Red Swarm: Smart Mobility in Cities with EAs

Daniel H. Stolfi, Enrique Alba
LCC, University of Malaga
ETSI Informática, Campus de Teatinos
29071, Málaga, Spain
{dhstolfi,eat}@lcc.uma.es

ABSTRACT

This work presents an original approach to regulate traffic by using an on-line system controlled by an EA. Our proposal uses computational spots with WiFi connectivity located at traffic lights (the Red Swarm), which are used to suggest alternative individual routes to vehicles. An evolutionary algorithm is also proposed in order to find a configuration for the Red Swarm spots which reduces the travel time of the vehicles and also prevents traffic jams. We solve real scenarios in the city of Malaga (Spain), thus enriching the OpenStreetMap info by adding traffic lights, sensors, routes and vehicle flows. The result is then imported into the SUMO traffic simulator to be used as a method for calculating the fitness of solutions. Our results are competitive compared to the common solutions from experts in terms of travel and stop time, and also with respect to other similar proposals but with the added value of solving a real, big instance.

Categories and Subject Descriptors

I.6.3 [**Simulation and Modeling**]: Applications

Keywords

Application, evolutionary algorithm, road traffic, smart city, smart mobility, WiFi connections

1. INTRODUCTION

Nowadays, cities are evolving quickly. Most citizens are living in, using the services of, or thinking about moving to big cities, which is a new source of complex problems. Smart Cities is a world initiative leading to better exploit on of the resources in a city in order to offer higher level services to people. Smart cities are related to sensing the city's status and acting in new intelligent ways at different levels: people, government, cars, transport, communications, energy, buildings, neighborhoods, resource storage, etc.

One of the headlines of the strategy Europe 2020 is the reduction of greenhouse gas emissions by at least 20% compared to 1990 levels, an increment of the share of renewable energy sources in the final energy consumption to 20%, and a 20% increase in energy efficiency [1].

We believe that proposals like the one described here are directly related to the strategy Europe 2020 as well as to the smart cities initiative because we focus on the reduction of travel times, fuel consumption, and CO_2 emissions. We do so by reducing traffic jams so that the daily life of citizens in a smartcity improves.

To do this, we take advantage of the existent infrastructure in cities, such as traffic lights, open WiFi network, etc., in order to deploy several spots as a part of a collective system, called Red Swarm, each one placed in junctions with traffic lights and with the purpose of redirecting traffic based on each vehicle's destination. New routes are constructed with paths between Red Swarm spots, which relieve congested roads by distributing vehicles in alternative itineraries. Drivers are informed of the reroute proposed by using an On Board Unit (OBU) installed inside the vehicle which communicates with the infrastructure (V2I - car or smartphone to Red Swarm spot), by using the electronic panels placed around the city, or by using a portable device such as a smartphone or tablet.

Traffic optimization has always been a healthy line of research; now, even more so, with the utilization of ICT. There are solutions that focus on adjusting the traffic light cycles by using Particle Swarm Optimization (PSO) [4], or by using a Genetic Algorithm (GA) and a Cellular Automaton (CA) based traffic simulator [9], or by using optical information systems to calculate the size of the jam at the affected junction [7]. However, they do not modify the route of vehicles in advance, allowing drivers to make intelligent decisions as they move around the city.

In [3], bus lanes are opened when an additional traffic demand is recognized and in [5], a solution based on virtual police agents which are aware of the traffic conditions in their vicinity is presented. The former, only adds new lanes to existing roads and does not reroute vehicles while the later changes routes based on local information.

Recently, an approach which uses periodically emitted beacons to analyze the current traffic state via VANETs [6] was presented. It depends on V2V communications and the number of vehicles emitting beacons. Our proposal does not rely on the number of vehicles with a communication device to reroute them (off-line), although it is necessary to tell the driver of the suggested new route to follow.

The rest of this paper is organized as follows. In the next section we introduce the System Architecture proposed in-

.

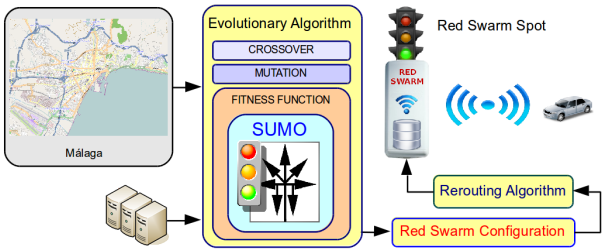


Figure 1: Component Schema of Red Swarm.

cluding the Evolutionary Algorithm and the steps taken to define its parameters and also the Rerouting Algorithm used in each Red Swarm spot. Section 3 presents the optimization results as well as the comparative between the experts' solution and the Red Swarm one, and finally in Section 4 conclusions and future work are given.

2. SYSTEM ARCHITECTURE

Our proposal, Red Swarm, is a computational system which consists of:

1. Several Red Swarm spots distributed throughout the city, installed at traffic lights which were chosen with the aim of redirecting the traffic efficiently.
2. The Evolutionary Algorithm which obtains the configuration of each spot.
3. The Rerouting Algorithm which suggests alternative routes based on the configuration of the system, so that drivers can take one of these routes when they enter the zone covered by a spot (WiFi)

In order to evaluate a configuration for the system we used the traffic simulator SUMO (Simulator of Urban Mobility) [2] in conjunction with TraCI (Traffic Control Interface) [10, 11]. SUMO is a microscopic traffic simulator which uses a microscopic model of discrete time and continuous space. Each moving vehicle implements the car-following model developed by Stefan Krauß in [8]. TraCI is an API which allows an external source (the Rerouting Algorithm in our case) to control SUMO as well as to obtain the state of the simulation and change it properly.

Figure 1 presents a schema with the components of Red Swarm. It shows the relation between the Evolutionary Algorithm, the map of the city, the simulator SUMO, and the configuration of the Red Swarm spots at design time. During execution time, the Rerouting Algorithm reads data from the configuration while it exchanges information with vehicles by using the Red Swarm spots. Each spot runs the same rerouting algorithm but only uses the configuration portion related to its own sensors.

2.1 Real Scenario

The scenario chosen is an area of the city of Malaga (in Spain) which is well-known for suffering traffic congestion at peak times, and it represents a typical challenge found in most European cities (non linear streets). The area is bordered to the north by Carretería Street, to the east by Gutemberg Street, to the west by the Guadalmedina River, and to the south by the Mediterranean Sea. In Figure 2

Figure 2: Malaga scenario for our smart mobility proposal.

we show the map of the zone chosen, obtained from the OpenStreetMap Project.

This zone contains eight inputs and eight outputs which represent real streets of the urban area. As we stated before, we used SUMO in order to simulate this vehicular environment by importing it from the project OpenStreetMap. This technique could be applied to any other middle-sized city in the world with little effort.

In order to implement the real model in SUMO, we had to divide each Red Swarm spot into a number of sensors which detect vehicles as they enter the area covered by the radio link of the spot. The real system could have sensors freely located or the detection could even be estimated off-line by the traffic municipal service.

Then, we defined all possible journeys between inputs and outputs from the zone in which vehicles follow the fastest path. As a result, we had 64 itineraries available. Furthermore, we defined four different kinds of vehicles (sedan, van, wagon, and transport), which are presented in Table 1 with the purpose of analyzing a truly general scenario.

Table 1: Type and characteristics of vehicles.

Type	Arriving prob.	MaxSpd. (Km/h)	Accel. (m/s^2)	Decel. (m/s^2)	Length (m)
sedan	0.50	160	0.9	5.0	3.8
van	0.25	100	0.8	4.5	4.2
wagon	0.15	50	0.7	4.0	4.3
transport	0.10	40	0.6	3.5	4.5

Finally, a problem instance was defined by different distributions of vehicle types, arriving time, arriving points and destinations (flows). In the experiments conducted we worked with three different instances. They were called *Sim1*, *Sim2*, and *Sim3*.

2.2 Evolutionary Algorithm

As this is a very complex problem, the technique developed for obtaining a configuration for the group of spots placed in the scenario was based on an Evolutionary Algorithm. By using it we were able to explore the solution space, looking to improve the current traffic flow in the city.

2.2.1 Representation

The structure defined, which contains the configuration of each input to a Red Swarm spot (sensors) and the destinations of the vehicles, is illustrated in Figure 3.

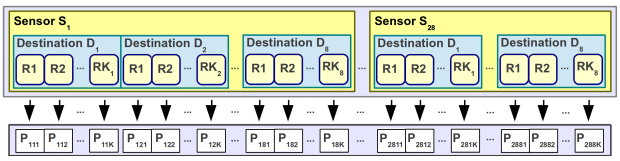


Figure 3: Problem representation.

If we look at the sensor S_1 for example, it contains all the possible destinations defined for the city (D_1 to D_8), and each destination contains all the probabilities for all the routes which start in S_1. For the routes of D_1 in S_1, there are RK_1 different paths, some of them to the rest of sensors and others to D_1 itself. Therefore, for the routes of D_2 in S_1, there are RK_2 different paths, and so on. Consequently, the solution vector contains all the probabilities associated with routes from sensors to other sensors and from sensors to destinations, arranged in blocks. In the scenario described, the solution vector is made up of 1119 probabilities values, which denotes the extreme complexity of this problem.

As presented in Equation 1, the result of the summation of the probabilities for the routes inside a destination block have to be equal to 1.0. Note that the subindex N identifies the sensor, the subindex M identifies the destination, and the subindex $R_i, i \in [1-K_{N,M}]$ identifies the different $K_{N,M}$ routes starting with the sensor S_N and belonging to the destination block D_M.

$$\begin{aligned} P_{S_N D_M} &= P_{N,M,R_1} + \cdots + P_{N,M,R_{K_{N,M}}} \\ &= \sum_{i=1}^{K_{N,M}} P_{N,M,R_i} = 1.0 \end{aligned} \quad (1)$$

2.2.2 Fitness Function

In order to evaluate each individual, we define the fitness function presented in Equation 2 which is done by the addition of three terms.

$$F = \alpha_1(N - n_{trips}) + \alpha_2 \frac{\sum t_{trip}}{N} + \alpha_3 \frac{\sum t_{delay}}{N} \quad (2)$$

The first one represents the penalization for the vehicles which are still in the city when the simulation is over. There, N is the total number of vehicles in the scenario (800 in this case), n_{trips} is the number of vehicles which complete their trip during the simulation, and α_1 is the weight of this term in the fitness function.

The second term represents the average travel time for the vehicles which complete the journey, being t_{trip} the travel time of each one. The third term is the average time spent by each vehicle waiting to enter the area because of red lights or traffic jams. This time is called t_{delay} in the equation.

Finally, α_2 and α_3 are the weights applied to the average travel time and to the average delay time, respectively. All values involved in the calculation of the fitness function are obtained from the results of the simulation run in SUMO when it is configured with the Red Swarm spots or with the experts' algorithm.

2.2.3 Recombination Operator

We propose two different recombination operators: Sensor Recombination and Destination Recombination. The first one consists of a two-point recombination which selects a range of sensors and their configurations from two individuals and exchanges them. As a consequence, the offspring obtained preserve part of their original sensor configuration and get a new configuration for only some of them, as depicted in Figure 4. Additionally, the initial position and the range of sensors are randomly chosen. Note that all the random selections made by the operators (recombination and mutation) are based on the uniform distribution implemented by the pseudo-random number generator included in the programming language, Python.

Figure 4: Sensor Crossover.

The second one, the Destination Crossover, preserves the configuration of the sensors along the routes that are being crossed, the exact opposite of what the other operator does. This behavior keeps the routes of each destination block unaltered after the crossing operation. Therefore, the offspring will be composed by complete routes obtained from both parents after the recombination.

2.2.4 Mutation Operators

We also propose several mutation operators. Each operator makes changes to the probabilities of the routes for sensors and destinations so that vehicles would take different routes when the Red Swarm spots are configured with different individuals. In this approach, we assigned probabilities equal to either 0 or 1, keeping only one route active in a destination block.

The ADOS operator (All Destinations - One Sensor) modifies the probabilities for all routes and all destinations in one sensor. Note that the sensor is randomly chosen, as depicted in Figure 6(a), where all routes for all destinations in S_2 are modified (shown in dark). This operator makes a number of changes to the status vector but it only affects one sensor.

The MDOS operator (Multiple Destination - One Sensor) changes all the routes for a few destination blocks in one sensor. The sensor and the destination blocks are randomly selected. As Figure 6(b) shows, this operator on average modifies fewer routes than the previous one but more than the next one. In this example, routes of blocks D_2 and D_8 in S_2 are modified. The destination blocks whose routes are changed are shown in a dark color.

The next operator, called ODOS (One Destination - One Sensor), makes the least number of changes of the status vector with respect to ADOS and MDOS. It selects only a random sensor and a random destination block and changes the routes associated with them. In Figure 6(c), we can observe that S_2 and D_2 have been selected as the target of the mutation operator.

The last two operators, ASMD (All Sensors - Multiple Destinations) and ASOD (All Sensors - One Destination),

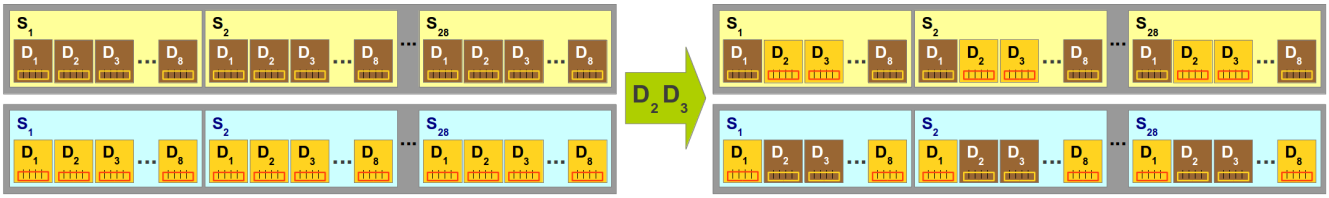
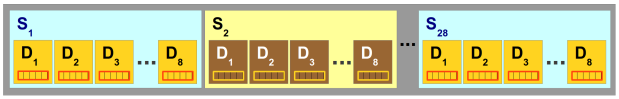
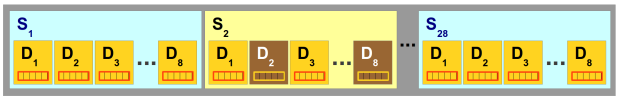


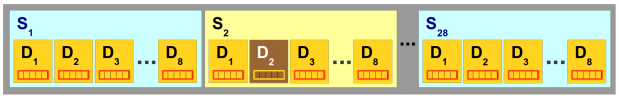
Figure 5: Destination Crossover.



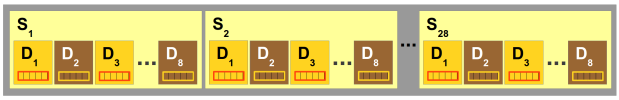
(a) ADOS: All Destinations - One Sensor



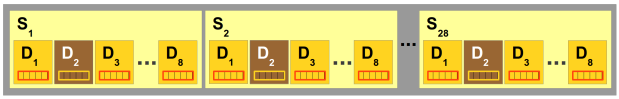
(b) MDOS: Multiple Destination - One Sensor



(c) ODOS: One Destination - One Sensor



(d) ASMD: All Sensors - Multiple Destinations



(e) ASOD: All Sensors - One Destination

Figure 6: Mutation Operators.

make changes to each flow individually. In the example presented in Figure 6(d) we can see how the routes in D_2 and D_8 are modified in all sensors, while in Figure 6(e), only D_2 is modified in all the sensors. Note that the selection of the destination blocks is made according to a uniform distribution of probability as in the rest of the operators.

2.2.5 *Parameterization*

First of all we need to evaluate the instances to know how long it takes for vehicles to complete their itineraries, and obtain a fitness value for each one in order to compare them with the solutions that we later calculate.

We worked with three different simulations called *EXP1*, *EXP2*, and *EXP3*, corresponding with the simulations in which routes are defined by using the experts' algorithm, run in the scenarios *sim1*, *sim2* and *sim3* respectively.

As stated before, we assigned 100 vehicles to each one of the eight flows to obtain the values presented in Table 2. Vehicles arrived in the scenario in the first 100 seconds and after 1500 seconds all the vehicles had left the city, as depicted in Figure 7. Based on this value, we set the maximum simulation time to 1800 seconds (30 minutes), in order not

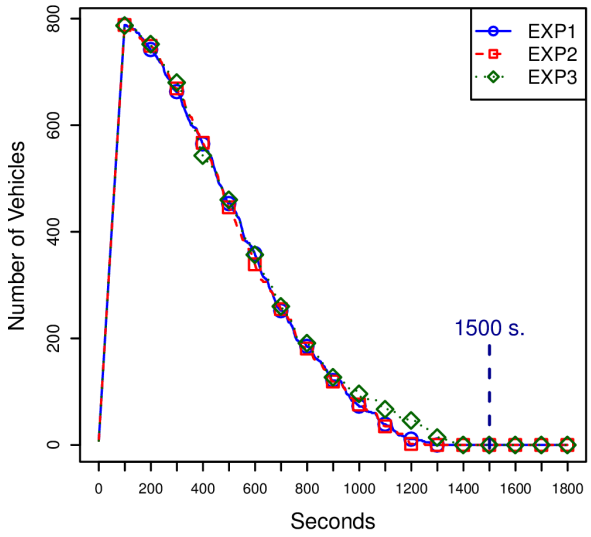


Figure 7: Vehicles in the city during the first 1800 seconds.

to excessively penalize solutions which last for more than the longest time spent in the EXPx simulations, but which could have a lower than average travel time.

Table 2: Results produced by the experts' algorithm.

Metric	EXP1	EXP2	EXP3
# Trips Completed	800	800	800
Simulation Time (s)	1283	1253	1385
Minimum Fitness	532.9	530.8	549.0

The parameters for the Evolutionary Algorithm implemented are presented in Table 3. We worked with a population of ten individuals and in each new generation, two new descendants were created (crossover probability equal to 1.00), in order to implement a (10+2)-EA algorithm. Then, three in four descendants were mutated (mutation probability equal to 0.75) to provide genetic variety and finally, the new generation was updated following an elitist strategy. The algorithm was executed for 5000 generations to achieve each solution.

Additionally, α_1 was set to 900 (half of the simulation time), and α_2 and α_3 were set to one. By using these values, we penalized vehicles which did not finish their itinerary (α_1) and therefore were still stuck on their journey through the city when the simulation ended. Moreover, we assigned the

Table 3: Parameters of the Evolutionary Algorithm.

Simulation Time (s)	1800
Population Size	10
Offspring Size	2
Crossover Probability	1.00
Mutation Probability	0.75
Number of Generations	5000
(α_1, α_2, α_3)	(900, 1, 1)

same weight to the travel time (α_2) and to the delay time (α_3) because they are equally important for us.

Once the algorithm had been configured, we evaluated the proposed operators. The results are reported in Table 4.

Table 4: Evaluation of operators.

Operators	Avg.	StdDev.	Min.	Conver.
Sensor: ADOS	3557.9	11403.5	555.8	93.8%
Sensor: MDOS	5057.4	9660.2	551.1	93.8%
Sensor: ODOS	6732.9	16455.4	548.4	81.3%
Sensor: ADOS-ODOS	1066.2	2901.1	**512.7**	94.6%
Sensor: ADOS-MDOS	1329.6	2908.2	516.0	**100.0%**
Dest.: ADOS	2347.5	6744.8	548.0	**100.0%**
Dest.: ODOS	**803.5**	**700.8**	555.9	**100.0%**
Dest.: ASMD	36199.5	18174.6	13309.0	87.5%
Dest.: ASOD	5273.9	16606.8	576.2	**100.0%**
Dest.: ADOS-ODOS	906.9	1758.6	518.8	90.0%
Dest.: ASMD-ASOD	5101.2	11110.6	591.9	43.3%

Note that we also tested combined mutation operators within the algorithm because they allowed us to get better results by using a faster mutation operator at the beginning of the execution, and then switching to a slower one when the fitness of the best individual is lower than a threshold. The value of the threshold was empirically set to 1000.

Based on the data in Table 4 we selected the Sensor Crossover as the recombination operator, while for the mutation operators we decided to start with ADOS and later change to ODOS. This combination was selected because it produced the best minimum value in spite of the fact that the average and the convergence percentage (Conver.) were not the best of the comparative.

Using the first mutation operator (ADOS) we quickly explore the data space. Then, when all the vehicles have finished their itineraries within the simulation time (which makes the best fitness of the population fall below the threshold value), we switch to the second one (ODOS).

By using the ODOS mutation, the search space is carefully explored by mutating only the probability of the routes included in just one destination block in one sensor. This only affects a small proportion of the Red Swarm spots, allowing the improvement of the solution in small steps.

2.3 Rerouting Algorithm

When a vehicle enters the simulation, SUMO assigns one of the eight destinations available as well as the corresponding route to it. Later, when the vehicle enters the coverage area of the Red Swarm spot (in the simulation it is detected by a sensor), the Rerouting Algorithm presented in Algorithm 1, is executed.

Algorithm 1 Rerouting Algorithm.

```
procedure REROUTING(vehicle)
    currentRoad ← getRoad(vehicle)
    if isDestination(currentRoad) then
        newroute ← currentRoad
    else
        routes ← getDestinationRoutes(currentRoad)
        if routes = [] then
            routes ← getSensorRoutes(currentRoad)
            newroute ← getRouteByProbability(routes)
        end if
    end if
    setNewRoute(newroute, vehicle)
end procedure
```

First, the current road is obtained from the vehicle and the destination is checked because if the vehicle is on the last road of its itinerary, no rerouting is done. Otherwise, all the routes from the current road to the destination of the vehicle are obtained from the configuration.

Then, if there are no routes to the destination (it is not reachable from the current position), the algorithm will obtain all routes from the current road to the rest of sensors.

Next, one of the routes from the current road to another road containing a sensor is chosen taking into account the probability stored in the configuration.

Finally, the route of the vehicle is changed to the new one and the process ends.

2.4 Experimental Settings

To optimize each traffic distribution we carried out 35 independent runs on the cluster belonging to the NEO (Networking and Emerging Optimization) group, consisting of 16 nodes, each equipped with an Intel Core2 Quad CPU (Q9400) @ 2.66GHz and 3.5 GB of RAM, running GNU/-Linux 3.2.0-34. By using this hardware, each new generation took about 15 seconds to be calculated, and each execution of the algorithm lasted about 22 hours, having spent about 9 days of computation on the whole optimization process.

3. ANALYSIS OF RESULTS

Table 5 presents the number of trips completed (# Trips Comp.), the total simulation time (Total Time), and the minimum fitness value (Min. Fitness) for the scenarios which are routed according to experts' criteria implemented in SUMO. These results were achieved by optimizing the three scenarios separately.

Table 5: Comparison between the experts' algorithm and Red Swarm.

Metric	Experts' algorithm			Red Swarm		
	Sim1	*Sim2*	*Sim3*	*Sim1*	*Sim2*	*Sim3*
# Trips Comp.	800	800	800	800	800	800
Total Time (s)	1283	1253	1385	**1154**	**1201**	**1194**
Min. Fitness	532.9	530.8	549.0	**512.7**	**530.2**	**513.0**

With this data, we conducted the robustness test by applying the three best available solutions (belonging to the three traffic distributions) to the other simulations in order to discover which one was the most robust. That is, *Sim1*

was simulated with the configuration of the solutions *sol2* and *sol3*; *Sim2*, with the configuration of *sol1* and *sol3*; and finally, *Sim3*, with *sol1* and *sol2*.

Table 6 demonstrates the improvement of using *sol1* to configure the second simulation instead of using *sol2*. Although *sol3* is a better configuration than *sol1* for *Sim3*, the average value achieved by *sol1* is the best of the three. These results encouraged us to select *sol1* as the most robust solution to configure the Red Swarm spots.

Table 6: Minimum fitnesses obtained from the robustness test.

	Red Swarm			Expert
	sol1	*sol2*	*sol3*	
Sim1	**512.7**	552.2	546.9	532.9
Sim2	**523.5**	530.1	555.4	530.8
Sim3	533.3	546.4	**513.0**	549.0
Average	**523.2**	542.9	538.4	537.6

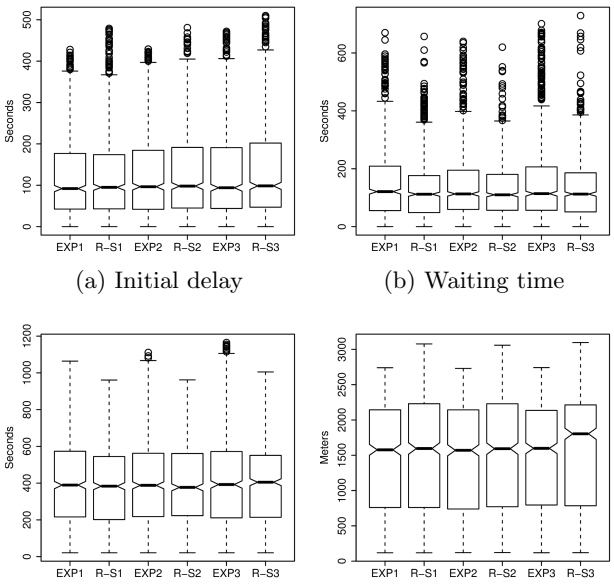
In tables 7 and 8 we present the best results achieved with the aim of optimizing the traffic in the city center of Malaga. The average values of waiting time and travel time for all Red Swarm simulations are lower than the experts' solution. There are values such us the initial delay and route length which were not improved by our Red Swarm system, especially the latter as it is not included in the fitness function. Initial delays are higher (avg. 2.3%) mainly due to the traffic lights situated near the input streets where the vehicles arriving still need to be rerouted by a Red Swarm spot. Route lengths are also longer (avg. 5.3%) because we were rerouting vehicles via roads which are not part of the best path selected by the experts.

However, we achieved a maximum improvement of 20.2% in the average waiting time and 5.3% in the average travel time. So, it seems that experts just take into consideration the traveled distance, while the time is much better if we want to consider the comfort of reduced waiting at traffic lights and at traffic jams, and the reduced gas emissions thanks to this (continuous driving).

The distribution of the results is demonstrated in Figure 8 where the improvement achieved is presented in four box plots. Note that waiting times and travel times present lower values for R-S than the experts' solution.

Furthermore, we compare the total time needed by the vehicles to leave the city when we increment the number of arrivals in each input, in steps of 80 vehicles (ten per input). Figure 9 indicates how the Red Swarm system is actually an effective solution when the number of vehicles is higher than a certain value. So, when the number of vehicles in the area being studied is above 360 (400, 540) the utilization of our Red Swarm is profitable for the city. This number of cars is of course realistic for this part of the city (many more in general), thus explaining why the town hall is currently interested in a prototype.

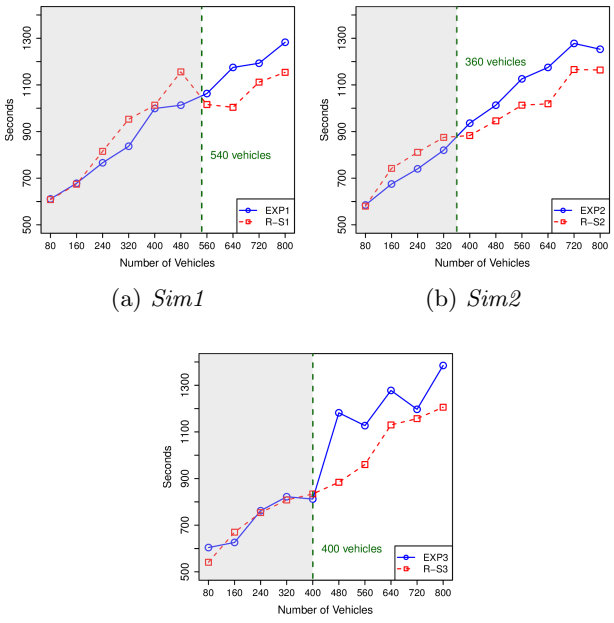
Finally, we compare our proposal with the best configuration obtained from the execution against the RANDOM algorithm, which randomly looks for solutions during a time equivalent to that spent by the Evolutionary Algorithm we used. The results obtained by using a random search show a highly negative impact on the quality of solutions: cars are not advised of useful routes to their destinations. Many vehicles even remain in the system at the end of the period studied (63 vehicles for *Sim1*, 90 for *Sim2*, and 152 for *Sim3*). The reader might be thinking why we are reporting the behavior of a pure random algorithm: the reason is to perform a sanity check on the validity of our EA. It is



(a) Initial delay (b) Waiting time

(c) Travel time (d) Route length

Figure 8: Box plot representation of the distribution of the results for the experts' solution (EXP) and the Red Swarm (R-S).



(a) *Sim1* (b) *Sim2*

(c) *Sim3*

Figure 9: Exit Time vs Number of Vehicles. Note that the vertical line indicates the lowest number of vehicles, from which the optimization is effective.

Table 7: Metrics of experts' solution, Red Swarm, and RANDOM.

Metric	Experts' solution			Red Swarm			RANDOM		
	Sim1	*Sim2*	*Sim3*	*Sim1*	*Sim2*	*Sim3*	*Sim1*	*Sim2*	*Sim3*
# Trips Completed	800	800	800	800	800	800	737	710	648
Total Time (s)	1283	1253	1385	**1154**	**1164**	**1206**	1800	1800	1800
Minimum Fitness	532.9	530.8	549.0	**512.7**	**523.5**	**533.3**	57459.5	81797.6	137547.3

Table 8: Results of experts' solution and Red Swarm.

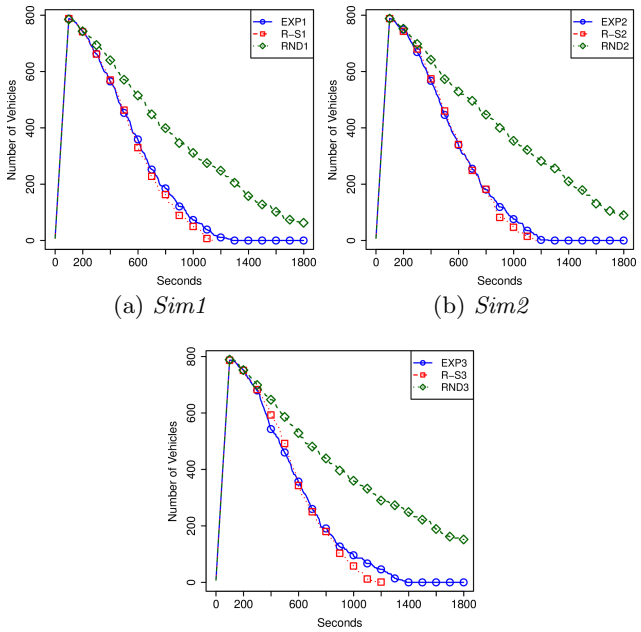
	Metric	Experts' algorithm			Red Swarm			Rate
		Avg.	StdDev.	Median	Avg.	StdDev.	Median	(Avg.)
sim1	Initial delay (s)	**123.0**	103.1	92.0	123.4	104.1	95.0	0.3%
	Waiting time (s)	151.8	130.3	121.0	**128.6**	106.4	112.0	-18.0%
	Travel time (s)	409.9	237.8	389.5	**389.2**	214.2	383.5	-5.3%
	Route length (m)	**1491.4**	767.6	1577.1	1565.4	853.6	1595.5	4.7%
sim2	Initial delay (s)	**124.7**	102.5	96.5	127.8	104.8	98.0	2.4%
	Waiting time (s)	146.3	127.4	113.0	**129.1**	98.1	110.0	-13.3%
	Travel time (s)	406.1	234.1	388.0	**395.7**	213.8	377.0	-2.6%
	Route length (m)	**1477.7**	777.9	1570.2	1566.3	845.9	1593.0	5.7%
sim3	Initial delay (s)	**127.0**	107.3	94.0	132.7	111.2	98.5	4.3%
	Waiting time (s)	161.2	153.3	114.0	**134.1**	110.9	112.5	-20.2%
	Travel time (s)	422.0	260.3	392.5	**400.6**	213.7	405.0	-5.3%
	Route length (m)	**1506.9**	753.9	1598.0	1594.8	834.3	1803.9	5.5%

quite common for researchers to not carry out such a sanity check, reporting results of algorithms that can be beaten by random search. We simply do not want this to happen here, and that is why we include its numerical results.

Data from the RANDOM executions are presented in Table 7. While in Figure 10 the graphs of the number of vehicles in the city vs. the total time for the three executions are illustrated.

The main reason for such a poor performance of RANDOM is that there were a number of vehicles moving in circles around the city because the Red Swarm spots did not reroute them properly towards their destination due to a suboptimal configuration.

In Figure 11 we show two photorealistic snapshots exported from SUMO to Google Earth™. The first one shows the path followed by a vehicle routed by the experts' algorithm and the second one presents the new route set by the Red Swarm spot placed at the junction in the center of the figure. This behavior is repeated each time a vehicle arrives in the surrounding area of a Red Swarm spot and the new route will depend on the destination of each vehicle as well as the configuration of the spot.



(a) *Sim1* (b) *Sim2*

(c) *Sim3*

Figure 10: Traffic Density for experts' algorithm (EXP), RANDOM (RND), and Red Swarm (R-S).

4. CONCLUSIONS

In this paper, we have presented an innovative approach for preventing traffic jams. The results confirm that road traffic can be improved by using our proposal, especially in high density conditions (e.g. peak times, construction, accidents reducing the available streets, etc.). Although the average travel time was only 5.3% better than the solution of the experts, we lowered the waiting time of the vehicles by 20.2%. That means that drivers are wasting less time (about 23 second less on average) waiting at a red light and in each traffic jam.

As a matter for future work, we plan to expand the size of the analyzed region as well as include a larger number of

Figure 11: Route computed by experts' algorithm vs. route computed using Red Swarm.

Red Swarm spots which will also provide more alternative routes between each other.

For preliminary results for our future work, we have tested Red Swarm in 30 extra scenarios, different from the ones analyzed here. Our proposal not only has worked in all these scenarios but has also outperformed the experts' solution in 20 of them (66.7%).

Red Swarm will not only provide alternative routes for every single car in the city, but will also be able to collect information from the cars (in an anonymous way) permitting local authorities to better know the on-line and historical data of the city to actually help the evolution to a modern smart city.

We are also working on reducing green house emissions. As road traffic is one of the main sources of pollution, in future work we will include metrics of pollutant emissions and fuel consumption as SUMO implements them.

5. ACKNOWLEDGMENTS

Authors acknowledge funds from the Ministry of Economy and Competitiveness and FEDER under contract TIN2011-28194 (roadME `http://roadme.lcc.uma.es`).

6. REFERENCES

[1] Communication From the Commission - Europe 2020. Technical report, 2010. A strategy for smart, sustainable and inclusive growth.

[2] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz. SUMO - Simulation of Urban MObility: An Overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, October 2011.

[3] L. Bieker and D. Krajzewicz. Evaluation of opening bus lanes for private traffic triggered via V2X communication. In *Integrated and Sustainable Transportation System (FISTS), 2011 IEEE Forum on*, pages 48–53. IEEE, 2011.

[4] J. Garcia-Nieto, E. Alba, and A. Olivera. Enhancing the urban road traffic with Swarm Intelligence: A case study of Córdoba city downtown. In *Intelligent Systems Design and Applications (ISDA), 2011 11th International Conference on*, pages 368–373. IEEE, 2011.

[5] A. Gomez, G. Diaz, and K. Boussetta. How Virtual Police Agents can help in the traffic guidance? In *Wireless Communications and Networking Conference Workshops (WCNCW), 2012 IEEE*, pages 360–364. IEEE, 2012.

[6] F. Knorr, D. Baselt, M. Schreckenberg, and M. Mauve. Reducing Traffic Jams via VANETs. *Vehicular Technology, IEEE Transactions on*, 61(8):3490–3498, oct. 2012.

[7] D. Krajzewicz, E. Brockfeld, J. Mikat, J. Ringel, C. Rössel, W. Tuchscheerer, P. Wagner, and R. Wösler. Simulation of modern traffic lights control systems using the open source traffic simulation SUMO. In *Proceedings of the 3rd industrial simulation conference*, pages 299–302, 2005.

[8] S. Krauß. *Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics*. PhD thesis, Universitat zu Koln., 1998.

[9] J. Sánchez, M. Galán, and E. Rubio. Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife. *Evolutionary Computation, IEEE Transactions on*, 12(1):25–40, 2008.

[10] A. Wegener, H. Hellbrück, C. Wewetzer, and A. Lubke. VANET Simulation Environment with Feedback Loop and its Application to Traffic Light Assistance. In *GLOBECOM Workshops, 2008 IEEE*, pages 1–7. IEEE, 2008.

[11] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J. Hubaux. TraCI: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, pages 155–163. ACM, 2008.