An Evolutionary Algorithm to Generate Real Urban Traffic Flows

Daniel H. Stolfi and Enrique Alba

LCC, University of Malaga, Spain
{dhstolfi,eat}@lcc.uma.es

Abstract. In this article we present a strategy based on an evolutionary algorithm to calculate the real vehicle flows in cities according to data from sensors placed in the streets. We have worked with a map imported from OpenStreetMap into the SUMO traffic simulator so that the resulting scenarios can be used to perform different optimizations with the confidence of being able to work with a traffic distribution close to reality. We have compared the results of our algorithm to other competitors and achieved results that replicate the real traffic distribution with a precision higher than 90%.

Keywords: Evolutionary algorithm, traffic simulation, SUMO, smart mobility, smart city

1 Introduction

Road traffic related problems have been studied frequently in the last decades. Several published articles are based on traffic simulations and, if the city modeled is big enough, the need for not only real maps, but also real traffic distribution is obligatory in order to obtain valid conclusions.

Therefore, a traffic flow study is an important aspect to be taken into account when modeling scenarios based on real maps. These maps can later be used to optimize traffic light cycles [12], to study the placement of LED Panels throughout the city [14], to reroute vehicles with the aim of reducing travel times [13], and for many other uses in smart mobility [1, 5, 6, 9].

Usually, local councils publish data about traffic (and other useful information about the city), under a smart city initiative: the so called *Open Data*. Within this information we can find origin-destination matrices, main vehicle flows, or the number of vehicles at specific measurement points. When these data are available they can be used to analyze drivers' habits, predict traffic congestion, or model a real traffic distribution consisting of traffic flows that match the real values.

The problem we are solving here consists in calculating the best vehicle flows on a real map so that the number of vehicles counted in several measurement points are as close as possible to the data available from the local council. Our proposal is a new strategy based on an evolutionary algorithm which is able to calculate several traffic flows based just on a few measurement points, so that it can reproduce the real traffic, which in addition to a real map imported from OpenStreetMap, achieves instances for traffic study which are close to reality.

2 Related Work

There are many studies (see survey in [2]) which focus on the estimation of origin-destination matrix based on traffic counting locations. They can be static [7, 8] or dynamic [3, 10]. However, these algorithms assume that all link costs are available, which may not be true in practical situations such as our case study. Unfortunately, authors do not usually detail the scalability of their algorithms for larger networks, which is a key issue for the interest in their solutions.

In [15] a Hopfield Neural Network (HNN) model is used to estimate the urban origin-destination distribution matrix. The author claims that due to the ability of quick computation, parallel distributed processing and hardware realization of neural networks, it is possible to overcome the difficulties of mathematical optimization models. He finds the global optimal solution to the problem and experiments on a graph made of just five nodes representing the same number of zones. To the contrary, our method focuses on individual streets rather than zones (finer grain, higher realism) and we need to route vehicles via individual streets. Consequently, there are several routes available between measurement points in which the vehicles are counted. Therefore, we need to use a different technique to calculate the flows.

An open-source software, called TrafficModeler, is presented in [11]. This program implements a traffic definition model describing traffic via a set of traffic layers placed over a road network. By using those layers it is possible to represent specific traffic patterns associated with different attributes. Additionally, traffic flows can be obtained from virtual populations based on demographic data (i.e. transportation between home, school and work). This tool for modeling traffic flows differs from our proposal in that it cannot be applied when the only source of data is the number of vehicles measured by sensors.

Finally, there are two utilities included in the SUMO [4] software package called ACTIVITYGEN and DFROUTER. The former computes the mobility wishes for a group of citizens matching a map, while the latter uses values from induction loops (sensors) to compute vehicle routes. ACTIVITYGEN is quite similar in some aspects to the aforementioned article, analyzed in this section, although it does not provide a graphic user interface. DFROUTER is a tool that may be used in the same way as the work we present in this article, however, it assumes that the map is completely covered by sensors, especially on its borders, and it requires the exact timestamp in which vehicles were detected and their speed in all the measurement points. None of these options are suitable for the problem we are solving as they cannot be applied to calculate the traffic flows in the city based on just the number of vehicles counted by each sensor.

3 Problem Description

The aim of this article is to present a new strategy to generate traffic distributions in a city by using the data previously collected from sensors which count vehicles in a few streets. It is based on an evolutionary algorithm especially adapted to work with the difficulties that are present in this problem, such as high complexity due to the high number of vehicles and large scenarios, long evaluation times, and the high probability of traffic jams occurring in an actual city scenario when the number of vehicles moving through its streets increases.

Furthermore, as is the case in many cities of the world, we do not have access to the actual vehicle flows, as this information is often not known (or not published) by the city traffic authorities. However, our algorithm is capable of dealing with this drawback by modifying the number of vehicles in each flow, in order to match the number counted by the sensors at several points of the simulated city, increasing the realism of the simulation.

Let $\boldsymbol{v}^* = (v_1^*, \ldots, v_N^*)$ be a vector containing the values collected from N sensors in the real city, and $\boldsymbol{v} = (v_1, \ldots, v_N)$ a vector containing the values obtained from the evaluation of the city map. Our objective is to minimize the error $\boldsymbol{e_i} = |\boldsymbol{v}_i^* - \boldsymbol{v}_i|, i \in \{1, \ldots, N\}$ by modifying the vehicle flows $f = (f_1, \ldots, f_M)$ in the city.

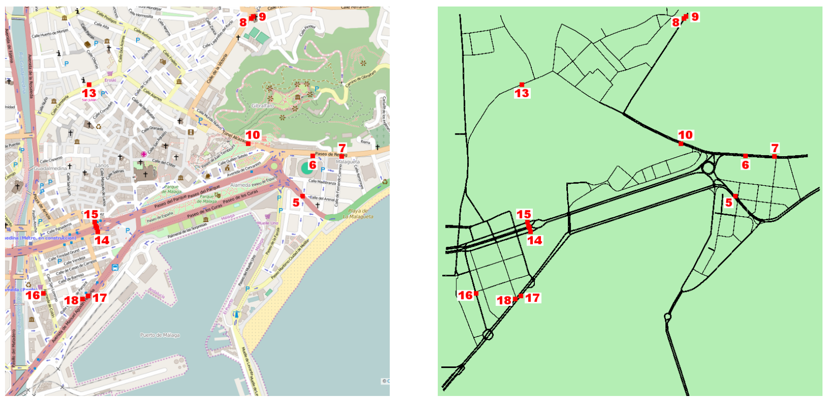
In short, by looking for appropriate flows (decision variables) we compute estimated flows on a simulator with the goal that they match real measured ones in the city where they are available. Of course, the set of flows contain a subset of proposed ones for the streets where no measurements are available at all, thus allowing the researcher to further study the city by using existing and approximated flows for all streets.

4 Case Study

In order to evaluate our proposal we have selected the downtown of Malaga, Spain, where several vehicle sensors have been installed by the local council. The geographical area under study is delimited to the north by San Bartolomé Street and Ferrándiz Street, to the west by the Guadalmedina River, to the east by Keromnes Street, and to the south by the Mediterranean Sea, which encompasses an area of about 3 km^2.

We imported the selected area from OpenStreetMap into SUMO [4] traffic simulator by using the program NETCONVERT included in the SUMO package so that we could work with a real road distribution consisting of the actual streets, traffic lights, roundabouts, and junctions. Then, we added the measurement points provided by the city council by using SUMO's induction loops (logical resource) in order to collect the number of vehicles in each of these points and compare them to the real ones.

In Fig. 1 the map from OpenStreetMap and a snapshot of the same map imported into SUMO are depicted. Note that the traffic sensors are located at the original positions in which the vehicles are counted.



(a) OpenStreetMap (b) SUMO

Fig. 1. Case study: Center of Malaga, Spain

Finally, we generated estimated flows between the input streets to the area, output streets, and local sources by using the DUAROUTE program included in SUMO to calculate routes by using the travel time as the weight function. All in all, we defined 63 flows, which determined the length of the status vector. Note that each flow consists in a sorted list of streets describing a route between the origin and destination points in the city

In all our results we analyzed the case study for one hour of traffic, as the values from the sensors correspond to this period of time. Additionally, we established a warm up period of 10 minutes so as not to start our analysis in an empty city, which is rather unrealistic.

5 Flow Generator Algorithm (FGA)

We propose here for the first time a Flow Generator Algorithm (FGA) to find the flows which minimize the differences between the vehicles measured in the city by the sensors and the values obtained after the simulation of the map.

Fig.2 shows the optimization process where we can see that FGA uses the available sensor data and the map from OpenStreetMap as inputs to evaluate individuals by using the SUMO traffic simulator. During the optimization process the algorithm generates and evaluates different individuals which represent the number of vehicles in each flow. Finally, when the optimization ends, the output produced is the number of vehicles in each flow plus the map of the city.

FGA is based on a (10+2)-EA, a steady state EA with a population of ten individuals generating two new individuals at each step. We used Binary Tournament as the selection operator, Uniform Crossover as the recombination operator, and an elitist replacement policy. We used local search instead of mutation after applying the recombination operator because it is necessary to provide extra information to the algorithm to avoid saturating the streets with a number

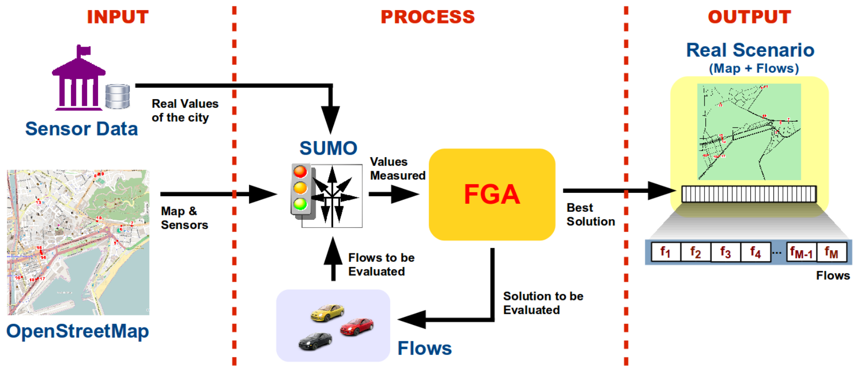


Fig. 2. Optimization Process.

of vehicles so large that they will always provoke jams (not enough capacity in the streets). The local search is described later in Section 5.3.

5.1 Problem Representation

In Fig.3 we can see the status vector containing flow values which are actually the number of vehicles that are following each flow. There are 63 flows ($M = 63$) in the case study we are optimizing, each one of the vector's values consists of an integer which can take values of between 10 and 500.

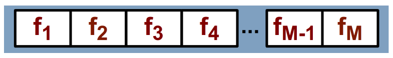


Fig. 3. Status vector containing the number of vehicles for the 63 flows.

5.2 Evaluation Function

Our evaluation function (Equation 1) assigns a numeric value to a configuration representing the vehicle flows (an individual of the FGA). We calculate the absolute value of the difference between the real values ($\boldsymbol{v}^{\star}$) measured in the city and the ones ($\boldsymbol{v}$) collected during the simulation of the map using the flows represented by the individual under evaluation. The fitness value of an individual is calculated by applying the evaluation function so that the numeric value of F is the summation of the absolute values of the differences for all the N sensors. This only happens if $C(\boldsymbol{v}) \leq 0.2$, otherwise, we apply a penalization of a large constant value in the algorithm (infinite) because we are minimizing, so the lower, the better.

$$F(\boldsymbol{v}) = \begin{cases} \sum_{i=1}^{N} \left|\frac{\boldsymbol{v}_i - \boldsymbol{v}_i^*}{\boldsymbol{v}_i^*}\right| & \text{if } C(\boldsymbol{v}) \leq 0.2, \\ \infty & \text{if } C(\boldsymbol{v}) > 0.2. \end{cases} \tag{1}$$

$$C(\boldsymbol{v}) = \max\left(\frac{\boldsymbol{v}_i - \boldsymbol{v}_i^*}{\boldsymbol{v}_i^*}\right), i \in \{1, \ldots, N\} \tag{2}$$

As we want to avoid traffic jams, we need to keep the number of vehicles in each street low while we target the real value as much as possible by adding vehicles to different flows. To do this, we set a limit of 20% ($C(\boldsymbol{v}) \leq 0.2$) as the maximum percentage each sensor can exceed the desired one (we have seen that bigger values lead to traffic jams which the algorithm cannot detect). In Equation 2 we present the calculation of $C(\boldsymbol{v})$ as the maximum difference between the $\boldsymbol{v}_i$ value and the $\boldsymbol{v}_i^*$ one, so that if one of the values obtained from the simulation is higher than 20% of the real one, the individual is invalidated. This threshold only affects high values while the lower ones are actually considered by the algorithm as we wish to match the desired number of vehicles by incrementing them progressively, preventing street congestions.

5.3 Operators

As we have said, we used Binary Tournament as the selection operator, Uniform Crossover as the recombination operator, and an elitist replacement policy. After applying the recombination operator we applied the local search operator whose pseudocode is shown in Algorithm 1.

Algorithm 1 Local Search

```
procedure LOCALSEARCH(individual,$\Delta(t)$)
    $v \leftarrow calculateSensorValues(individual)$
    $f \leftarrow getFlows(individual)$
    for all $f_i \in f$ do
        if $random() \leq P_{LS}$ then
            $sensors \leftarrow getRelatedSensors(f_i)$
            $s \leftarrow selectSensorRND(sensors)$
            if $v[s] - sensors[s] < 0$ then
                $incrementVehicles(f_i, \Delta(t))$
            else
                $decrementVehicles(f_i, \Delta(t))$
            end if
        end if
    end for
    return $individual$
end procedure
```

First, the algorithm obtains the number of vehicles measured in each sensor by simulating the scenario with the flow configuration provided by the individual and stores the values in the vector v so that it is able to decide whether the number of vehicles measured by each sensor is under or over the real value. Second, it obtains all the flows from the individual and stores them in the vector f to iterate over it. Third, it selects which flows are modified according to the P_{LS} parameter. Then, all the sensors which depend on the selected flow f_i are stored in the vector *sensors*, and just one of them is randomly selected and stored in s. This dependency data is available from the case study and depends on the positions of the sensors and the routes of the vehicles which do not change during the optimization process. If more vehicles are needed in the selected measurement point $v[s]$, the number of vehicles in the flow f_i is incremented by $\Delta(t)$, otherwise $v[s]$ is reduced by the same amount $\Delta(t)$. Note that we have set a minimum value of 10 vehicles and a maximum of 500 for each flow, the former, to be sure that all the flows are in use and the latter, as a value sufficiently higher than the maximum obtained in our initial experiments. Finally, after the loop, the local search ends and the individual is returned in order to be evaluated by the fitness function.

The value of $\Delta(t)$ is calculated based on the minimum fitness value of the population according to Equation 3 so that the lower the fitness value, the lower the variation the algorithm makes to the individual. This mechanism allows us to make big changes when the optimization begins (exploration) and lower ones when the values from the simulation are close to the real ones (exploitation). We have set α equal to 20 as this is the lower bound of $\Delta(t)$ and β equal to 3 in order to control the reduction pace of $\Delta(t)$ as the FGA converges.

$$\Delta(t) = \lceil \alpha * e^{(\min Fitness_i(t))/\beta} \rceil, i \in [1, \lambda], \lambda = 10 \tag{3}$$

In Fig. 4 we show and example of the local search performed on an individual. First, f_i is selected to be modified. Second, one of the sensors (s_j) whose value depends on the flow f_i is randomly selected. Finally, the number of vehicles in f_i is incremented or decremented according to the number of vehicles counted by s_j in order to be closer to the real value v_j.

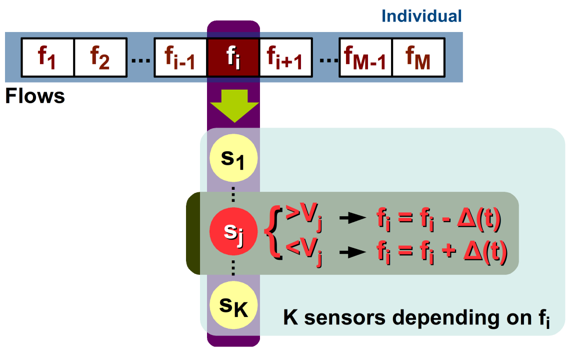


Fig. 4. Local search example.

We have experimentally found the values of the crossover probability ($P_C = 0.9$) and the local search probability ($P_{LS} = 1/L$). The maximum number of generations was set to 3000, however, if the best individual does not change for more than 500 generations, the FGA execution ends.

6 Competitors for our FGA

In order to evaluate our local search operator, we propose two different alternatives to configure the traffic flows in our case study: i) Random Search (RS) and ii) an Evolutionary Algorithm (EA).

6.1 Random Search (RS)

We report the behavior of a pure random algorithm to perform a sanity check on the validity of our FGA. We have implemented Random Search (RS) to obtain configurations of the traffic flows by generating different random configurations, saving the best of them until a better one is found. We have set a limit of 3000 iterations for the search process, the same number as in the FGA.

6.2 Evolutionary Algorithm (EA)

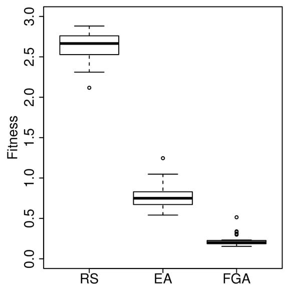
The Evolutionary Algorithm (EA) is exactly the same algorithm as the FGA but for the mutation operator that is used instead of the local search. This operator randomly selects flows according to the mutation probability (P_M) and modifies them with a constant number of vehicles (Γ). However, unlike FGA, the current value of the flow is incremented or decremented with a probability of 0.5, as this operator does not have further knowledge of the sensors affected by each flow. The main idea here is to support our choice of local search operator instead of the mutation one, rather than compete against a basic EA which is clearly unfair. Moreover, the parameters of the EA are the same as the FGA, that is, $P_C = 0.9$, $P_M = 1/L$, $\Gamma = 50$, 3000 generations, and 500 generations as convergence criterion.

7 Results

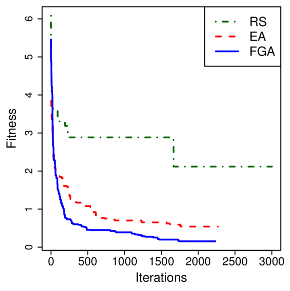
First, we have conducted the comparison between algorithms in order to analyze the performance of our proposal (FGA). We carried out 30 independent runs of each algorithm optimizing the same scenario and then collected and processed the results which are shown in Table 1. There, we present the average fitness value, the standard deviation, and the minimum (best) fitness solution. Additionally, we have calculated the Friedman Rank and the Wilcoxon *p-value* in order to know the statistical significance of our results.

Table 1. Comparison between algorithm.

Algorithm		Fitness		Friedman	Wilcoxon
	Average	StdDev	Minimum	Rank	*p-value*
RS	2.610	**7.01%**	2.117	3.00	0.00
EA	0.775	20.22%	0.541	2.00	0.00
FGA	**0.224**	32.53%	**0.154**	**1.00**	—



(a) Fitness distributions.



(b) Convergence curves.

Fig. 5. Comparison between the RS, EA, and FGA. We show in the box plot the fitness distribution of the 30 runs of each algorithm on the left, and the convergence curve of the best of those runs on the right.

We can see that FGA outperforms the rest of algorithms of this comparison as it achieves the lowest fitness (both absolute and on average). Moreover, FGA is the best ranked algorithm for this study, according to the Friedman Rank. Having calculated the Wilcoxon *p-value*, we can say that the values reported are statistically significant.

As was to be expected, RS produces the worst solution because it searches in the solution space without any information of the problem's characteristics. Additionally, the results achieved by EA indicate configurations of the flows far worse than the ones achieved by FGA, which better represents the real city traffic. This confirms that the local search operator proposed (FGA) performs much better than the ordinary mutation implemented in an EA as was expected.

Using the data from the 30 runs performed we have drawn a box plot, shown in Fig. 5(a), to represent the fitness distribution of the algorithms. This figure confirms the differences between the algorithms and how the fitness values of FGA are the lowest ones. Furthermore, in Fig. 5(b) we present the fitness evolution during the best run of each algorithm where the more appropriate behavior of the FGA is clearly depicted.

Second, we have tested the best solution achieved by the FGA by running the simulation of the city with the configuration of the flows according to that solution. In Table 2 we can see the names of the sensors, the same names as given by the local council of Malaga, the real number of vehicles (third four-month period of 2014), the number of vehicles measured in SUMO when we used the flow values calculated by the FGA, and the percentage difference.

We can observe that the differences are under 1% except for sensor 16 (9.96%). Moreover, the number of vehicles measured by sensors 7 and 8 in the simulation are exactly the same as in the real world. We have analyzed the anomaly observed in sensor 16 and found that the main source of vehicles for the street in which sensor 16 is placed (see Fig. 1), includes a left-turn intersection controlled by a traffic light. We have observed that many vehicles are waiting at this traffic light so that they cannot reach the sensor during the simulation. We think that a modification of the light cycle of that traffic light is

Table 2. Very accurate results found by FGA (best individual) for our case study.

Sensor	Vehicles		Difference
	Real	FGA	
5	1088	1078	-0.92%
6	349	351	0.57%
7	289	289	**0.00%**
8	265	265	**0.00%**
9	263	265	0.76%
10	653	648	-0.77%
13	228	230	0.88%
14	510	512	0.39%
15	663	658	-0.75%
16	522	470	-9.96%
17	850	852	0.24%
18	571	570	-0.18%

necessary in the simulated map to allow more vehicles to cross the junction and be detected. The analysis of previous optimization processes to be accomplished on the map before launching FGA is part of our future work. In this study we have used the traffic light cycles calculated by the experts from SUMO.

8 Conclusions and Future Work

In this article we have presented a strategy based on an evolutionary algorithm, called FGA, to add vehicle flows to maps imported from OpenStreetMap into the SUMO traffic simulator. As we use the real number of vehicles counted by sensors placed throughout the city, the resulting flows can be used to perform different types of optimizations with the confidence of being able to work with a traffic distribution close to reality.

We have tested our proposal against other competitor algorithms, obtaining results that show the best characteristics of our FGA. Furthermore, we have optimized the number of vehicles in measurement points and compared them against the real ones published by the mobility department of Malaga. Our results show that the number of vehicles in 11 out 12 sensors of the simulated map are under 1% the real value, and 0.8% on average if we consider all of them.

As a matter of future work we wish to study the previous optimizations process (traffic lights, routes, etc.) required before applying our solution, as well as extend the geographical area to be analyzed to include the entire city which represents not only more vehicles and analysis time, but also more sensors. It will be interesting to compare our FGA with other competitor algorithms which could be applied to our case study by using the available data (sensors) as inputs. However, at the moment they are hard to find.

Acknowledgments This research has been partially funded by project number 8.06/5.47.4142 in collaboration with the VSB-Technical University of Ostrava and Universidad de Málaga UMA/-FEDER FC14-TIC36, programa de fortalecimiento de las capacidades de I+D+i en las universidades 2014-2015, de la Consejería de Economía, Innovación, Ciencia y Empleo, cofinanciado por el fondo europeo de desarrollo regional (FEDER). Also, partially funded by the Spanish MINECO project

TIN2014-57341-R (`http://moveon.lcc.uma.es`). The authors would like to thank the FEDER of European Union for financial support via project "Movilidad Inteligente: Wi-Fi, Rutas y Contaminación" (maxCT) of the "Programa Operativo FEDER de Andalucía 2014-2020". We also thank all Agency of Public Works of Andalusia Regional Government staff and researchers for their dedication and professionalism. Daniel H. Stolfi is supported by a FPU grant (FPU13/00954) from the Spanish Ministry of Education, Culture and Sports. University of Malaga. International Campus of Excellence Andalucia TECH.

References

1. Angius, F., Reineri, M., Chiasserini, C., Gerla, M., Pau, G.: Towards a realistic optimization of urban traffic flows. In: Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on. pp. 1661–1668 (Sep 2012)
2. Bera, S., Rao, K.V.K.: Estimation of origin-destination matrix from traffic counts: The state of the art. European Transport - Trasporti Europei 49(49), 3–23 (2011)
3. Hazelton, M.L.: Statistical inference for time varying origin–destination matrices. Transportation Research Part B: Methodological 42(6), 542–552 (2008)
4. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent Development and Applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements 5(3), 128–138 (2012)
5. Krajzewicz, D., Wagner, P.: Large-Scale Vehicle Routing Scenarios Based on Pollulant Emissions. In: Meyer, G., Valldorf, J. (eds.) Advanced Microsystems for Automotive Applications, pp. 237–246. Springer Berlin Heidelberg (2011)
6. Kwatirayo, S., Almhana, J., Liu, Z.: Adaptive Traffic Light Control using VANET: A case study. In: Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International. pp. 752–757 (Jul 2013)
7. Li, B.: Bayesian inference for origin-destination matrices of transport networks using the EM algorithm. Technometrics 47(4) (2005)
8. Lo, H.P., Chan, C.P.: Simultaneous estimation of an origin–destination matrix and link choice proportions using traffic counts. Transportation Research Part A: Policy and Practice 37(9), 771–788 (2003)
9. Mckenney, D., White, T.: Distributed and Adaptive Traffic Signal Control Within a Realistic Traffic Simulation. Eng. Appl. Artif. Intell. 26(1), 574–583 (Jan 2013)
10. Nie, Y.M., Zhang, H.M.: A variational inequality formulation for inferring dynamic origin–destination travel demands. Transportation Research Part B: Methodological 42(7), 635–662 (2008)
11. Papaleondiou, L.G., Dikaiakos, M.D.: TrafficModeler: A Graphical Tool for Programming Microscopic Traffic Simulators through High-Level Abstractions. In: Vehicular Technology Conf., 2009. VTC Spring 2009. IEEE 69th. pp. 1–5 (Apr 2009)
12. Sánchez-Medina, J., Galán-Moreno, M., Rubio-Royo, E.: Traffic Signal Optimization in La Almozara District in Saragossa Under Congestion Conditions, Using Genetic Algorithms, Traffic Microsimulation, and Cluster Computing. Intelligent Transportation Systems, IEEE Transactions on 11(1), 132–141 (2010)
13. Stolfi, D.H., Alba, E.: Red Swarm: Reducing travel times in smart cities by using bio-inspired algorithms. Applied Soft Computing 24(0), 181–195 (Nov 2014)
14. Stolfi, D.H., Alba, E.: Smart Mobility Policies with Evolutionary Algorithms: The Adapting Info Panel Case. In: Proceedings of the 2015 Conference on Genetic and Evolutionary Computation Conference, GECCO'15. In press. Madrid (2015)
15. Zhejun Gong: Estimating the urban OD matrix: A neural network approach. European Journal of Operational Research 106(1), 108–115 (1998)