New Urban Mobility Algorithm Based On User Experience

Meriem JOUNDI
University Hassan II Casablanca
RITM Laboratory EST
Smart City Team GreenTic
Casablanca, Morocco
meriemjoundi@gmail.com

Mahdi EL ALAOUI
University Hassan II Casablanca
RITM Laboratory CED ENSEM
Smart City Team GreenTic
Casablanca, Morocco
el_alaoui_mahdi@hotmail.fr

Aawatif HAYAR
University Hassan II Casablanca
RITM Laboratory ENSEM
Smart City Team GreenTic
Casablanca, Morocco
aahayar@gmail.com

***Abstract*—Nowadays, getting around has become an essential aspect of everyday life. The urban road traffic of the city of Casablanca the same all the other big cities, is at the heart of many issues related to economy and environment. In order to find a way around these problems, several vehicles routing algorithms have been proposed to compute the shortest path between two given locations in a road traffic network, and considerable efforts have been made to improve routing algorithms, but existing approaches have forgotten that there are several factors that affect the efficiency of the algorithm. In this work we propose a new vehicle routing algorithm taking into account multiple traffic environment factors based on the user experience. Subsequently, we have highlighted the important role of these traffic factors and their impact to select rapid paths. To achieve it, we have conducted a comparison between the proposed algorithm and Dijkstra.**

Keywords: Smart mobility, navigation routing, traffic factors, user experience

I. Introduction

In recent years, Casablanca has become an open-air construction site which makes traffic management more difficult with the number of vehicles that continues to increase. Crowsourcing apps used like waze or foursquare, provide an online routing service to users. But remains incomplete because the navigation is always going to be on the tougher side as there are so many real-time and local context conditions that need consider.
Indeed, a significant amount of effort was deployed to develop algorithms to calculate the shortest path between two specific locations in a road network and rare of them have taken into account user experience and traffic factors.
Accordingly, in this study, we suggest multi routing algorithm which adopts user experience approach and formulates it as an optimization problem of minimizing network congestion. Under this framework, we consider multiple factors. First, to choose the least used path given user experience database. Afterwards, to minimize the number of traffic lights and the number of turns for a given path.

The rest of the paper is organized as follows:
Section 2 present an overview of vehicles routing algorithms as well as the related works. Section 3 shows the application-side analysis which stimulate our model needs and query supports. The problem definition and the body of the algorithm are given in section 4. A comparison is made between the proposed algorithm and Dijkstra in the section 5. Finally, the conclusion and future work are outlined in section 6.

II. Related work

Shortest path problems are the most popular optimization issues. The basic concept of this kind of problem is a specific model of a certain graph and a path which respects a number of constraints to answer a given question. This is also known as a routing algorithm. While it is relatively simple to come up with an algorithm that just solves the problem, it is much harder to develop an efficient, yet accurate algorithm. Furthermore, the problems vary in different scenarios.

Many studies lay to tackle congestion phenomena. [1] proposes Astar-based algorithm for vehicle navigation systems, it has a performance such as computing speed and veracity in a large-scale road network better than a Dijkstra-based one and takes into account several constraints related to the network architecture.
[2] proposes an optimized algorithm to prune off the non-target direction nodes and those not on the way to target nodes, which reduce the amount of the heuristic function calculation.
[3] suggests a set of algorithms that calculate the shortest paths between a source/destination pair in the case of free flow or congested traffic, allowing location-based systems to track the movement of traffic and prevent them in case of emergencies or of critical situations.
[4] proposes an algorithm highlighting adaptive signal control while to stress the close relationship between travellers route choice and traffic signal control in a suitable framework and test the algorithm by simulating in OmNet++ and SUMO.
[5] develops a traffic model based on semi-microscopic approach and predicts the congestion problems.
[6] uses dynamic routing of the city Varanasi based on a Dijkstra algorithm via the PgRouting geospatial database to allow users to make changes to the data and attributes following the changes.
[7] proposes a dynamic navigation system that recommends the route to the desired point of interests (POI) using Location Based Services (LBS) and Geographical Information Services (GIS) taking advantage from their benefits such as real-time navigation, location tracking, traffic control etc.
Our contribution is detailed in the next section.

III. Motivation

This section would give the motivation behind our model.

We propose here to introduce a set of factors that have influence on traffic route planning in a road network. These factors have a huge influence on the travelling time of a certain path.

In reality, people are looking for the quickest path to a destination instead of the shorter path, as there are shorter paths with a significant number of traffic lights that take longer to travelling.

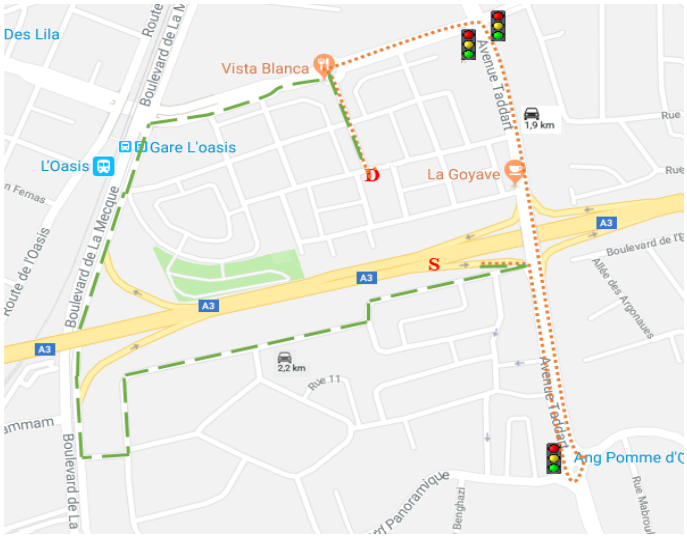

Figure 1: Traffic-light effect illustration

Indeed, figure 1 shows that driving along path of green color will be faster than the path of red color, even if the distance is longer. Among reasons is that the green path do not contain traffic lights and is also less used than the second one.

IV. Algorithm Framework

A. System model

In this section, we introduce the system model, and define the factors considered.

We consider a graph $G = (N, E)$ orientated and bidirectional, with $|N| = V$ the nodes that represent the intersections in our case, and E represent the link between the two nodes. And the following variables are defined :

P_i	=	path from start to destination nodes
t_{P_i}	=	travel time of P_i
f_{P_i}	=	average weekly flow of vehicles that have traveled P_i
f_T	=	total flow traveled throughout the area considered
Nl_i	=	number of traffic light in all the P_i
Nt_i	=	number of turns in all the P_i

We define the following factors:

Let FIT_{P_i} denotes influence factors related to traffic. The factor we will consider here includes the number of traffic light and number of turns in all the path P_i, and is defined as follows :

$$FIT_{P_i} = Nl_i + \ln(1 + Nt_i)$$

The reason for assigning the logarithmic function to the variable Nt_i, is that the number of turns in a path is less influential than the number of traffic light in the same path.
Let Π_{P_i} denotes the probability that path P_i is used, weighted by travel time, and is defined as follows :

$$\Pi_{P_i} = t_{P_i} \times \frac{f_{P_i}}{f_T}$$

Our objective is to find all the paths that lead from a source to a destination that minimizes the two factors defined above.

B. Layout Algorithm

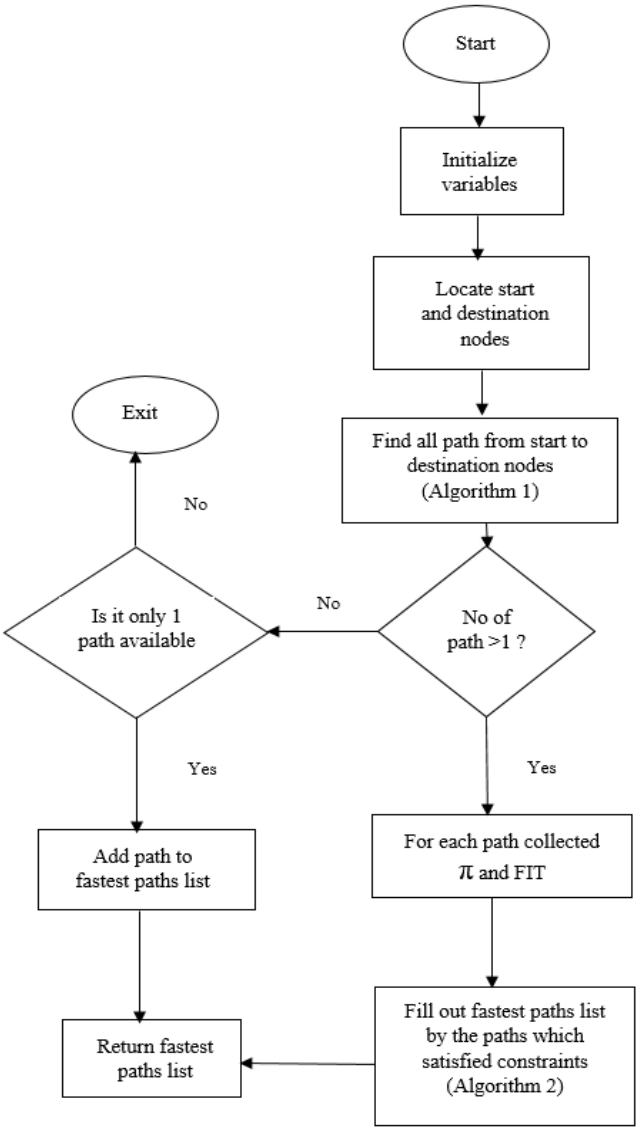

Figure 2: Layout algorithm

As shown in figure 2, the algorithm starts by initialization of the variables, then locate start and destination nodes. Thereafter, it breaks and starts a new path search from the starting node through a recursive function to determine all paths that link between a source and a given destination. If no path is found, the algorithm ends. Else, if only one path is available between s and d nodes, the algorithm add path to fastest path list. If many paths exist, then for each paths, collected the probability that path be used weighted by travel time and influence factors related to traffic. At the end, the algorithm fill out fastest paths list by the dominates paths.

C. Preprocessing

Our approach is based on two steps:

Algorithm 1 : call a recursive function that allows to list all the paths that lead from a source to a destination.

Algorithm 2 : return fastest paths list taking into account on the two factors Π and FIT, by calling the recursive function defined below.

Algorithm 1: Find all paths between two nodes

Function *FindAllPaths*

Data:
u : current node;
d : destination node;
$VN(u)$: keeps track of nodes in current path

Result:
$OpenList()$: List of all paths;
k : Path count;

/* Mark current node u visited and store it in path $P_k()$ */
$VN(u) = True$;
$P_k.add(u)$;

/* If current node is same as destination then reconstruct the current path */
if $u = d$ **then**
  $k = k + 1$;
  $OpenList.add(P_k)$;

foreach $v \in adj(u)$ **do**
  /* If current node is not destination then recur for all the nodes adjacent to current node */
  if $VN(v) = False$ **then**
    Go to Function $FindAllPaths$;

/* Remove current node from $P_k()$ and mark it as unvisited */
$P_k.remove(u)$;
$VN(u) = False$;

Algorithm 2: Fastest paths between start and destination nodes

Function *FastestPaths*

Data:
V : number of nodes
s : start node
d : destination node
Π_{P_k} : probability that path P_k be used, weighted by travel time
FIT_{P_k} : influence factors related to traffic

Result:
$CloseList$: list forming the fastest paths

/* Initialize path count */
$k = 0$; /* Mark all nodes as not visited */
$VN(u) = false \ \forall \ u \ \in \{1, \ldots, V\}$;
/* Call the recursive function to find all paths between "s" and "d" and their count "k" */
Go to Function $FindAllPaths$;
if $k > 1$ **then**
  for $i \ \in \{1, \ldots, k\}$ **do**
    if $CloseList$ *is empty* **then**
      $CloseList.add(P_i)$;
    else Request Π_{P_i} and FIT_{P_i};
    foreach $P_j \in CloseList$ *except* $\{P_i\}$ **do**
      Request Π_{P_j} and FIT_{P_j};
      if *(*$\Pi_{P_i} < \Pi_{P_j}$ *and* $FIT_{P_i} < FIT_{P_j}$*) or (*$\Pi_{P_i} = \Pi_{P_j}$ *and* $FIT_{P_i} < FIT_{P_j}$*) or (*$\Pi_{P_i} < \Pi_{P_j}$ *and* $FIT_{P_i} = FIT_{P_j}$*)* **then** /* P_i dominates P_j */
        $CloseList.remove(P_j)$;
        $CloseList.add(P_i)$;
      else if *(*$\Pi_{P_i} > \Pi_{P_j}$ *and* $FIT_{P_i} > FIT_{P_j}$*) or (*$\Pi_{P_i} = \Pi_{P_j}$ *and* $FIT_{P_i} > FIT_{P_j}$*) or (*$\Pi_{P_i} > \Pi_{P_j}$ *and* $FIT_{P_i} = FIT_{P_j}$*)* **then**/* P_j dominates P_i */
        $DoNothing()$;
      else /* P_i and P_j are incomparable */
        $CloseList.add(P_i)$;
    Return $CloseList$;
else if $k = 1$ **then**/* It exists only one path */
  $CloseList.add(P_k)$;
  Return $CloseList$;
else Exit; /* No path is found */

V. Results and Analysis

This section concerns a simulation on a network of 12 nodes illustrated in figure 3, in order to compare between the proposed algorithm and Dijkstra's one.

Table I: *All Paths between* $s = 1$ *and* $d = 12$

P_i	d_i (km)	t_{P_i} (min)	f_{P_i}/f_T	Π_{P_i} (min)	Nl_i	Nt_i	FIT_i
$P_1 = [1, 2, 3, 4, 5, 6, 11, 12]$	2,06	9	0,09	0,81	2	8	4,19
$P_2 = [1, 2, 3, 4, 5, 12]$	1,86	7,5	0,085	0,6375	2	5	3,79
$P_3 = [1, 2, 3, 4, 6, 5, 12]$	2,07	7	0,05	0,35	2	8	4,20
$P_4 = [1, 2, 3, 4, 6, 11, 12]$	2,05	7,5	0,09	0,675	2	8	4,20
$P_5 = [1, 2, 3, 7, 11, 12]$	1,9	7	0,105	0,735	2	5	3,79
$P_6 = [1, 10, 9, 8, 7, 3, 4, 5, 6, 11, 12]$	2,96	8	0,085	0,68	1	11	3,48
$P_7 = [1, 10, 9, 8, 7, 3, 4, 5, 12]$	2,76	6,5	0,04	0,26	1	8	3,20
$P_8 = [1, 10, 9, 8, 7, 3, 4, 6, 5, 12]$	2,97	6	0,065	0,39	1	11	3,48
$P_9 = [1, 10, 9, 8, 7, 3, 4, 6, 11, 12]$	2,95	6,5	0,095	0,6175	1	10	3,40
$P_10 = [1, 10, 9, 8, 7, 11, 12]$	2,2	4	0,065	0,26	0	7	2,08
$P_11 = [1, 10, 9, 11, 7, 3, 4, 5, 12]$	2,77	9,5	0,08	0,76	1	10	3,40
$P_12 = [1, 10, 9, 11, 7, 3, 4, 6, 5, 12]$	2,98	9	0,06	0,54	1	12	3,56
$P_13 = [1, 10, 9, 11, 12]$	2,04	5	0,09	0,45	0	7	2,08

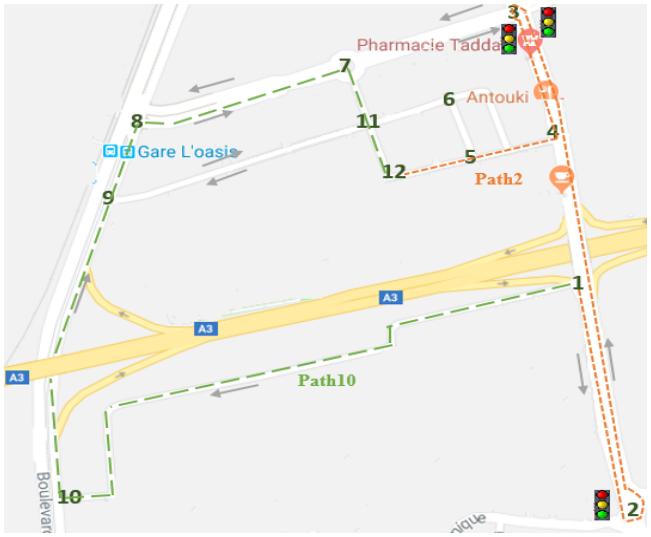

Figure 3: Network considered

The table I shows the thirteen paths that link between the two nodes s = 1 and d = 12 of the network considered, as well as the distance, the number of traffic lights and turns corresponding to each path. The different values assigned to the Π_{P_i} factor depend on the user experience.

Analysis 1 : the path 2 recommended by the Dijkstra algorithm takes more time despite its minimal distance, but the path 10 recommended by our algorithm has a greater distance but optimal in terms of time, which justifies the importance of factor related to the number of traffic lights and turns FIT.

Analysis 2 : the path 13 has not been selected by our algorithm despite having a lower distance and the same $FIT_{13} = FIT_{10}$ as path 10 because it is more used by users $\Pi_{P_{13}} > \Pi_{P_{10}}$ and therefore congested.

VI. Conclusion

In this article, we proposed a new routing algorithm approach to optimize urban mobility.
The reason behind this work is to show the important role of user experience and traffic factors to find the fastest path between two locations. This work aims also at exploring more the potential from involving and engaging users in a collaborative approach to contribute to improve traffic conditions. In this paper we have shown preliminary results which demonstrate the added value, and how they lead to fast and practical search results based on the user experience in comparison with the Dijkstra algorithm.
Work in progress includes the generalization on a global map of Casablanca and the generalization of traffic factors, also the implementation of a traffic simulation platform of empirical experiences of the proposed routing algorithm.

References

[1] W. YIN and X. YANG, *A Totally Astar-based Multi-path Algorithm for the Recognition of Reasonable Route Sets in Vehicle Navigation Systems*, Social and Behavioral Sciences. 96, 1069-1078, 2013.

[2] Z. ZHAO and R. LIU, *An Optimized Method for A* Algorithm Based on Directional Guidance*, Software Engineering and Service Science (ICSESS), 6th IEEE International Conference on, 2015.

[3] A. FARO and D. GIORDANO, *Algorithms to find shortest and alternative paths in free flow and congested traffic regimes*, Transportation Research Part C. 73, 1-29, 2016.

[4] H. CHAI, H.M. ZHANG, D. GHOSAL and C. CHUAH, *Dynamic traffic routing in a network with adaptive signal control*, Transportation Research Part C. 85, 64-85, 2017.

[5] L. KARIM, A. DAISSAOUI and A. BOULMAKOUL, *Robust routing based on urban traffic congestion patterns*, The 8th International Conference on Ambient Systems, Networks and Technologies. 2017.

[6] P. KUMARI and R.D. GARG, *Identification of Optimum Shortest Path using Multipath Dijkstras Algorithm Approach*, International Journal of Advanced Remote Sensing and GIS. V 6, N 1, 2442-2448, 2017.

[7] S. NAYAK and M. NARVEKAR, *Real-Time Vehicle Navigation using Modified A* Algorithm*, International Conference on Emerging Trends and Innovation in ICT (ICEI). 2017.