ARTICLE IN PRESS

Applied Mathematics and Computation xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city

José L. Galán-García *, Gabriel Aguilera-Venegas, María Á. Galán-García, Pedro Rodríguez-Cielos

Department of Applied Mathematics, University of Málaga, Spain

ARTICLE INFO

Keywords:
Extension of Dijkstra's Algorithm
Accelerated-time simulations
Smart cities
Probabilistic algorithms

ABSTRACT

Dijkstra's algorithm to solve the shortest path problem (SPP) is a very well-known algorithm. When applied to real situations, although the shortest path can be computed with Dijkstra's algorithm, it is not always the one that is chosen. In traffic situations, for example, the driver may not know the exact length of the lanes or the shortest path to follow. Even more compelling is the fact that although the driver knows the shortest route, he/she may prefer choosing a different route.

In this paper we present the new algorithm PEDA (Probabilistic Extension of Dijkstra's Algorithm) which introduces probabilistic changes in the weight of the edges and also in the decisions when choosing the shortest path. When PEDA is applied to traffic flow, more realistic simulations, in which the shortest path is not always chosen, are obtained. This more accurately simulates the more normal behavior of drivers.

As an example of an application, we introduce the ATISMART$^+$ model, an extension of the ATISMART model, where an accelerated-time simulation of car traffic in a smart city was described. In that previous work, all cars in the system used Dijkstra's algorithm to choose improved ATISMART$^+$ model, for accelerated time simulations of traffic flow in smart cities, uses the new PEDA algorithm. The results obtained show that ATISMART$^+$ produces more realistic simulations considering different drivers' behaviors.

1. Introduction

Car traffic has become one of the most important problems facing cities during the last decades. Problems arise in different areas:

- *Pollution*: slow traffic flow and jams provoke an increase in gas emissions which seriously increase the greenhouse effect.
- *Economy*: the amount of fuel consumption produces huge loss of money and natural resources. Additionally, wasted time in traffic jams turns into huge economic losses.
- *Emergencies*: police cars, ambulances, fire trucks and other emergency vehicles have severe delays due to traffic jams which can even lead to loss of life.

* Corresponding author.

E-mail addresses: jlgalan@uma.es (J.L. Galán-García), gabri@ctima.uma.es (G. Aguilera-Venegas), magalan@ctima.uma.es (M.Á. Galán-García), prodriguez@uma.es (P. Rodríguez-Cielos).

http://dx.doi.org/10.1016/j.amc.2014.11.076
0096-3003/

Please cite this article in press as: J.L. Galán-García et al., A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city, Appl. Math. Comput. (2014), http://dx.doi.org/10.1016/j.amc.2014.11.076

- *Mental health*: anxiety, frustration, stress, depression, among others, are some of the effects that traffic congestion can cause.

For example, the study developed in [1] estimated that "the monetized value of $PM_{2.5}$-related mortality attributable to congestion in 83 cities in 2000 was approximately $31 billion (2007 dollars), as compared with a value of time and fuel wasted of $60 billion".

Without doubt, slow traffic flow and jams are problems which highly decrease the quality of life.

Nowadays, Smart cities try to face the traffic congestion problems, among others, with different approaches. The use of Smart Traffic Signals (in the following Smart Signals) as reversing lanes signals and smart traffic lights (which can dynamically change the red and green periods) are one of the most important techniques. Examples of use of these kind of Smart Signals can be found in [2,3].

A physical implementation of such smart traffic lights and reversible lanes is very useful but it is also expensive. Therefore, computer simulations of the traffic behavior can lead to a good design of the system and hence, savings in costs.

The ATISMART model [3] was developed by the authors of this paper using these types of computer simulations. In this model, Dijkstra's algorithm [4] is used to compute the path that a car follows from a starting point to its destination. Dijkstra's algorithm is one of the most well-known algorithms to solve the shortest path problem (SPP). But, when applied to real situations, although the shortest path can be computed with Dijkstra's algorithm, it is not always the path that is the chosen one. There are many situations where the driver does not know the shortest path to follow. Furthermore, even if the driver knows the shortest path, he/she may prefer choosing a different one.

There are different studies [5–7] where extensions of Dijkstra's algorithm are used when the lengths of edges are not fixed. These extended versions of Dijkstra's algorithm can be adapted in order to simulate real situations in traffic. Other approaches consist of considering fixed lengths on edges and introducing some variations simulating the drivers' behaviors.

In this work we present the new PEDA algorithm (Probabilistic Extension of Dijkstra's Algorithm) in order to simulate real situations in which the shortest path is not always chosen. This approach includes modifications on the way that Dijkstra's Algorithm calculates the path with non-fixed lengths on edges (simulating the fact that a driver may not know the right length of the lanes).

As an example of an application of the PEDA algorithm, we will extend the ATISMART model to the new model ATISMART$^+$. This model includes more realistic situations considering different drivers' behaviors. A prototype of ATISMART$^+$ was introduced, as a talk, during the ESCO 2014 conference. The abstract of this talk can be found in [8].

As in the previous model, the implementation of ATISMART$^+$ is carried out using a Computer Algebra System (CAS), specifically MAXIMA, together with a graphic user interface, developed in JAVA.

In Section 2 a brief summary of the ATISMART model is shown. In Section 3, the new PEDA algorithm is introduced. In Section 4, the ATISMART model is extended to the new ATISMART$^+$ model as an example of use of PEDA. Finally, in Sections 5 and 6, some conclusions and acknowledgments are presented.

2. Background: the ATISMART model

In order to make the paper self-contained, in this section a brief summary of the ATISMART model is shown. For a complete description of this model with examples, see [3].

2.1. Main characteristics of the ATISMART model

The ATISMART model produces accelerated time simulations of car traffic behavior using smart traffic lights and reversible lanes. The theoretical basis of the model combines ideas from Neural Network and Cellular Automaton theories [9,10].

The ATISMART model deals with smart traffic signals (traffic lights and reversible lanes) and uses a matrix M for representing a part of a city. Specifically, the considered map corresponds to a neighborhood of Málaga, Spain, but it can be easily adapted to any city map. The matrix, M, stores the different elements of the map (lines, directions, intersections, traffic lights, entrances and exits). Cars enter and leave the system using any of the different entrances and exits.

The map is also represented as a weighted directed graph in which all access and exit points and all intersections are the nodes. The edges are defined according to the lanes connecting the points associated with the nodes. The weight assigned to each edge corresponds to the length of the lane.

In order to simulate how the cars access the system, a probability distribution for each input is considered. By default, a Poisson distribution with a dynamically changeable parameter is used.

Once a car enters the system, a state vector, associated with it, dynamically stores all the information on the car. Among other information, the state vector contains the target exit (randomly chosen) and the car's current position and speed.

Any car in the system computes the path to its exit using Dijkstra's algorithm. This is done not only at the starting point but also at any time the car reaches an intersection, since the graph may be modified because some lanes may be reversed.

The car leaves the system when it reaches its target exit.

Please cite this article in press as: J.L. Galán-García et al., A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city, Appl. Math. Comput. (2014), http://dx.doi.org/10.1016/j.amc.2014.11.076

2.2. On the implementation of the ATISMART model

The implementation of the ATISMART model was carried out combining a Computer Algebra System (CAS), where the mathematical engine of the model was implemented, and a Graphical User Interface (GUI) used to communicate with the user and to provide a graphical display of the simulation.

The CAS used was MAXIMA, a software package that is under the GNU General Public License (GPL) downloadable from: http://maxima.sourceforge.net/

The GUI was implemented using JAVA, freely available at: https://www.java.com/en/

Since both MAXIMA and JAVA are multi-platform, ATISMART can be run on most widely extended operative systems.

3. PEDA: a Probabilistic Extension of Dijkstra's Algorithm

Dijkstra's algorithm [4] efficiently provides an optimal path for moving from a starting node to a target node in a weighted graph. But when a person estimates a path to follow in a real situation (for example when driving between two different points in a city), the shortest driving time is not always obtained. Therefore, using Dijkstra's algorithm to simulate the behavior of drivers is not realistic.

To achieve more realistic simulations, we introduce in this paper the new PEDA algorithm, a Probabilistic Extension of Dijkstra's Algorithm based on two different approaches:

- P–1. In most situations, drivers do not know the exact length of lanes. In this first approach, we introduce probability distributions to change the real length of lanes simulating the possible drivers' lack of accuracy.
- P–2. Although drivers could know the exact length of lanes, they may not know how to calculate the optimal path. To simulate this behavior, we introduce two different probabilistic parameters, p_v and p_h. The greater these parameters are, the further the chosen path is from the optimal solution.

The PEDA algorithm is a combination of approaches P–1 and P–2. If the probability distribution considered in P–1 is the one which assigns the right length to each lane and both, p_v and p_h in P–2 are equal to zero, then the PEDA algorithm agrees with Dijkstra's algorithm. In this sense, PEDA is an extension of Dijkstra's algorithm.

3.1. Description of P–1

As mentioned before, P–1 deals with the fact that drivers normally do not know the exact length of lanes. To simulate this situation, a probability distribution is considered to assign the length to the lanes. But this is done only to compute the path to follow from the starting point to the target. Once the path is computed, the real length of lanes is used to calculate the exact length of the path.

P–1 has two associated parameters: the first one, P–1_1, is the considered probability distribution, while P–1_2 is the associated parameter for the probability distribution.

By default, the normal distribution $\mathcal{N}(\mu_i, \sigma)$ is used for P–1_1, where μ_i is the exact length of lane i and $\sigma =$ P–1_2 is the standard deviation.

Another option for P–1_1 is the continuous uniform distribution, $\mathcal{U}_i$, in the interval $[(1-p)\mu_i, (1+p)\mu_i]$, where μ_i is the exact length of lane i and $p =$ P–1_2 is the maximum error (parts per unit) in the length of lanes.

Given a weighted graph G, after applying P–1, a new graph G_P is obtained. Edges and vertices remain equal but the weighs associated with each edge are updated using the values obtained according to the assigned distribution function with its corresponding parameter(s). In this stage, Dijkstra's algorithm is applied to G_P to compute the shortest path. The real cost of the obtained path is the length of this path in the original graph G.

3.2. Example of P–1

Let us consider the graph G given in Fig. 1. G is the graph associated with the lanes of a neighborhood of the city of Málaga that will also be used as the map for the ATISMART[+] model.

As an example of the use of P–1, we consider the computation of a path from input 1 (In1) to output 6 (O6).

When Dijkstra's algorithm is used, an optimal path is obtained. Specifically, the resulting path is:

$$\text{In1-a-n-m-l-k-s-B-z-I-N-d-f-O6} \qquad (\text{length}: \ 147).$$

Some examples of execution of P–1 are shown in Table 1. Some observations are:

- $\mathcal{N}(\mu_i, \sigma)$ denotes the Normal Distribution with mean μ_i (the exact length of lane i) and σ, the standard deviation (parameter P–1_2).

Please cite this article in press as: J.L. Galán-García et al., A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city, Appl. Math. Comput. (2014), http://dx.doi.org/10.1016/j.amc.2014.11.076

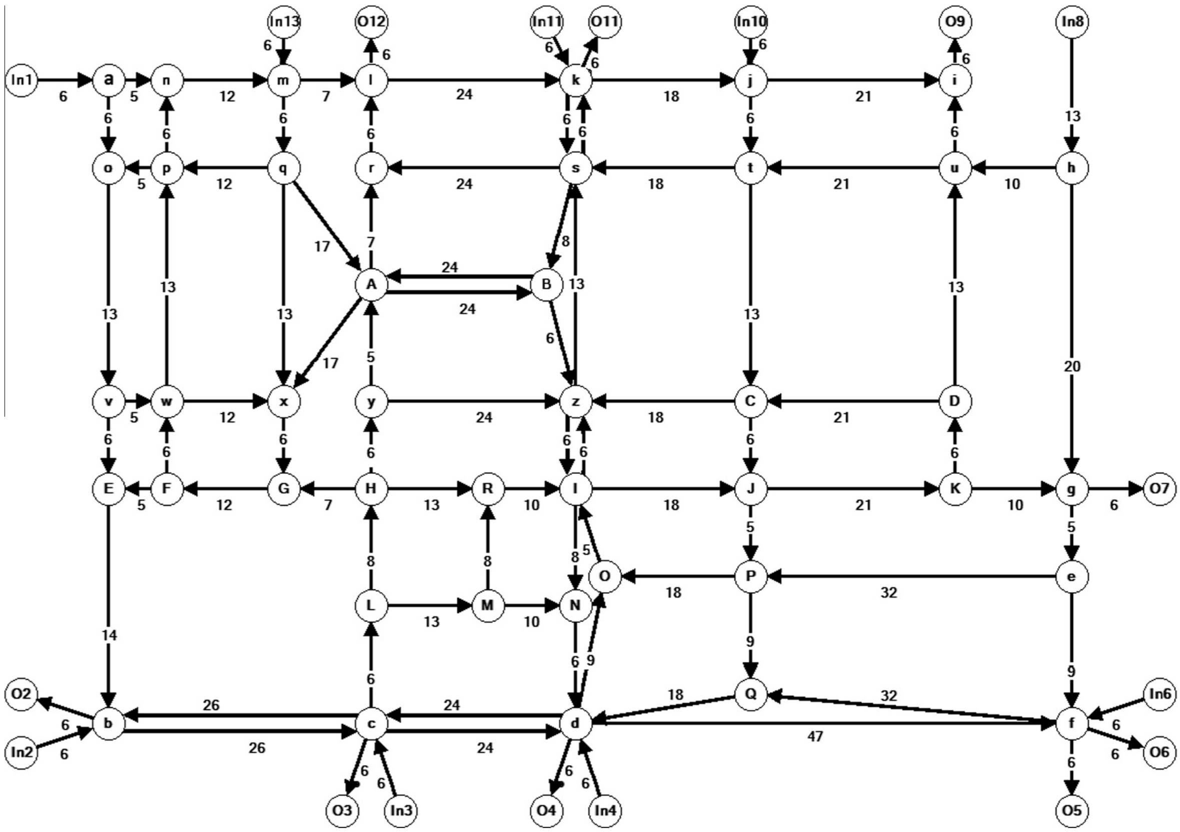


Fig. 1. Graph *G*.

Table 1
Examples of P–1.

P–1_1	P–1_2	Path	Length
$\mathcal{N}(\mu_i, \sigma)$	$\sigma = 1$	In1-a-n-m-l-k-s-B-z-I-N-d-f-O6	147
$\mathcal{N}(\mu_i, \sigma)$	$\sigma = 5$	In1-a-o-v-E-b-c-d-f-O6	148
$\mathcal{N}(\mu_i, \sigma)$	$\sigma = 10$	In1-a-n-m-l-k-j-t-C-J-K-g-e-f-O6	148
$\mathcal{U}[(1-p)\mu_i, (1+p)\mu_i]$	$p = 0.1$	In1-a-n-m-l-k-s-B-z-I-N-d-f-O6	147
$\mathcal{U}[(1-p)\mu_i, (1+p)\mu_i]$	$p = 0.5$	In1-a-o-v-E-b-c-d-f-O6	148
$\mathcal{U}[(1-p)\mu_i, (1+p)\mu_i]$	$p = 0.9$	In1-a-n-m-q-A-B-z-I-J-K-g-e-f-O6	151

- $\mathcal{U}[(1-p)\mu_i, (1+p)\mu_i]$ denotes the Continuous Uniform Distribution in the interval $[(1-p)\mu_i, (1+p)\mu_i]$ where μ_i is the exact length of lane i and p is the maximum error (parts per unit) in the length of lanes (P–1_2).
- Different paths are obtained from different parameters of P–1. In many cases, the optimal path provided by Dijkstra's algorithm is not obtained.
- In both cases of P–1_1 (Normal and Uniform distributions), when the corresponding parameter P–1_2 increases, the probability of greater error in determining the length of the lanes is higher. Consequently, the probability of obtaining a longer path with respect to the optimal one, is higher when P–1_2 increases. Since these results are non-deterministic (depend on the probabilistic parameters), even with high values of P–1_2, the optimal path could be obtained.
- Note that the length of the generated paths are not very far from the optimal length. This is a realistic behavior since, although the driver may have some doubts in the length of the lanes, they normally know approximately which lanes are longer than others. This is enough to obtain a "good" solution with Dijkstra's algorithm.

3.3. Description of P–2

P–2 simulates that the driver, even knowing the exact length of the lanes, may make some errors in the selection of the better path or even may prefer driving on a non-optimal path.

P–2 has two associated parameters: $p_v = \text{P–}2_1$ and $p_h = \text{P–}2_2$. Both of them introduce the probability of a mistake when using Dijkstra's algorithm. In order to describe how p_v and p_h act, let us briefly describe Dijkstra's algorithm introducing some modifications involving p_v and p_h.

Let $G = \{V, E, W\}$ be a weighted graph where V is the set of vertices and $E \subseteq V \times V$, the set of edges and $W : E \times E \longrightarrow \mathbb{R}$ the weight function. Our goal is to find a path from the starting vertex a to the target one, z.

1. Assign zero to a ($L(a) = 0$) and ∞ to the other vertices w_j ($L(w_j) = \infty$) and select $v = a$ as the current vertex.
2. Check all the neighbors v_j to the current vertex v. For each v_j, compute $N_j = L(v) + W(v, v_j)$.
3. With probability $1 - p_v$, assign $L(v_j) = \min(L(v_j), N_j)$ and with probability $p_v, L(v_j)$ remains the same.
4. Sort v_j in ascending order with respect to $L(v_j)$. Let s be the first element of this ordered list or, with probability p_h, the second element if exists.
5. Set $v = s$ as the current vertex.
6. If $v = z$ end. Otherwise, go back to step 2.

In step 3, the value $p_v = \text{P–}2_1$ is introduced. In this step, with probability p_v, the value that Dijkstra's algorithm would assign, is not selected. In the same way, the value $p_h = \text{P–}2_2$ is introduced in step 4, where, with probability p_h, the vertex with minimal value is not chosen. Note that when $p_v = p_h = 0$, this algorithm matches Dijkstra's algorithm.

3.4. Example of P–2

As an example of the use of P–2, we consider again the computation of a path from input 1 (In1) to output 6 (O6). Remember that the path obtained using Dijkstra's algorithm is:

$$\text{In1-a-n-m-l-k-s-B-z-I-N-d-f-O6} \qquad (\text{length}: \; 147).$$

Some examples of the execution of P–2 are shown in Table 2. Note that different paths are obtained from different parameters of P–2. In many cases, the optimal path provided by Dijkstra's algorithm is not obtained.

Paths are obtained in a non-deterministic way since the algorithm is a probabilistic one. For this reason, we have developed a new example of use, where the algorithm is run 10 times in the same conditions for different values of the parameters, in order to obtain an average length for each case. Let us consider now the problem of obtaining a path from Input 11 (In11) to Output 6 (O6). An optimal path, found using Dijkstra's algorithm is:

$$\text{In11-k-s-B-z-I-N-d-f-O6} \qquad (\text{length}: \; 99).$$

The data obtained in the new experiment, varying p_v from 0.1 to 1 in steps of 0.1, is shown in Table 3.

From these data we can observe that the higher p_v is, the further from the optimal path the solution is. This fact is more visually shown in Fig. 2. Note also that when p_v increases, different paths are obtained. Nevertheless, when $p_v = 1$ just one path is obtained. This is an expected result since when $p_v = 1$ the probabilistic decisions are deterministic (taken with probability 1).

Note that changes start with $p_v > 0.5$ and the length (in the mean) of the obtained paths increases up to 134 which is approximately a 35% more than the optimal value 99.

Let us remark that all the results are probabilistic. This means that even with high values of the parameters, the optimal solution or near to optimal can be obtained. The variability of options also depends of the specific characteristics of the graph.

3.5. The PEDA algorithm: description and example

As mentioned before, the PEDA algorithm combines P–1 and P–2. Therefore, the PEDA algorithm deals with 4 parameters: the probability distribution and its parameter(s), p_v and p_h. The specific values for these 4 parameters depend on the characteristics of the problem to be tackled. The former two parameters will provide a new weight function while the last two parameters may result in taking different probabilistic alternatives in the selection of the path.

Table 2
Examples of P–2.

P–2_1	P–2_2	Path	Length
$p_v = 0.1$	$p_h = 0$	In1-a-n-m-l-k-s-B-z-I-N-d-f-O6	147
$p_v = 0.5$	$p_h = 0$	In1-a-o-v-E-b-c-d-f-O6	148
$p_v = 0$	$p_h = 0.1$	In1-a-n-m-l-k-s-B-z-I-N-d-f-O6	147
$p_v = 0$	$p_h = 0.5$	In1-a-o-v-E-b-c-d-f-O6	148
$p_v = 0.1$	$p_h = 0.1$	In1-a-n-m-l-k-s-B-z-I-N-d-f-O6	147
$p_v = 0.5$	$p_h = 0.5$	In1-a-o-v-E-b-c-d-f-O6	148

Please cite this article in press as: J.L. Galán-García et al., A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city, Appl. Math. Comput. (2014), http://dx.doi.org/10.1016/j.amc.2014.11.076

Table 3
Example varying p_v.

p_v	Paths	Times	Length	Mean
$0.1,\ldots,0.5$	In11-k-s-B-z-I-N-d-f-O6	10	99	99
0.6	In11-k-s-B-z-I-N-d-f-O6	8	99	102.6
	In11-k-j-t-C-J-K-g-e-f-O6	1	100	
	In11-k-j-t-C-J-P-Q-d-f-O6	1	134	
0.7	In11-k-s-B-z-I-N-d-f-O6	4	99	116.6
	In11-k-j-t-C-J-K-g-e-f-O6	1	100	
	In11-k-j-t-C-J-P-Q-d-f-O6	1	134	
	In11-k-j-t-C-z-I-N-d-f-O6	4	134	
0.8	In11-k-s-B-z-I-N-d-f-O6	2	99	123.6
	In11-k-j-t-C-J-K-g-e-f-O6	1	100	
	In11-k-j-t-C-z-I-N-d-f-O6	6	134	
	In11-k-j-t-C-J-P-Q-d-f-O6	1	134	
0.9	In11-k-s-B-z-I-N-d-f-O6	1	99	130.4
	In11-k-j-t-C-J-K-g-e-f-O6	2	100	
	In11-k-j-t-C-z-I-N-d-f-O6	6	134	
	In11-k-s-B-A-x-G-F-E-b-c-d-f-O6	1	201	
1	In11-k-j-t-C-z-I-N-d-f-O6	10	134	134

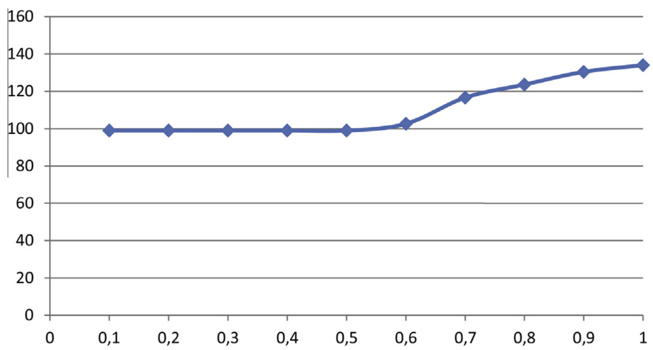


Fig. 2. Mean of lengths varying p_v.

As an example of PEDA, we now compute the path from Input 8 (In8) to Output 2 (O2). The optimal path obtained using Dijkstra's algorithm is:

$$\text{In8-h-u-t-C-J-P-Q-d-c-b-O2} \qquad (\text{length}: \ 151).$$

The data generated with different values for the four parameters are shown in Table 4. Once again, different paths than the optimal one from Dijkstra's algorithm have been obtained.

4. An example of use: the ATISMART+ model

As an example of use of the new PEDA algorithm, we have extended the ATISMART model [3] to the new ATISMART+ model. In this new model, Dijkstra's algorithm has been substituted by the extended algorithm PEDA. The ATISMART+ model provides simulations of car traffic flow using smart signals (traffic lights and reverse lanes) in a more realistic way when

Table 4
Examples of PEDA.

$P\text{–}1_1$	$P\text{–}1_2$	$P\text{–}2_1$	$P\text{–}2_2$	Path	Length
Normal	5	0.5	0.5	In8-h-g-e-f-Q-d-c-b-O2	153
Normal	5	0.9	0.9	In8-h-g-e-P-O-I-N-d-c-b-O2	163
Uniform	0.5	0.5	0.5	In8-h-g-e-f-Q-d-c-b-O2	153
Uniform	0.5	0.9	0.9	In8-h-u-t-s-B-A-x-G-F-E-b-O2	154
Uniform	0.8	0.8	0.2	In8-h-g-e-P-Q-d-c-b-O2	153

Please cite this article in press as: J.L. Galán-García et al., A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city, Appl. Math. Comput. (2014), http://dx.doi.org/10.1016/j.amc.2014.11.076

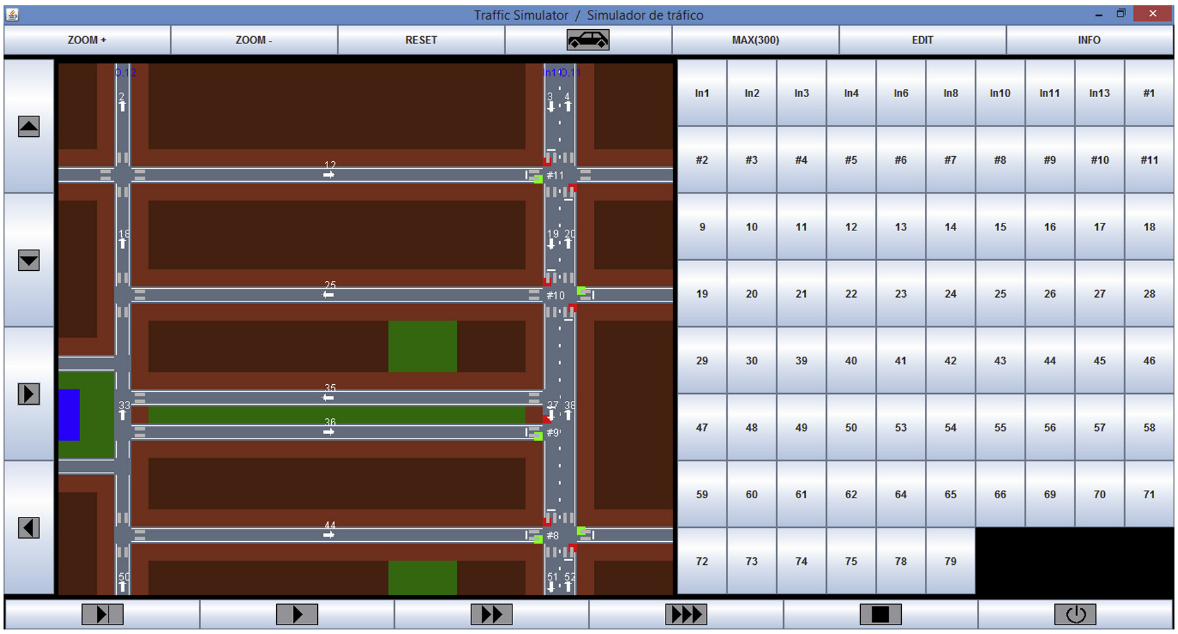


Fig. 3. Screenshot of the GUI of ATISMART+.

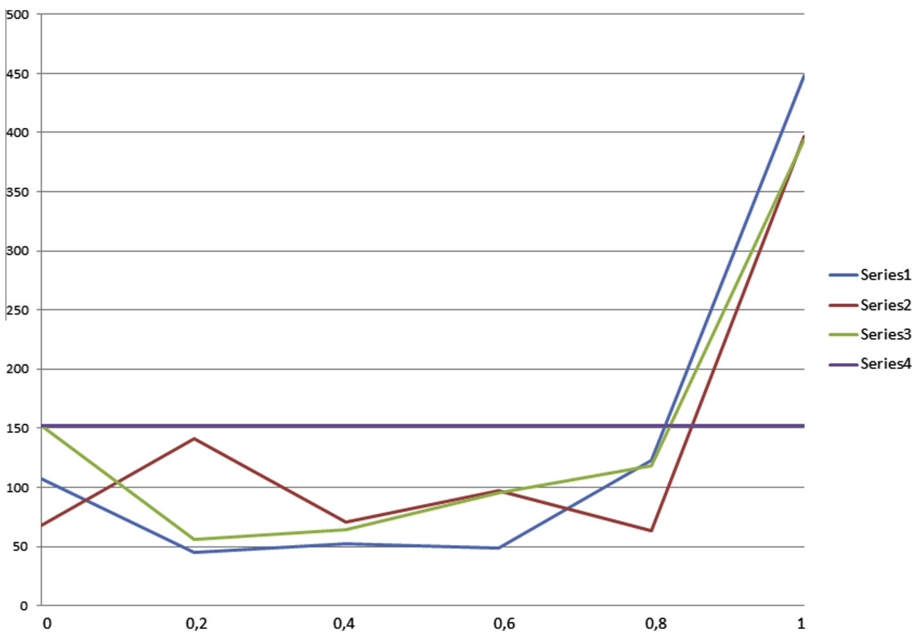
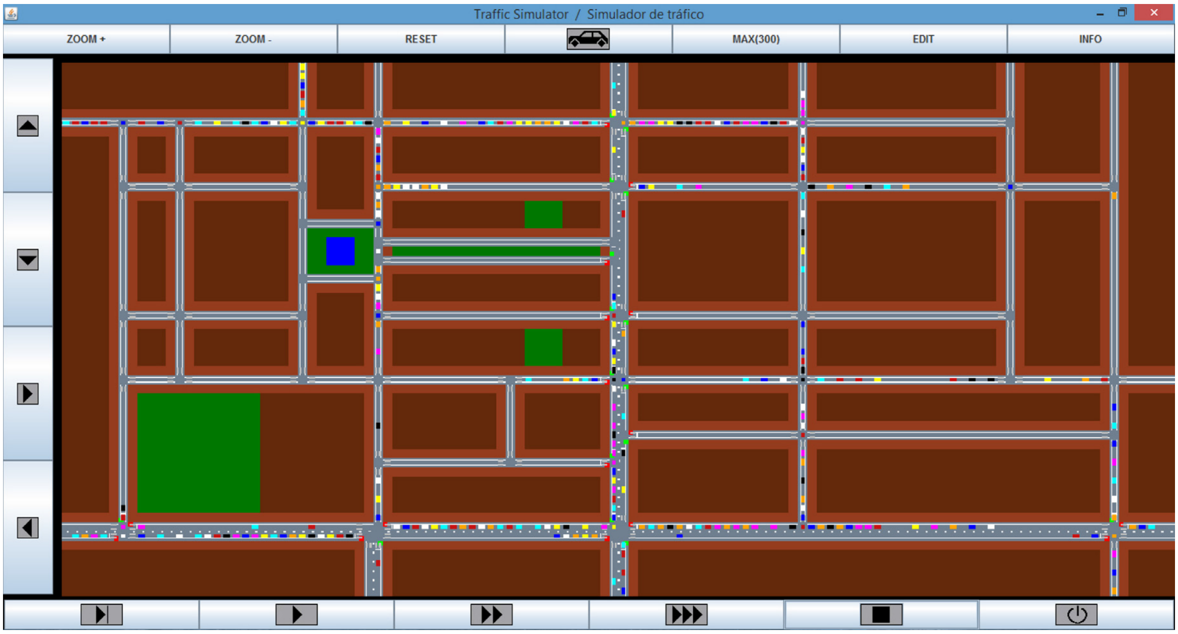


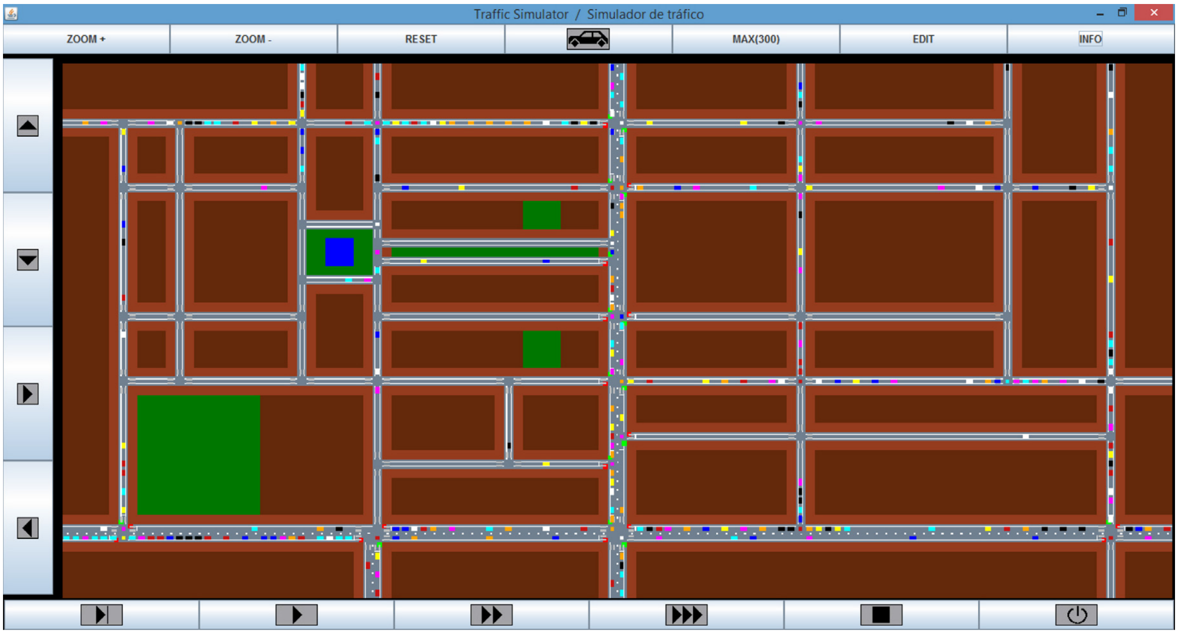
Fig. 4. Number of traffic jams with respect to p_v and p_h.

using PEDA. The graphical user interface (GUI) of ATISMART+ is an environment for both: introducing data and providing a visual images step by step of the generated simulation. The map that the GUI shows, corresponds to the real neighborhood of Málaga modeled by the graph in Fig. 1. A screenshot of the GUI can be seen in Fig. 3. In this figure, the editing window is displayed. This window allows dynamic changes in the parameters of the probability distribution functions of the inputs, the timing of traffic lights and the directions of lanes.

In order to see how PEDA algorithm acts in ATISMART+ some simulations have been carried out varying the parameters of PEDA. The results of these simulations are shown in Fig. 4.



Without using PEDA.



Using PEDA.

Fig. 5. Screenshots of ATISMART+ without and with PEDA.

The X-axis represents the variations of $p_v = p_h$ from 0 to 1 in steps of 0.2. In the Y-axis the number of traffic jams after 200 steps of the simulation are represented. The four different curves on the graphic correspond to the following series of simulations:

Series1 P–1_1 is the uniform distribution and P $- -1_2 = 0.8$.
Series2 P–1_1 is the uniform distribution and P $- -1_2 = 0.2$.

Series3 No distribution is used in P–1_1. That is, in this case, only P–2 acts.
Series4 Represents a constant value obtained with Dijkstra's algorithm (with no modifications) and it is plotted as a reference for the others curves.

Note that Series1 to Series3 remain under the fixed Series4 in all cases except when $p_v = p_h = 1$. In this case, the number of traffic jams is considerably higher than that obtained with Dijkstra's algorithm. According to the images provided by the GUI of ATISMART+ during the simulations, our interpretation of these two facts is as follow:

- We have observed that, when applying Dijkstra's algorithm, there exists some zones in the map with very low car traffic flow. This is due to the fact that this algorithm is deterministic and "force" all cars with the same input and output to follow the same path. However, when using the PEDA algorithm, since it is non-deterministic, cars with the same goals do not have to follow the same paths. Consequently, this diversification of paths produces fewer traffic jams.
- When $p_v = p_h = 1$ both algorithms, Dijkstra and PEDA, are deterministic but the former one provides optimal paths while the later one generates longer paths. Both facts, the determinism and longer paths generated by PEDA, results in cars staying for more time in the system using the same paths. Therefore, the number of traffic jams is greatly increased.

The GUI of ATISMART+ makes us see that, when Dijkstra's algorithm is used, most cars circulate only throughout a few lanes, leaving some zones of the map almost empty. This behavior is not the common situation in this part of Málaga. However, when using the PEDA algorithm, the distribution of the cars in the map is more uniform. As an example, Fig. 5 shows two screenshots of ATISMART+ without and with PEDA respectively, both for the same number of steps. In the upper picture, without using PEDA, it can be seen that there are some zones with no cars and others with many cars which cause many traffic jams. In the bottom picture, using PEDA, there are some cars in these isolated zones, which leads to a lower number of traffic jams.

Therefore, we can state that, the use of PEDA in ATISMART+ produces more realistic simulations of car traffic.

The source code of ATISMART+ model (both, the core of ATISMART+, developed in MAXIMA, and the GUI, developed in JAVA) can be freely downloaded at:

http://www.matap.uma.es/jlgalan/ATISMART+/.

5. Conclusions and ideas for future work

Some conclusions obtained in this work are:

- The new PEDA algorithm extending Dijkstra's algorithm in a probabilistic way, has been introduced.
- Each part of PEDA, P–1 and P–2, and the combination of both, has been shown to produce more alternatives in the generated paths than Dijkstra's algorithm. This fact is more realistic since different drivers moving from and to the same points, normally do not use the same path.
- The ATISMART+ model has been shown to be a flexible and easy-to-use tool to simulate traffic flow in a city using smart signals.
- Using PEDA in ATISMART+, produces a greater dispersion of cars within the considered map. This is also a more realistic situation.
- The use of the GUI has allowed the authors to visualize the behavior of cars in the simulations with and without using PEDA. This fact has turned out decisive for concluding that the use of PEDA in ATISMART+ produces more realistic simulations.

Some ideas for future work are:

- Adaptation of the ATISMART+ model for other accelerated-time simulations such as:
  - the design and improvement of a bus network in a smart city.
  - the design and improvement of bicycle lanes in a city.
- Adaptation of the ATISMART+ model when considering different types of vehicles such as: cars, emergency vehicles (ambulances, police cars, fire trucks), motorbikes, trucks, buses, trams and bicycles.
- Introduction of changes in the drivers' behaviors (change of target exit while driving in the system, different speeds, search for a place for parking, ...).
- Introduction of street repair work in some lanes of the system.

Acknowledgments

This work was partially supported by Grant No. TIN2011–28084 of the Science and Innovation Ministry of Spain, co-funded by the European Regional Development Fund (ERDF) and by Spanish project TIN2012–39353-C04–01.

Finally, we thank the anonymous reviewers for their suggestions and comments which have improved the quality of the paper.

References

[1] Levy et al. Evaluation of the public health impacts of traffic congestions: a health risk assessment. Environmental Health 9:65 (2010).
[2] S.F. Smith, G.J. Barlow, X.F. Xie, Z.B. Rubinstein, Smart urban signal networks: initial application of the SURTRAC adaptive traffic signal control system, in: Proceedings of the Twenty-Third International Conference on Automated Planning and Scheduling, Rome, Italy, 2013.
[3] José L. Galán-García, Gabriel Aguilera-Venegas, Pedro Rodríguez-Cielos, An accelerated-time simulation for traffic flow in a smart city, J. Comput. Appl. Math. 270 (2014) 557–563.
[4] E.W. Dijkstra, A note on two problems in connexion with graphs, Numer. Math. 1 (1959) 269–271.
[5] Yong Denga, Yuxin Chenb, Yajuan Zhanga, Sankaran Mahadevanc, Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment, Appl. Soft Comput. 12 (3) (2012) 1231–1237.
[6] Wang Shu-Xi, The improved dijkstra's shortest path algorithm and its application, Procedia Eng. 29 (2012) 1186–1190.
[7] Sathi Mukherjee, Dijkstras algorithm for solving the shortest path problem on networks under intuitionistic fuzzy environment, J. Math. Model. Algorithms 11 (4) (2012) 345359.
[8] José Luis Galán-García, Gabriel Aguilera-Venegas, Marí Á. Galán-García, Pedro Rodríguez. Simulating realistic traffic flow in a smart city (Abstract). in: Proceedings of the Fourth European Seminar on Computing ESCO 2014. Pilsen, Czech Republic, 2014, pp. 29.
[9] J.v. Neumann, The general and logical theory of automata, in: Jeffress L.A. (Ed.), Cerebral Mechanism in Behavior – The Hixon Symposium, John Willey & Sons, New York, 1951, pp. 1–31.
[10] W. Knospe, L. Santen, A. Schadschneider, M. Schreckenberg, Towards a realistic microscopic description of highway traffic, J. Phy. A: Math. Gen. 33 (48) (2000) 477–485.