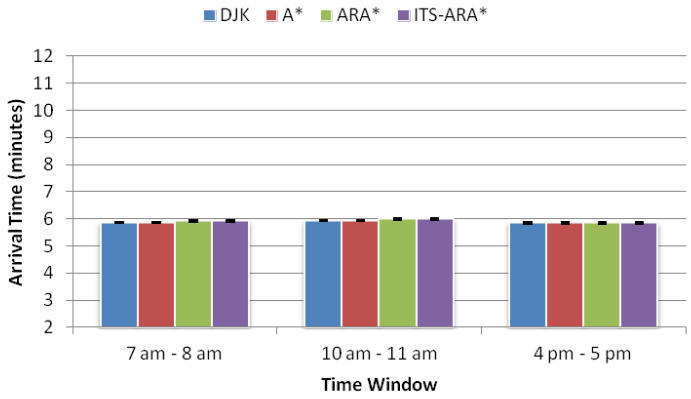


Figure 13: Average Arrival Time (meters) – Travel Time weight.

Travel time costs produce a higher frequency in the pathways, differently from the distance cost, with maximum value of 45. This is due to more uniformity of weights on the edges. In other words, speeds in the streets are more

uniform than the length of the streets. As a consequence, the algorithms had similar performance. Pathway 847 shows an interesting frequency decrease, starting with 45 in two times windows (Figures 14 and 15), and then another decrease to 37 times (Figure 16). Pathway 847 corresponds to Astilbe Street at Jardim Carioca (an urban district). Pathway 1006 corresponds to Galeão Road, which has a speed limit of 80 km/h, shows frequencies around 45 times. Pathway 1007 shows frequencies of 45 for the first two time windows (7 am to 8 am and 11 am to 12 am), and then 35 times.

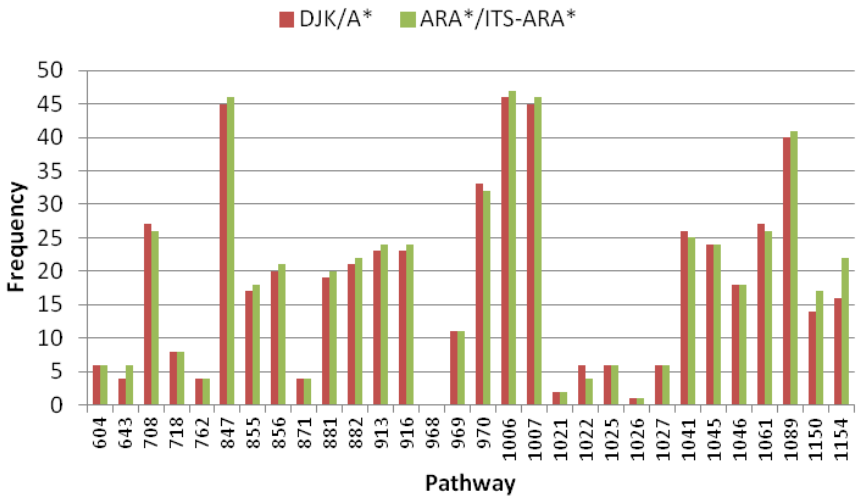


Figure 14: Pathway – 7 am to 8 am – Travel Time weight.

4.3. Pathways: Distance and Travel Times

Figures 17 and 18 plot the Probability Density Function (PDF) of the pathways for each algorithm, respectively for the weights Distance and Travel Time. The X-axis represents the pathway identifiers. Each bar represents a set of 600 pathways grouped. We can observe that there is a reduction in the concentration of the pathway frequency in ITS-ARA* and ARA* (Figures 17(b) and 18(d)), respectively), when compared with A* and Dijkstra (Figures 17(a) and 17(c)). Such a flatter distribution favors a better balance between the pathways. Note that we do not observe the same behavior when the Travel Time weight is used as metric (Figure 18). In this case, all algorithms showed similar results.

We can conclude that, combined with ARA* and ITS-ARA*, the distance metric is a better option in terms of better distributing the vehicular traffic among the different routes, with the side benefit of reducing traffic jams.

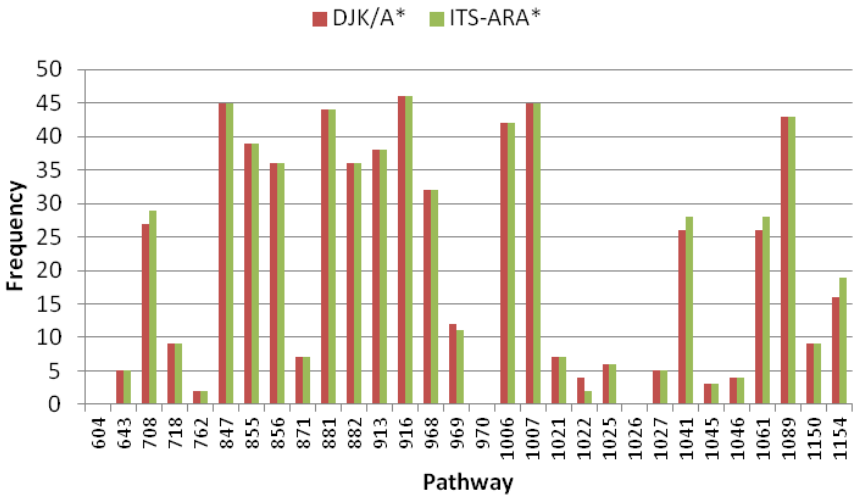


Figure 15: Pathway – 11 am to 12 am – Travel Time weight.

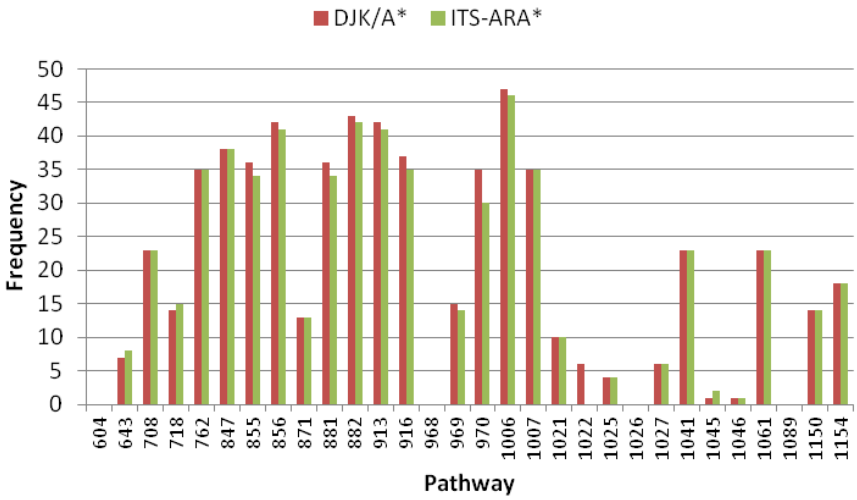
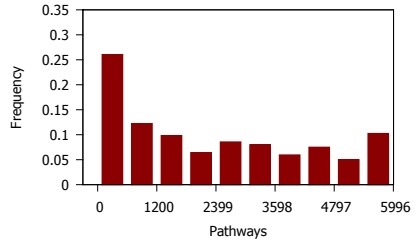
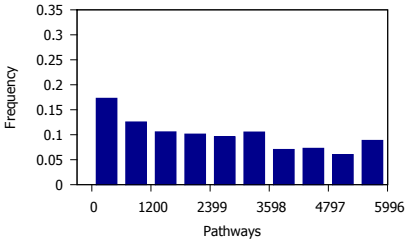


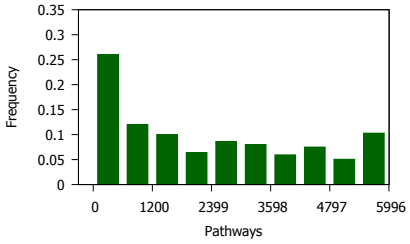
Figure 16: Pathway – 4 pm to 5 pm – Travel Time weight.



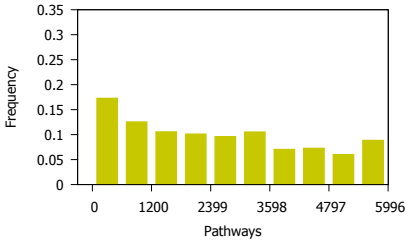
(a) A* algorithm.



(b) ARA* algorithm.

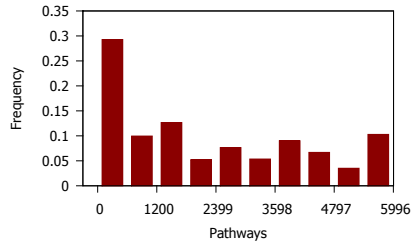


(c) Dijkstra algorithm.

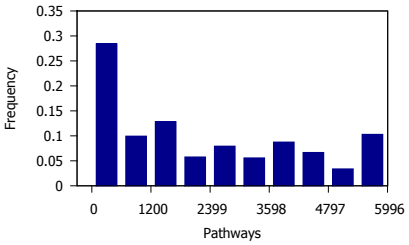


(d) ITS-ARA* algorithm.

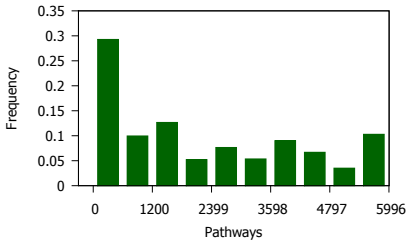
Figure 17: PDF results – 7 am to 8 am – Distance weight.



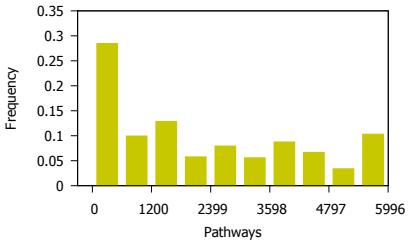
(a) A* algorithm.



(b) ARA* algorithm.



(c) Dijkstra algorithm.



(d) ITS-ARA* algorithm.

Figure 18: PDF results – 7 am to 8 am – Travel Time weight.

5. Conclusion and Future Work

This work presented an experimental analysis of the performance of anytime search algorithms for route planning, running on resource-constrained devices and using a real road network, that of Rio de Janeiro. Our analysis was based on the execution of the different algorithms over a Raspberry Pi prototype. Distance and travel time weights were used in the experiments. Real data extracted from GPS onboard the buses of Rio and OpenStreetMap data were used to calculate the route planning. For the execution time, using distance weight, on average ITS-ARA* algorithm was 15.98 times faster than the Dijkstra algorithm, 5.25 times faster than the A* algorithm and 1.03 times faster than ARA*. For the Travel Time results, on average the ITS-ARA* algorithm was 16.69 times faster than the Dijkstra algorithm, 6.22 times faster than the A* algorithm and 1.03 times faster than ARA*.

ITS-ARA* and ARA* algorithms, keep the speed and efficiency of A* and aggregate the reuse of found results, not requiring new searches when the weights change. This last feature is an advantage for applications of dynamic systems such as traffic on road networks. The experimental results were satisfactory in terms of algorithm performance ITS-ARA*/ARA* and the feasibility of the vehicular Raspberry Pi prototype to support in route planning in the cities, producing alternative routes in a context-sensitive application.

As future work, we intend to analyze the tracks more used in the time window and further explore load balancing techniques issued from other fields. Moreover, we plan to make practical experiments with COTraMS [22] and a real prototype. Another interesting direction is to evaluate the properties of road networks using the complex networks theory and temporal graphs.

Acknowledgements

This work was partially funded by CAPES, CNPq, Fapeam, Faperj and Tribunal de Justiça do Amazonas.

References

[1] Y. Sugiyama, M. Fukui, M. Kikuchi, K. Hasebe, A. Nakayama, K. Nishinari, S. Tadaki, S. Yukawa, Traffic jams without bottleneck - experimental evidence for the physical mechanism of the formation of a jam, Springer New Journal of Physics 10 (3) (2008) 033001.

[2] L. Iftekhar, R. Olfati-Saber, Autonomous driving for vehicular networks with nonlinear dynamics, in: IEEE Intelligent Vehicles Symposium (IV), 2012, pp. 723–729.

[3] Y.-C. Chiu, J. Bottom, M. Mahut, A. Paz, R. Balakrishna, T. Waller, J. Hicks, Dynamic Traffic Assignment: A Primer, 1st Edition, Transportation Research Circular, Transportation Research Board, 2011.

[4] B.-F. Wu, H.-Y. Huang, C.-J. Chen, Y.-H. Chen, C.-W. Chang, Y.-L. Chen, A vision-based blind spot warning system for daytime and nighttime driver assistance, Elsevier Computers & Electrical Engineering 39 (3) (2013) 846–862.

[5] F. Angius, M. Reineri, C. Chiasserini, M. Gerla, G. Pau, Towards a realistic optimization of urban traffic flows, in: IEEE Intelligent Transportation Systems, 2012, pp. 1661–1668.

[6] K. Faez, M. Khanjary, UTOSPF with waiting times for green light consideration, in: IEEE Systems, Man and Cybernetics, 2009, pp. 4170–4174.

[7] W. Kim, M. Gerla, NAVOPT: Navigator assisted vehicular route OPTimizer, in: IEEE Innovative Mobile and Internet Services in Ubiquitous Computing, 2011, pp. 450–455.

[8] D. Ferguson, M. Likhachev, A. Stentz, A guide to heuristic based path planning, in: International Conference on Automated Planning and Scheduling, 2005.

[9] A. Furda, L. Vlacic, Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making, IEEE Intelligent Transportation Systems Magazine 3 (1) (2011) 4–17.

[10] J. C. Mohanta, D. R. Parhi, S. K. Patel, Path planning strategy for autonomous mobile robot navigation using Petri-GA optimisation, Elsevier Computers & Electrical Engineering 37 (6) (2011) 1058–1070.

[11] Tomtom traffic index – measuring congestion worldwide - accessed in November/2015, https://www.tomtom.com/en_gb/trafficindex (2015).

[12] U. Lee, E. Magistretti, M. Gerla, P. Bellavista, P. Liu, K. Lee, Bio-Inspired Multi-Agent Collaboration for Urban Monitoring Applications, 1st Edition, Vol. 5151, Springer, 2008.

[13] C. Li, S. Anavatti, T. Ray, Adaptive route guidance system with real-time traffic information, in: IEEE Intelligent Transportation Systems, 2012, pp. 367–372.

[14] B. Li, J. Gong, Y. Jiang, H. Nasry, G. Xiong, ARA*+: Improved path planning algorithm based on ARA*, in: IEEE Web Intelligence and Intelligent Agent Technology, 2012, pp. 361–365.

[15] A. Mustafa, A. Jan, S. A. Mahmud, Z. Shafiq, G. M. Khan, M. H. Zafar, Point-of-interests based best path selection using cluster-based routing, in: IEEE Wireless Communications Systems ISWCS 2014, 2014, pp. 690–696.

[16] M. Likhachev, G. Gordon, S. Thrun, ARA*: Anytime A* with provable bounds on sub-optimality, in: Neural Information Processing Systems, 2004.

[17] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, S. Thrun, Anytime search in dynamic graphs, Artificial Intelligence 172 (14) (2008) 1613–1643.

[18] S. Zilberstein, Using anytime algorithms in intelligent systems, AI Magazine 17 (3) (1996) 73–83.

[19] M. Likhachev, Search-based Planning for Large Dynamic Environments, Ph.D. thesis, School of Computer Science – Carnegie Mellon University (2005).

[20] M. Haklay, P. Weber, Openstreetmap: User-generated street maps, IEEE Pervasive Computing 7 (4) (2008) 12–18.

[21] T. Hayakawa, Y. Imi, T. Ito, Analysis of quality of data in OpenStreetMap, in: IEEE Commerce and Enterprise Computing CEC, 2012, pp. 131–134.

[22] J. G. Ribeiro Júnior, M. Mitre Campista, L. Costa, COTraMS: A collaborative and opportunistic traffic monitoring system, IEEE Transactions on Intelligent Transportation Systems 15 (3) (2014) 949–958.

[23] V. Vujovic, M. Maksimovic, Raspberry Pi as a sensor web node for home automation, Elsevier Computers & Electrical Engineering 44 (2015) 153–171.

Biographies

Marcus de Lima Braga received his D.Sc. degree in electrical engineering from the Federal University of Rio de Janeiro (UFRJ), Brazil, in 2016, and the M.Sc. degree in electrical engineering from the Federal University of Amazonas (UFAM), Brazil, in 2012. He has been a Systems Analyst at the Court of Justice of Amazonas (TJAM) since September 2007. His major research interests are vehicular networks, embedded systems and formal specification.

Alyson de Jesus dos Santos received his D.Sc. degree in electrical engineering from Federal University of Rio de Janeiro (UFRJ), Brazil, in 2016 and the M.Sc. degree in electrical engineering from Federal University of Amazonas (UFAM), Brazil, in 2011. Currently, he is a Director of Information Technology at the Department of Administration and Management of Amazonas (SEAD-AM).

Aloysio de Castro Pinto Pedroza graduated in Electronic Engineering in 1975 and Masters in Electrical Engineering in 1980 at the Federal University of Rio de Janeiro, Brazil. He obtained the title of Ph.D. in Industrial Automatic Computing at Universit Toulouse III Paul Sabatier, Toulouse, France, in 1985. Currently he is a Full Professor with Federal University of Rio de Janeiro.

Luís Henrique M. K. Costa received his Eng. and M.Sc. degrees in electrical engineering from Federal University of Rio de Janeiro (UFRJ), Brazil, and the Dr. degree from Université Pierre et Marie Curie, Paris, France, in 2001. Since August 2004 he has been an associate professor with COPPE/UFRJ. His major research interests are on Internet of things and vehicular networks.

Marcelo Dias de Amorim received the B.Sc. and M.Sc. degrees in electronic engineering from the Federal University of Rio de Janeiro (UFRJ), Brazil, and the Ph.D. degree from Universit de Versailles, France. He is a research director at the National Center for Scientific Research (CNRS) and member of the computer science laboratory (LIP6) of UPMC Sorbonne Universités, France.

Yacine Doudane received an engineering degree from the National Institute of Computer Science (INI), Algiers, Algeria, in 1998, an M.S. degree from National Institute of Applied Sciences (INSA), Lyon, in 1999, and a Ph.D. degree in computer networks from the Pierre & Marie Curie University, both in France, in 2003. He is currently full professor at University of La Rochelle, France.