Optimal Cycle Program of Traffic Lights With Particle Swarm Optimization

José García-Nieto, Ana Carolina Olivera, and Enrique Alba

Abstract—**Optimal staging of traffic lights, and in particular optimal light cycle programs, is a crucial task in present day cities with potential benefits in terms of energy consumption, traffic flow management, pedestrian safety, and environmental issues. Nevertheless, very few publications in the current literature tackle this problem by means of automatic intelligent systems, and, when they do, they focus on limited areas with elementary traffic light schedules. In this paper, we propose an optimization approach in which a particle swarm optimizer (PSO) is able to find successful traffic light cycle programs. The solutions obtained are simulated with simulator of urban mobility, a well-known microscopic traffic simulator. For this study, we have tested two large and heterogeneous metropolitan areas with hundreds of traffic lights located in the cities of Bahía Blanca in Argentina (American style) and Málaga in Spain (European style). Our algorithm is shown to obtain efficient traffic light cycle programs for both kinds of cities. In comparison with expertly predefined cycle programs (close to real ones), our PSO achieved quantitative improvements for the two main objectives: 1) the number of vehicles that reach their destination and 2) the overall journey time.**

Index Terms—**Particle swarm optimization, programming cycles of traffic lights, simulator of urban mobility (SUMO).**

I. Introduction

POLLUTION, congestion, security, parking, and many other problems derived from vehicular traffic are present every day in most cities around the world. Since changes in urban area infrastructure are usually not possible, researchers often agree that a correct staging of traffic lights can help to reduce these problems by improving the flow of vehicles through the cities [1]–[3]. Nevertheless, as traffic lights are installed in cities and their number grows, their joint programming becomes more complex due to the huge number of combinations that appear, and hence, the necessity of implementing automatic systems to optimally program the cycles of traffic lights is beyond doubt.

This work was supported in part by the Spanish Ministry of Economy and Competitiveness (MEC) and FEDER under Contract TIN2011-28194 (roadME project). The work of J. García-Nieto was supported by the MEC under Grant BES-2009-018767. The work of A. C. Olivera was supported by the Consejo Nacional de Investigaciones Científicas y Técnicas of Argentina and the ANPCyT under Contract PICT-2011-0639.

J. García-Nieto and E. Alba are with the Department Lenguajes y Ciencias de la Computación, University of Málaga, Málaga 29071, Spain (e-mail: jnieto@lcc.uma.es; eat@lcc.uma.es).

A. C. Olivera is with the Departamento de Ciencias e Ingeniería de la Computación, Universidad Nacional del Sur, Bahía Blanca 8000, Argentina (e-mail: aco@cs.uns.edu.ar).

In this sense, current research efforts in the field of automatic traffic control signals are directed to two main initiatives. On the one hand, automatic models of adaptation of signal control are designed [4]–[6] to change cycle program duration throughout the day as vehicles in queues demand these changes. The operation of these kinds of tools is directly related to the sensor system and real-time computation of the traffic flow. Although these tools successfully perform in several cities around the world [4], [7], the real management of the traffic network has a high operational cost and the real world generally tends to repeat traffic flow patterns (rush hour, holidays, etc.).

On the other hand, modern simulators [8]–[10] are very useful for helping in traffic management, since they provide researchers with an immediate and continuous source of information about traffic flow. In addition, economical issues are also taken into account in this kind of research, since the use of real traffic tests implies the necessity of additional staff and sensoring platforms. Many studies in traffic flow simulation have been performed representing both macroscopic [1] and microscopic [2], [11] traffic views. Over the past few years, efforts have concentrated on combining an accurate microscopic modeling of traffic flow [2], [9] and the programming of suitable traffic light cycles [12]. Accordingly, the use of intelligent methods has proven to be useful for the optimization of programming traffic light cycles [2], [13]. However, in general, authors have addressed specific urban areas with few intersections and a small number of traffic lights (from 1 to 4 intersections with around two traffic lights controlling each intersection) [14], and most of them consist of *ad hoc* algorithms designed for only one specific instance [2], [13]. The use of intelligent techniques for large and heterogeneous study cases is still an open issue [15]. It is a complex problem since the greater the number of adjacent intersections, the greater the interaction between the traffic lights (which increases the complexity of the problem by introducing a high epistasis between variables).

All this has motivated us to propose a technique based on a particle swarm optimizer (PSO) [16]–[18] that will be shown to find successful traffic light cycle programs coupled with simulator of urban mobility (SUMO) [19], a well-known microscopic traffic simulator.[1] Several features led us to use PSO

[1]All the materials generated in the experimentation, software, scenario instances, scripts, cycle programs, traces, figures, etc., are available online at http//neo.lcc.uma.es/problems/traffic-lights.

instead of other optimization techniques. First, the PSO is a well-known algorithm shown to perform a fast convergence to suitable solutions [20]. This is a highly desirable property for an optimal traffic light cycle program, where new immediate traffic light schedules could be required to address updated events in traffic scenarios. Second, the canonical PSO is easy to implement and requires few tuning parameters [17], [20]. Third, PSO is a kind of swarm intelligence algorithm that can inform us of future issues when dealing with this problem using independent agents in the system for online adaptation (a future line of research for us).

The task of SUMO is to evaluate cycle programs (codified as vectors provided by our PSO) of the traffic lights that control the scenario instance. In this paper, we have tested our proposal with two large and heterogeneous metropolitan areas with hundreds of traffic lights located in the cities of Bahía Blanca in Argentina, and Málaga in Spain. Concretely, our main contributions are as follows.

1) We propose a new PSO approach capable of obtaining efficient cycle programs for realistic urban scenarios. In this new approach, the initialization method, the solution encoding, the fitness function, and the velocity calculation have been adapted to deal with optimal traffic light cycle programs.
2) The behavior of our proposal is analyzed under different conditions of road network dimension and traffic density. An analysis of the computational effort is also carried out.
3) In comparison with predefined cycle programs close to real ones, our PSO obtains quantitative improvements in terms of the two main objectives: 1) the number of vehicles that reach their destination and 2) the overall journey time.
4) Further comparisons against other optimization methods [random search (RANDOM), differential evolution (DE), and standard PSO 2011] will justify the use of our PSO for the problem in question.

The remainder of this paper is organized as follows. In Section II, a review of related work in the literature is presented. In Section III, basic concepts of PSO and SUMO are given. In Section IV, our optimization technique proposal is described. Sections V and VI present the experimental methodology used and the results obtained, respectively. Conclusions and future work are given in Section VII.

II. Literature Overview

There are different approaches in the state of the art that deal with traffic light staging problems. Adaptive traffic lights consider the real-time impact of the traffic cycle duration on the traffic network. Much effort has been made in this sense, mainly concerning the use of detectors to sense the traffic and to change the duration of cycle programs, taking into account the actual flow of vehicles [4]–[6].

In this regard, several research studies employ a fuzzy part inside the intersection system control generally combined with other computational intelligence technique or heuristic [21]. In [22], the authors adopted a type-2 fuzzy set and designed a distributed multiagent traffic-responsive signal control system. This system was tested on virtual road networks with several scenarios. Results showed superior performance of the approach in handling unplanned and planned incidents and obstructions. An adaptive traffic control model of signal lights is introduced in [4], consisting of the split cycle offset optimization technique (SCOOT) platform. SCOOT is an adaptive system for managing and controlling traffic signals in urban areas that responds automatically to fluctuations in traffic flow through the use of on-street detectors embedded in the road. This tool is especially useful for areas where traffic patterns are unpredictable.

Another adaptive method is the urban traffic optimization by integrated automation (UTOPIA)/system for priority and optimization of traffic (SPOT), which was designed and developed by the FIAT Research Centre, ITAL TEL, and MIZAR Automazione (Turing) [23]. This system aims to improve the flow in traffic for both private and public transport vehicles. UTOPIA/SPOT is a distributed real-time traffic-control system, especially suitable for countries with advanced public transport services (tested in Italy, Norway, The Netherlands, Sweden, Finland, and Denmark). This system uses a hierarchical-decentralized control strategy, involving intelligent local controllers to communicate with other signal controllers as well as with a central computer.

Different authors have analyzed the use of fuzzy logic controllers at intersections of streets for adaptive tools. In an early study, Lim *et al.* [10] proposed a fuzzy logic controller for real-time local optimization of only one intersection. Later, in [9], a traffic simulator using fuzzy logic agents was developed for traffic lights at isolated junctions. The results showed a minimization of the queue of vehicles on the roads; however, their implementation is very compromised from an economic point of view, and the system's deployment required a great inversion. Other authors applying fuzzy logic were Rahman and Ratrout [24], with satisfactory results in a segment of the King Abdullah Road in Saudi Arabia. The scenario shown in that paper was composed of four intersections with two traffic lights at each one. An exhaustive review of automatic adaptive systems can be found in [5] and [6].

According to the way in which the traffic flow is modeled in stochastic traffic flow methods, we can differentiate between macroscopic and microscopic models [11]. Concerning the optimization strategy, we can find publications in which different resolution techniques have been applied: 1) mathematic models; 2) fuzzy logic approaches; and 3) biologically inspired optimizers.

Several authors employed mathematical techniques for tackling this kind of problem. For example, McCrea and Moutari [1] combined continuous calculus-based and knowledge-based models in order to describe the traffic flow in road networks. Tolba *et al.* [11] introduced a Petri net-based model to represent the traffic flow, from a macroscopic viewpoint (where only global variables are observed) and from a microscopic one (where the individual trajectories of vehicles are considered). More recently, Lammer and Helbing [25] designed a multiagent traffic model inspired by the self-organizing fluctuations of vehicles in traffic jams. They used a simplistic simulation model considering only one direction of movement at a time.

In [8], the authors proposed a special-purpose simulation tool for optimizing traffic signal light timing. This tool provided complete traffic information, although it was limited to working at only four intersections.

Recently, biologically inspired techniques such as cellular automata (CA) and neural networks (NN) have been used for tackling the underlying combinatorial optimization problems, particularly for solving traffic light staging problems. Brockfeld *et al.* [14] applied a CA model in which the city network was implemented as a simple square with a few normal streets and four intersections. Spall and Chin [3] presented an NN for the configuration of control parameters in traffic lights. In this approach, the vehicles needed an additional module for the data management.

Related to the biologically inspired techniques, metaheuristic algorithms [26] have become very popular for solving traffic light staging problems. A first attempt was made by Rouphail *et al.* [27], where a genetic algorithm (GA) was coupled with the CORSIM [28] microsimulator for the timing optimization of nine intersections in the city of Chicago (USA). The results, in terms of total queue size, were limited due to the delayed convergence behavior of the GA.

In [29], the impact of signal time changes with respect to the drivers was analyzed. More precisely, the authors considered the problem of determining optimum signal timings while anticipating the responses of drivers as an instance of the network design problem (NDP). An NDP aims to improve an existing network so that a total network performance measure is optimized with respect to some discrete or continuous design variables, while considering the user's reaction to the improvement. In order to solve the traffic equilibrium problem, they used the simulation-assignment modeling package (SATURN) [30]. The authors applied a macroscopic point of view of the traffic flow and employed a GA to compute the signal setting NDP (cycle time, offset, and green light times for stages). It is important to note that the chromosome (gray code) encoding was done differently for each particular instance being studied. The algorithm was tested with the city of Chester in the U.K., mainly addressing a complete GA parameter analysis, not really the traffic problem.

In [2], following the model proposed in [14], the authors designed a GA with the objective of optimizing the cycle programming of traffic lights. This GA was tested in a commercial area in the city of Santa Cruz de Tenerife, Spain. In this paper, they considered that every intersection had independent cycles. For individual encoding, they used a similar binary (gray code) representation to the one used in [29]. The computation of valid states was done before the algorithm began, and it strongly depended on the scenario instance tackled.

Turky *et al.* [31] used a GA to improve the performance of traffic lights and pedestrian crossing control in a single four-way two-lane intersection. The algorithm solved the limitations of traditional fixed-time control for passing vehicles and pedestrians, and it employed a dynamic control system to monitor two sets of parameters.

A few publications related to the application of PSO for the schedule of traffic lights also exist. One of the most representative was developed by Chen and Xu [32], where they applied a PSO for training a fuzzy logic controller located at each intersection by determining the effective time of green for each phase of the traffic lights. A very simple network with two basic junctions was used for testing this PSO.

Recently, Peng *et al.* [33] presented a PSO with isolation niches for the scheduling of traffic lights. In this approach, a custom microscopic view of the traffic flow was proposed to evaluate the solutions. A purely academic instance with a restrictive one-way road with two intersections was used to test the PSO. Nevertheless, this paper focused on the capacity of isolation niches to maintain the diversity of the swarm and was not particularly concerned with the problem itself.

Finally, Kachroudi and Bhouri [34] applied a multiobjective version of PSO for optimizing cycle programs using a predictive model control based on a public transport progression model. In this paper, private and public vehicles' models are used to carry out simulations on a virtual urban road network made up of 16 intersections and 51 links. Each intersection is then controlled by a traffic light with the same cycle time of 80 s.

All these approaches have focused on different aspects of the traffic light programming. However, three common features (limitations) can be found in all of them.

1) They tackle limited vehicular networks with few traffic lights and a small number of other traffic elements (roads, intersections, directions, etc.). In contrast, our PSO can find optimized cycle programs for large scenarios with hundreds of traffic lights, vehicles, and other elements.
2) Almost all of them have been designed for only one specific scenario. Some of them study the influence of traffic density. Our approach can be easily adapted to represent different scenario topologies. In this paper, we tackle two real scenarios with different combinations of traffic lights and vehicles, fixing a number of 18 instances.
3) They are not compared with other techniques. Our PSO is compared here with four different approaches: 1) a RANDOM algorithm; 2) a DE; 3) the standard PSO 2011; and 4) the cycle program generator provided by SUMO.

III. Basic Solver and Simulator

In this section, the basic concepts of PSO (the core of our solver technique) and the SUMO simulator (involved in the evaluation of solutions) are introduced.

A. Particle Swarm Optimization

Particle swarm optimization [17], [18] is a population-based metaheuristic inspired by the social behavior of birds within a flock, and was initially designed for continuous optimization problems. In PSO, each potential solution to the problem is called a particle position and the population of particles is

Algorithm 1 Pseudocode of PSO

```
initializeSwarm()
computeLeader($b$)
while $g$ < maxIterations do
    for each particle $x^i_g$ do
        $v^i_{g+1}$=updateVelocity($w$, $v^i_g$, $x$ï¿½$_g$, $\varphi_1$, $p_g$, $\varphi_2$, $b_g$)
        $x^i_{g+1}$=updatePosition($x^i_g$, $v^i_{g+1}$)
        evaluate($x^i_{g+1}$)
        $p^i_{g+1}$=update($p^i_g$)
    end for
    $b_{g+1}$=updateLeader($b_g$)
end while
```

called the swarm. In this algorithm, each particle position x^i is updated each iteration g by means of

$$x^i_{g+1} = x^i_g + v^i_{g+1} \tag{1}$$

where term v^i_{g+1} is the velocity of the particle, given by

$$v^i_{g+1} = w \cdot v^i_g + U[0, \varphi_1] \cdot (p^i_g - x^i_g) + U[0, \varphi_2] \cdot (b_g - x^i_g). \tag{2}$$

In this formula, p^i_g is the best solution that the particle i has seen so far, b_g is the global best particle (also known as the leader) that the entire swarm has ever created, and w is the inertia weight of the particle (which controls the trade-off between exploration and exploitation). Finally, φ_1 and φ_2 are the acceleration coefficients that control the relative effect of the personal and global best particles, while $U[0, \varphi_k]$ is a uniform random value in $[0, \varphi_k]$, $k \in 1, 2$, which is sampled anew for each component of the velocity vector and for every particle and iteration.

Algorithm 1 describes the pseudocode of PSO. The algorithm starts by initializing the swarm (line 1), which includes both the positions and velocities of the particles. The corresponding p^i of each particle is randomly initialized, and the leader b is computed as the best particle of the swarm (line 2). Then, for a maximum number of iterations, each particle flies through the search space updating its velocity and position (lines 5 and 6); it is then evaluated (line 7) and its personal best position p^i is also updated (line 8). At the end of each iteration, the leader b is also updated.

The particle swarm optimization algorithm is currently employed in a multitude of engineering problems [18], [35]–[37] showing a successful performance, even when compared with other modern optimization techniques [38], [39]. Nevertheless, the use of PSO for the optimal cycle program and other problems related to the traffic light staging is still limited.

B. SUMO

SUMO [19] is a well-known traffic simulator that provides an open source, highly portable, and microscopic road traffic simulation tool designed to handle large road scenarios. SUMO requires several input files that contain information about the traffic and the streets to be simulated. A network (.net.xml file) holds the information about the structure of the map: 1) nodes, 2) edges, and 3) connections between them. The network can be imported from popular digital maps such as OpenStreetMap (OSM) [40] and converted to a valid SUMO network by means of a series of scripts provided in the SUMO package. We have chosen OSM because it provides both geographic data and traffic light information.

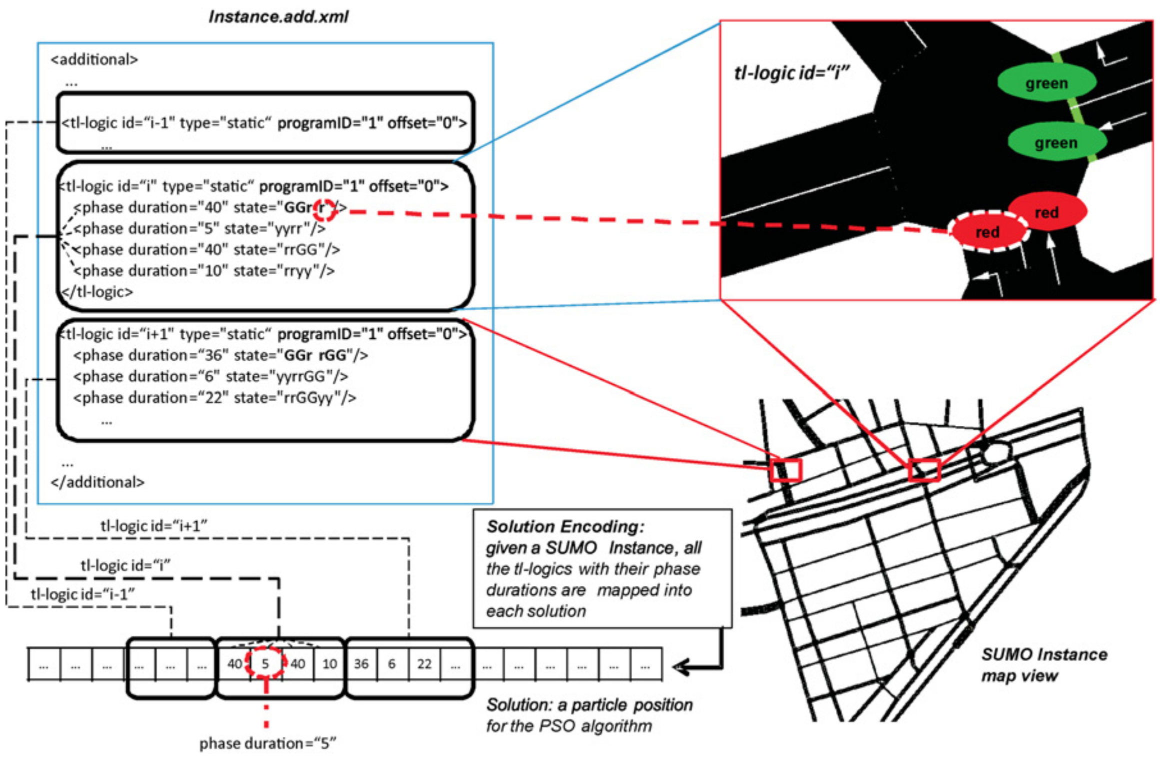
A journey is a vehicle movement from one location to another defined by the starting edge (street), the destination edge, and the departure time. A route is an extended journey, meaning that a route definition contains not only the first and the last edges, but also all the edges the vehicle will pass through. These routes are stored in a demand file (.rou.xml file) either through a route generator given by SUMO, existing routes imported from other software, or by hand. Additional files (.add.xml) can be added to SUMO information about the map or about the traffic lights. SUMO allows replacing and editing information on the cycles of traffic lights by manipulating a file with .add.xml extension. It is important to note that SUMO by default provides the valid combination of states that the traffic light controller can go through inside the map specification file (.net.xml file) [19], and an approximation of interval times for these states [41]. This means that SUMO already incorporates a solver algorithm for the cycle program of traffic lights based on greedy and human knowledge. That solver will be called SUMO cycle program generator (SCPG) in this paper and it will be used in a comparison with our PSO.

The output of a SUMO simulation is registered in a journey information file (.tripinfo.xml) that contains information about each vehicle's departure time, the time the vehicle waited to start at (offset), the time the vehicle arrived, the duration of its journey, and the number of steps in which the vehicle speed was below 0.1 m/s (temporal stops in driving). This information is used to evaluate traffic lights cycle programs.

C. SUMO Data Structure

As previously mentioned, the main objective of our approach is to find optimized cycle programs (duration of color states of traffic lights) for all the traffic lights located in a given urban area. At the same time, these programs have to coordinate traffic lights in adjacent intersections aiming to improve the global flow of vehicles circulating within the established routes. For this reason, we have focused on a microscopic view of the management of traffic agents, but at the same time, we want to evaluate the behavior of all the vehicles in the complete urban scenario during a given time span. The evaluation of the resulting traffic light programs is carried out by means of automatic simulations. For this task, we use SUMO.

The simulation structure of SUMO comprises a series of elements that we have taken into account when developing our traffic scenarios. A SUMO instance for a urban traffic scenario is basically composed of intersections, traffic lights, roads, and vehicles moving along their previously specified routes. The traffic lights are located at intersections (junctions in SUMO) and control the flow of vehicles by following their programs of color states and cycle durations. In this context, all traffic lights located at the same intersection are governed by a common program, since they have to be necessarily



```
<additional>
  ...
<tl-logic id="i-1" type="static" programID="1" offset="0">
  ...
<tl-logic id="i" type="static" programID="1" offset="0">
  <phase duration="40" state="GGrr"/>
  <phase duration="5" state="yyrr"/>
  <phase duration="40" state="rrGG"/>
  <phase duration="10" state="rryy"/>
</tl-logic>
<tl-logic id="i+1" type="static" programID="1" offset="0">
  <phase duration="36" state="GGr rGG"/>
  <phase duration="6" state="yyrrGG"/>
  <phase duration="22" state="rrGGyy"/>
  ...
  ...
</additional>
```

Fig. 1. Intersection with four traffic lights selected from the SUMO instance map. Phase durations (cycles) are specified in the instance.add.xml file and encoded inside a PSO tentative solution.

synchronized for traffic safety. In addition, for all the traffic lights in an intersection, the combination of color states during a cycle period is always kept valid [41] and must follow the specific traffic rules of intersections, in order to avoid vehicle collisions and accidents. For example, two traffic lights located at the same intersection but controlling conflicting movements must not be green at the same time instance. In this regard, as illustrated in Fig. 1, SUMO provides a complete set of valid combinations of color states for each intersection, which cannot be modified during the optimization process. This avoids invalid combinations of color states and restricts the optimization approach to work only with feasible states.

Fig. 1 shows an illustration of the main elements constituting the traffic light cycle programs in SUMO. This program staging is implemented in an XML file (instance.add.xml) that SUMO uses to load cycles and states, prior to the simulation process. In this file, each tl-logic element corresponds to an intersection. Following the model designed by Krajzewicz *et al.* [19], a tl-logic cyclically comprises a sequence of phases during the simulation time. Each phase indicates the corresponding color states (attribute state) of all the traffic lights at the intersection and the duration of this state (attribute duration).

An example of this mechanism can be observed in Fig. 1 where the tl-logic with id=i, which corresponds to an intersection of the SUMO instance, contains four phases with durations of 40, 5, 40, and 10 s (simulation steps). In these phases, the states have four colors, each corresponding to one of the four traffic lights located at the intersection being studied. These states are the valid ones generated by SUMO adhering to real traffic rules. In this instance, the first phase contains the state *GGrr* meaning that two traffic lights are in green (G) and the other two are in red (r) for 40 s. The following phase changes the state of the four traffic lights to *yyrr* (y is amber) for 5 s, and so on. The last phase is followed by the first one and this cycle is repeated throughout the simulation. All the tl-logics in the complete SUMO instance perform their own programming cycles of phases at the same time, thereby constituting the global staging of the traffic lights. Therefore, programming cycles are the main focus of this paper, since we are interested in optimizing the combination of phase durations of all traffic lights (at all intersections) with the aim to improve the global flow of vehicles circulating in an urban scenario instance.

A final indication along these lines concerns the behavior of the vehicles involved in the SUMO instance scenario, which depends on both road directions and speed. SUMO employs a space-discrete extended model as introduced by Krauß [42]. In this model, the streets are divided into cells and the vehicles circulating through the streets go from one cell to another, if allowed. The speed of each vehicle depends on its distance from the vehicle in front of it, with a preestablished maximum speed typical of urban areas (50 km/h in this paper).

IV. PSO for Traffic Light Scheduling

This section describes our optimization solver proposed for the optimal cycle programs of traffic lights. It describes the solution encoding, the fitness function, and finally the global optimization procedure.

A. Solution Encoding

Following the structure of programming cycles adopted by SUMO, the global staging of traffic lights has been easily encoded by means of a vector of integers, where each element represents a phase duration of one state of the traffic lights involved in a given intersection. This way, as shown in Fig. 1, all the phase durations in the tl-logic elements are successively placed in the solution vector, thereby mapping the complete staging of traffic lights in a simple array of integers. The reason for working with this representation is twofold. First, the SUMO simulator itself works with integer values to represent the discrete sequence of time steps (seconds) that make up the complete simulation procedure. Second, real traffic lights also employ integer values to specify the duration of phases in their internal programs.

Despite its simplicity, this solution representation allows our PSO to take into account the interdependency of variables, not only between phase durations in a common `tl-logic` element, but also between traffic lights at adjacent intersections. In this regard, PSO is known to successfully perform in nonseparable problems [20], [43], which is the case in this approach. This last fact is an interesting feature since solutions with coordinated traffic lights (located in different but close intersections) could then be promoted by our optimization algorithm.

B. Fitness Function

Each solution vector (s), codifying the cycle program of the traffic light programs, is evaluated considering the information obtained from the events happening during the simulation by means of the following equation:

$$F_{(s)} = \frac{\left(\sum_{v=0}^{V} j_v(s)\right) + \left(\sum_{v=0}^{V+C} w_v(s)\right) + (C(s) \cdot St)}{V^2(s) + Cr}. \quad (3)$$

The main objective is to maximize the number of vehicles that reach their destination (V) during the simulation time (St), namely, minimizing the number of vehicles that do not reach their destination and remain circulating (C) after the simulation time is reached. A secondary but important objective is to minimize the overall duration of the vehicle's journeys (j_v). It is clear that the overall duration concerns the journey time of the vehicles that reach their destination during the simulation process. To the contrary, vehicles with incomplete journeys (C) consume all the simulation time St, and then, an additional penalization is induced by multiplying these two factors. It is worth mentioning that the terms in (3) are in the range of values $[1e{+}0 \cdots 5e{+}2]$; therefore, additional weighting values were not considered in this formulation. Only the number of vehicles that arrive at their destinations is squared (V^2) in order to prioritize it over the other terms and factors.

An important factor concerns the state of the traffic lights in each precise moment, since it influences the time that each vehicle must stop and wait (w_v), with the consequent delay in its own journey time, e.g., a prolonged state of traffic lights in red could collapse the intersection where it is, and even close other intersections. However, a prolonged state in green could improve the traffic flow in a given area or direction, but also make the traffic flow of other areas and directions worse. In this respect, a balanced number of color lights in the phase duration of the states should promote those states with more traffic lights in green located on streets with a high number of vehicles circulating, and traffic lights in red located on streets with a low number of vehicles moving. The ratio of colors in each phase state of all the tl-logic tl (intersections) can be formulated as

$$Cr = \sum_{k=0}^{tl} \sum_{h=0}^{ph} s_{k,h} \cdot \left(\frac{G_{k,h}}{R_{k,h}}\right) \quad (4)$$

where $G_{k,h}$ is the number of traffic lights in green (G), and $R_{k,h}$ is the number of traffic lights in red in the phase state h (with duration $s_{k,h}$) and in the tl-logic k. The minimum value of $r_{k,h}$ is 1 in order to avoid division by 0.

C. Optimization Strategy

Our optimization strategy is composed of basically two main parts: 1) an optimization algorithm and 2) a simulation procedure. The optimization part is carried out by means of the particle swarm optimization algorithm, which has been specially adapted to find optimal (or quasi-optimal) cycle programs for traffic lights. It works as follows.

1) The initial swarm is composed of a number of particles (solutions) initialized with a set of random values representing the phase durations. These values are within the time interval $[5, 60] \in Z^+$ and constitute the range of possible time spans (in seconds) a traffic light can be kept on a signal color (only green or red; the time for amber is a constant value). We have specified this interval by following several examples of real traffic light programs provided by the City Council of Málaga, Spain.
2) The velocity calculation has been softly modified in order to deal with integer combinatorial values by truncating (with floor $\lfloor\cdot\rfloor$ and ceiling $\lceil\cdot\rceil$ functions) all elements (j) of the new velocity vector as

$$v^i_{g+1}(j) = \begin{cases} \lfloor v^i_{g+\frac{1}{2}}(j) \rfloor & if \quad U(0,1)^i(j) \le \lambda \\ \lceil v^i_{g+\frac{1}{2}}(j) \rceil & \text{otherwise.} \end{cases} \quad (5)$$

In this formula, $v^i_{g+\frac{1}{2}}$ is the intermediate velocity value obtained from (11). The parameter λ determines the probability of performing ceil or floor functions in the velocity calculation ($\lambda = 0.5$ for this paper).
3) The inertia weight changes linearly through the optimization process by using the following equation:

$$\omega = \omega_{\max} - \frac{(\omega_{\max} - \omega_{\min}) \cdot g}{g_{\text{total}}}. \quad (6)$$

In this way, at the beginning of the process, a high inertia ($\omega_{\max}$) value is introduced, which decreases until reaching its lowest value ($\omega_{\min}$). A high inertia value provides the algorithm with exploration capability and a low inertia promotes exploitation.

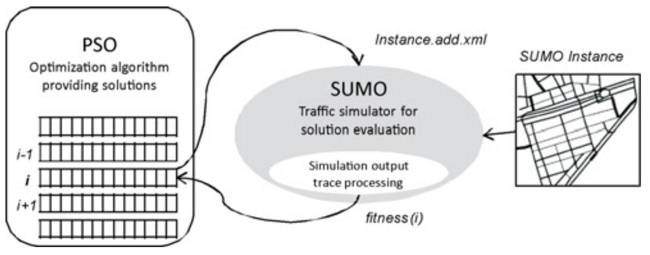

PSO
Optimization algorithm providing solutions
i-1
i
i+1
Instance.add.xml
SUMO
Traffic simulator for solution evaluation
Simulation output trace processing
fitness(i)
SUMO Instance

Fig. 2. Optimization strategy for the cycle program configuration of traffic lights. The algorithm invokes SUMO for each solution evaluation.

The simulation procedure is the way of assigning a quantitative quality value (fitness) to the solutions, thus leading to optimized cycle programs tailored to a given urban scenario instance. This procedure is carried out by means of the SUMO traffic simulator, which accepts new cycle programs of traffic lights and computes the required values in (3).

As Fig. 2 illustrates, when PSO generates a new solution, it is immediately used to update the cycle program. Then, SUMO is started to simulate the scenario instance with streets, directions, obstacles, traffic lights, vehicles, speeds, routes, etc., under the new defined staging of the cycle programs. After the simulation, SUMO returns the global information necessary to compute the fitness function. Each solution evaluation requires only one simulation procedure since vehicle routes in SUMO are generated deterministically. In fact, as suggested in [44], stochastic traffic simulators obtain similar results to deterministic ones, the latter allowing huge computing savings.

In addition, we must note that each new cycle program is statically loaded for each simulation procedure. Our aim here is not to dynamically generate cycle programs during an isolated simulation as is done in agent-based algorithms [45], but rather to obtain optimized cycle programs for a given scenario and timetable. In fact, what real traffic light schedulers actually demand are constant cycle programs for specific areas and for preestablished time periods (rush hours, nocturne periods, etc.), which led us to take this approach.

V. Methodology of Our Paper

This section presents the experimental framework followed to assess the performance of our optimization solver. First, we describe the traffic light scenario instances generated specifically for this paper. Later, the implementation details and parameter settings are presented.

A. Instances

As we are interested in developing an optimization solver capable of dealing with close-to-reality and generic urban areas, we have generated two scenarios by extracting actual information from real digital maps. These two scenarios cover similar areas of approximately 0.42 km^2 and are physically located in the cities of Bahía Blanca in Argentina, and Málaga in Spain. The information used concerns traffic rules, traffic element locations, buildings, road directions, streets, intersections, etc. Moreover, we have set the number of vehicles circulating, as well as their speeds by following current specifications available in the mobility delegation of the city hall of Málaga (http://movilidad.malaga.eu/). This information was collected from sensorized points in certain streets obtaining a measure of traffic density in several time intervals. In the case of Bahía Blanca, we could not obtain this information and so we considered the same number of vehicles as used for the Málaga scenario.

In Fig. 3, the selected areas of the two cities are shown with their corresponding capture views of OSM and SUMO (as explained in Section III-B). Other driving styles such as the Commonwealth/British one could be also tackled with our approach, since we can easily capture areas of U.K. cities with OSM and export them to SUMO, to then work with them by following their directions and traffic rules. The specific features of the selected areas in this paper are as follows.

1) *Rivadavia Square:* Located in the city center of Bahía Blanca (see Fig. 3, top), it has 53 intersections between streets that form a practically regular grid of blocks, as is usual in American cities. Except for the main avenue, almost all streets are one way in opposite directions to each other. Therefore, the great majority of traffic logics (junctions) in this scenario have four traffic lights: straight on, left, and the two on the perpendicular street.
2) *Alameda Avenue:* The city center of Málaga (see Fig. 3, bottom), represents the common irregular structure of European cities, having different street widths and lengths. It has 73 junctions between streets and roundabouts. Each intersection includes from four to 16 traffic lights.

We have considered these two scenarios since they constitute quite different urban areas with heterogeneous structures and traffic organization. Moreover, in order to obtain generalized concluding results, the number of instances used in the experimentation has been increased by incorporating different numbers of vehicles moving through these streets and different numbers of traffic lights operating within the selected areas. Table I contains the combination of traffic logics (intersections) and vehicles used in each instance for each scenario, constituting a total number of 18 instances: nine for Rivadavia Square and nine for Alameda Avenue. We have to note that despite both scenarios having similar scales of traffic logics (20, 30, and 40), the number of traffic lights is not the same, as they contain different intersection shapes.

Concerning the number of vehicles, we have considered three different scales of 100, 300, and 500 cars for each instance (as shown in Table I) circulating throughout the simulation time. Each one of the vehicles takes its own route from origin to destination, circulating with a maximum speed of 50 km/h (typical in urban areas). The routes were previously generated by following random paths and covering as far as possible all network entries. Starting times of vehicles were also randomly (uniform) specified throughout the entire simulation. This means that only a subset of the entire set of vehicles is circulating through the network at the same time. The simulation time was fixed at 500 s (iterations of microsimulation) for each instance. This time was determined

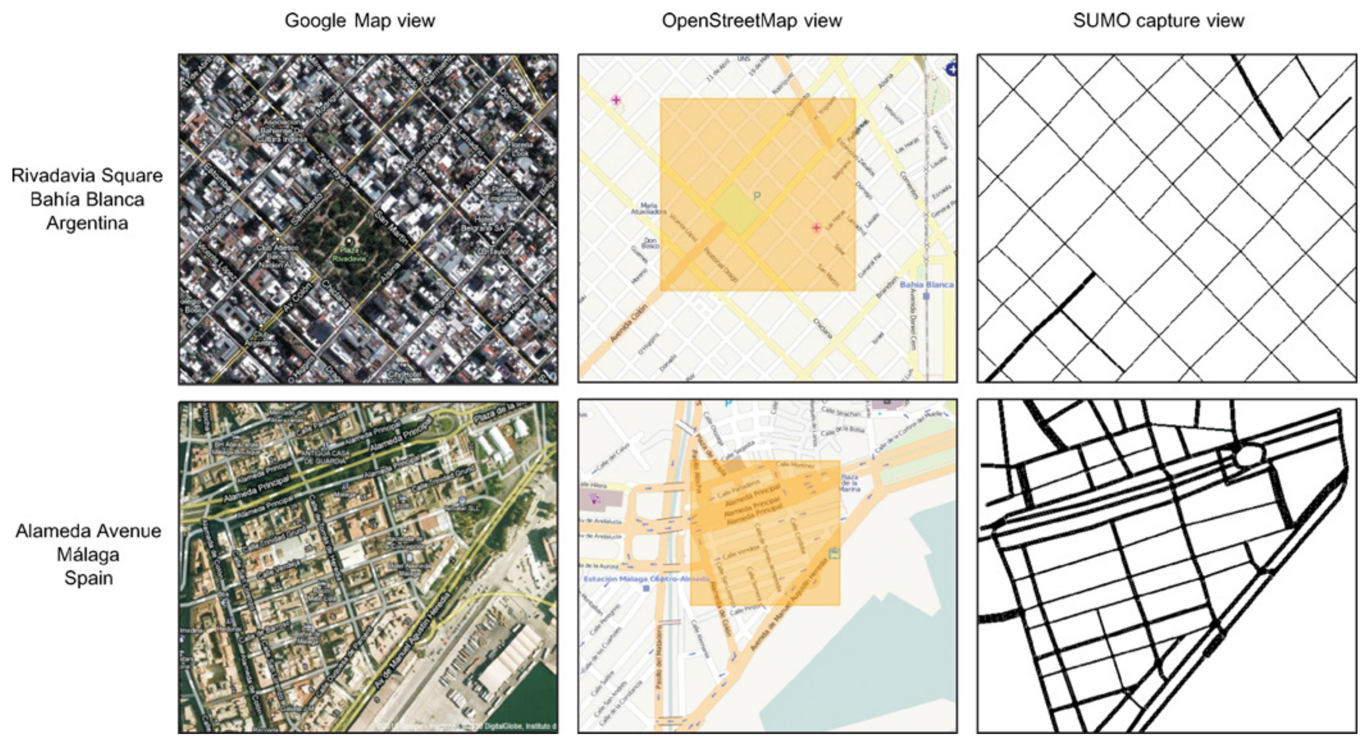


Fig. 3. Process of creation of real-world instances for study. Rivadavia Square (38°43′03′′ S 62°15′56′′O) and Alameda Avenue (36°43′60′′N 4°25′87′′O) instance views. After selecting our area of interest (Google Map view), it is interpreted by means of the OSM tool, and then exported to SUMO in an XML format.

TABLE I
RIVADAVIA SQUARE AND ALAMEDA AVENUE INSTANCES

City instance	Number of traffic logics	Number of traffic lights	Number of vehicles
Rivadavia Square (Bahía Blanca)	20	88	100 300 500
	30	136	100 300 500
	40	176	100 300 500
Alameda Avenue (Málaga)	20	78	100 300 500
	30	130	100 300 500
	40	184	100 300 500

as the maximum time needed for a car to complete its route, even if it must stop at all the traffic lights it comes to. When a vehicle leaves the scenario network, it does not appear again.

B. Experimental Setup

We have used the implementation of the PSO algorithm provided by MALLBA [46], a C++ based framework of metaheuristic algorithms for solving optimization problems. The simulation phase is carried out by executing (in the evaluation of particles) the traffic simulator SUMO release 0.12.0 for Linux. The experiments were performed on computers at the laboratories of the Department of Computer Science at the University of Málaga (Spain). Most of them are equipped with modern dual core processors, 1 GB RAM, and Linux Debian operating system. They operate under a Condor [47] middleware platform that acts as a distributed task scheduler (each task dealing with one independent run of PSO).

For each scenario instance, we have carried out 30 independent runs of our PSO. The swarm (population) size was set to 100 particles performing 300 iteration steps, resulting in 30 000 solution evaluations (SUMO simulations) per run and instance. The choice of these two parameters (swarm size and maximum iteration steps) corresponds to previous tuning experiments as described in Section VI-A. The particle size directly depends on the number of traffic lights of each instance (shown in Table I). The remaining parameters are summarized in Table II. These parameters were set after preliminary executions of PSO with the smallest instances of Rivadavia Square and Alameda Avenue (with 20 traffic logics and 100 vehicles). Specific parameters of PSO were selected as recommended in the studies about the convergence behavior of this algorithm in [20] and [48]. In accordance with these, acceleration coefficients φ_1 and φ_2 were set to 2.05 and inertia weight (ω) decreases linearly along with the increment of the iteration steps from 0.5 to 0.1.

Additionally, we have implemented three algorithms also in the scope of the MALLBA [46] library, in order to establish comparisons against our PSO. These three algorithms are a RANDOM, a DE [49], and the standard PSO 2011

TABLE II

SIMULATION AND PSO PARAMETERS

Solver phase	Parameter	Value
Simulation details	Simulation time (steps)	500 s.
	Simulation area	0.45 km^2
	Number of vehicles	100/300/500
	Vehicle speed	0–50 km/h
	Number of traffic logics	20/30/40
PSO parameters	Maximum number of evaluations	30 000
	Swarm size	100
	Particle size (number of traffic lights)	88/136/176 78/130/184
	Local coefficient (φ_1)	2.05
	Social coefficient (φ_2)	2.05
	Maximum inertia ($w_{\max}$)	0.5
	Minimum inertia ($w_{\min}$)	0.1
	Velocity truncation factor (λ)	0.5

Algorithm 2 Pseudocode of RANDOM

```
1: initializeSolution($x$)
2: $i \leftarrow 0$
3: while $i <$ Max_Number_of_Evaluations do
4:     generate($x_i$)    //new solution
5:     if f($x$) $\geq$ f($x_i$) then
6:         $x \leftarrow x_i$
7:     end if
8:     $i \leftarrow i + 1$
9: end while
```

(SPSO2011) [16]. Thus, performing the same experimentation procedure, we expect to obtain some insights into the power of our proposal (how intelligent it is) regarding a technique without any heuristic information in its operation (RANDOM), and with regard to two other difference-vector-based metaheuristics: 1) DE and 2) SPSO2011. In the case of SPSO2011, it is the last PSO proposal in [16] and uses a different quantisation/discretization method to our PSO. The maximum number of evaluations was set to 30 000, as for PSO.

1) *RANDOM:* The pseudocode of the RANDOM algorithm is shown in Algorithm 2. It basically performs by keeping just the best solution found so far in the optimization procedure.

2) *DE:* In DE, the task of generating new individuals is performed by differential operators such as the differential mutation and crossover. A mutant individual w^i_{g+1} is generated by

$$w^i_{g+1} = v^{r1}_g + F \cdot (v^{r2}_g - v^{r3}_g) \qquad (7)$$

where $r1, r2, r3 \in \{1, 2, \ldots, i-1, i+1, \ldots, N\}$ are random mutually different integers, which are also different from index i. The mutation constant $F > 0$ stands for the amplification of the difference between the individuals v^{r2}_g and v^{r3}_g, and it avoids the stagnation of the search process.

In order to further increase the diversity in the population, each mutated individual undergoes a crossover operation with the target individual v^i_g, by means of which a trial individual u^i_{g+1} is generated. A randomly chosen position is taken from the mutant individual to prevent the trial individual replicating the target individual

$$u^i_{g+1}(j) = \begin{cases} w^i_{g+1}(j) & \text{if} r(j) \leq Cr \text{or} j = j_r \\ v^i_g(j) & \text{otherwise.} \end{cases} \qquad (8)$$

As shown in (8), the crossover operator randomly chooses a uniformly distributed integer value j_r and a random real number $r \in (0, 1)$, also uniformly distributed for each component j of the trial individual u^i_{g+1}. Then, the crossover probability Cr and r are compared just like j and j_r. If r is lower or equal to Cr (or j is equal to j_r), then we select the jth element of the mutant individual to be allocated in the jth element of the trial individual u^i_{g+1}. Otherwise, the jth element of the target individual v^i_g becomes the jth element of the trial individual. For this paper, F and Cr have been set to 0.5 and 0.9, respectively, as initially recommended in [49].

Finally, a selection operator decides on the acceptance of the trial individual for the next generation if and only if it yields a reduction (assuming minimization) in the value of the fitness function $f()$, as shown by

$$v^i_{g+1} = \begin{cases} u^i_{g+1} & \text{if} f(u^i_{g+1}) \leq f(v^i_g) \\ v^i_g & \text{otherwise.} \end{cases} \qquad (9)$$

Algorithm 3 Pseudocode of DE

```
1: initializePopulation()
2: while $g <$ maxIterations do
3:     for each individual $v^i_g$ do
4:         choose mutually different($r_1, r_2, r_3$)
5:         $w^i_{g+1}$ = mutation($v^{r1}_g$, $v^{r2}_g$, $v^{r3}_g$, $F$)   //(7)
6:         $u^i_{g+1}$ = crossover($v^i_g$, $w^i_{g+1}$, $cp$)   //(8)
7:         evaluate($u^i_{g+1}$)
8:         $v^i_{g+1}$ = selection($v^i_g$, $u^i_{g+1}$)   //(9)
9:     end for
10: end while
```

Algorithm 3 shows the pseudocode of DE. After initializing the population, the individuals evolve during a number of iterations (maxIterations). Each individual is then mutated (line 5) and recombined (line 6). The new individual is selected (or not) following the operation of (9) (lines 7 and 8).

In order to make a fair comparison, we also adapted the DE for dealing with integer values in the solution codification, that is, using the same mechanism of ceiling/flooring ($\lceil \cdot \rceil / \lfloor \cdot \rfloor$) functions as done in the velocity vector calculation of PSO [see (5)]

$$w^i_{g+1}(j) = \begin{cases} \lfloor w^i_{g+\frac{1}{2}}(j) \rfloor & \text{if} \;\; U(0,1)^i(j) \leq \lambda \\ \lceil w^i_{g+\frac{1}{2}}(j) \rceil & \text{otherwise.} \end{cases} \qquad (10)$$

In the case of DE, the truncation method is applied to the mutant vector w^i_{g+1}, as specified in (10), also with $\lambda = 0.5$.

3) *Standard PSO 2011:* We have selected the standard PSO 2011 (to compare with our proposal) from all the existing versions of PSO in the literature, since it includes a series of new advances proposed by prominent researchers in this area [16]. Some of these interesting advances consist of rotation invariance method, new particles generation in

Algorithm 4 Pseudocode of standard PSO 2011

```
1:  initializeSwarm()
2:  while $g$ < maxIterations do
3:      for each particle $x_g^i$ do
4:          $b_g^n$=bestNeighbourSelection($x_g^i$, $n$)
5:          $v_{g+1}^i$=updateVelocity($w$, $v_g^i$, xï¿½$_g$, $\varphi_1$, $p_g$, $\varphi_2$, $b_g^n$)
6:          $x_{g+1}^i$=Q(updatePosition($x_g^i$, $v_{g+1}^i$))
7:          evaluate($x_{g+1}^i$)
8:          $p_{g+1}^i$=update($p_g^i$)
9:      end for
10: end while
```

hypersphere, Gaussian random number generator, and mid-tread quantization/discretization method.

The main feature of the standard PSO 2011 consists in the velocity vector (v_{g+1}^i) calculation which is given by

$$v_{g+1}^i = w \cdot v_g^i + Gr_g^i - x_g^i + HS(Gr, \parallel Gr - \vec{x_g} \parallel) \tag{11}$$

with

$$Gr_g^i = \frac{x_g^i + p_g'^i + l_g'^i}{3} \tag{12}$$

$$p_g'^i = x_g'^i + c \cdot (p_g^i - x_g^i) \tag{13}$$

$$l_g'^i = x_g^i + c \cdot (l_g^i - x_g^i). \tag{14}$$

In these formulas, p_g^i is the best solution that the particle i has seen so far, l_g^i is the best particle of a neighborhood of k other particles (also known as the social best) randomly (uniform) selected from the swarm, and w is the inertia weight of the particle (it controls the trade-off between exploration and exploitation). The acceleration coefficient $c > 1$ is a normal (Gaussian) random value with $\mu = 1/2$ and $\rho = 1/12$. This coefficient is sampled anew for each component of the velocity vector. Finally, HS [16] is a distinctive element of the standard PSO 2011 with regard to the previous ones. HS is basically a random number generator within a hypersphere space, with Gr as the center of gravity. That is, Gr is calculated as the equidistant point to p_g', l_g', and x_g.

Since the optimal cycle programming requires solutions encoded with a vector of integers (representing phase durations), we have used the quantization method provided in the standard specification of PSO 2011 [16]. This quantization is applied to each new generated particle [in (1)], and transforms the continuous values of particles to discrete ones. It consists of a mid-thread uniform quantizer method as specified in (15). The quantum step is set here to $\Delta = 1$

$$Q(x) = \Delta \cdot \lfloor x/\Delta + 0.5 \rfloor. \tag{15}$$

Algorithm 4 describes the pseudocode of the standard PSO 2011. The algorithm starts by initializing the swarm (line 1). The corresponding elements of each particle (solutions) are initialized with random values representing the phase durations. These values are within the time interval $[5, 60] \in Z^+$ and constitute the range of possible time spans (in seconds). Then, for a maximum number of iterations, each particle moves through the search space updating its velocity and position (lines 4–6), it is then evaluated (line 7), and its personal best position p^i is also updated (line 8). Finally, the best particle found so far is returned.

4) *Deterministic Cycle Programs Generator:* Finally, as previously commented on in Section III-B, SUMO provides a deterministic algorithm for generating cycle programs (SCPG). Then, we also compare the cycle programs obtained by our PSO with those obtained by SUMO. This last algorithm basically consists of assigning the phase durations of the traffic logics with fresh values in the range of [6, 31], according to three factors:

1) the proportion of green states in the phases;
2) the number of incoming lanes entering the intersection;
3) the braking time of the vehicles approaching the traffic lights.

Further information on this algorithm can be found in [19].

VI. Analysis and Discussion of Results

The results and the analyses are presented in this section from several viewpoints. First, we study the performance of our optimization solver in comparison with other techniques, and its ability to report successful cycle programs for the different instances. After this, we present a brief report on the computational effort required for the experiments. Later, we focus on the problem domain and we examine representative reported solutions with the aim of justifying the use of our PSO with a potentially truly positive impact on traffic flow.

A. Performance Analysis of Algorithms

Before any comparison takes place, we first wish to show a representative view of the internal behavior of our PSO under different conditions of swarm size and maximum number of iterations. We used this investigation as a basis for setting the most convenient values in the following experimentation. So, Fig. 4 plots the traces of progress of the best fitness values (median run out of 30 independent executions) of PSO tackling with the Alameda Avenue instance with 30 traffic logics and 300 vehicles. These traces correspond to different configurations of swarm size (SS) with 50, 100, and 200 particles, and maximum number of iterations (MaxIt) with 100, 300, and 500 steps to the stop condition. It is worth noting that the number of iteration steps directly influences the inertia weight (in the velocity calculation of PSO), and hence, this parameter should be studied separately in combination with all the different values of the swarm size.

As shown in Fig. 4, for almost all the combinations (of SS and MaxIt), our PSO got to converge on the interval of 100 and 300 iterations, showing the combination of 100 particles in the swarm and 300 iteration steps the best performance results. In fact, for this configuration, the fitness clearly improved after 100 iterations to finally converge just before 200 iteration steps (20 000 function evaluations). We have to mention that other configurations of PSO with SS=200 and MaxIt=500 also obtained such successful results although it required a higher computational cost with more than 50 000 function evaluations

TABLE III

MEDIAN FITNESS VALUES OBTAINED BY PSO FOR ALL OF THE SCENARIO INSTANCES

Instance	NTL	Number of Vehicles								
		100			300			500		
		PSO	RANDOM	SCPG	PSO	RANDOM	SCPG	PSO	RANDOM	SCPG
Rivadavia Square	20	**1.64E+00**	2.91E+00	2.38E+00	**8.40E−01**	1.45E+00	9.24E−01	**7.93E−01**	1.51E+00	9.56E−01
	30	**1.80E+00**	3.11E+00	2.45E+00	**9.09E−01**	1.65E+00	9.57E−01	**8.79E−01**	1.72E+00	9.89E−01
	40	**1.79E+00**	3.08E+00	2.49E+00	**9.11E−01**	1.75E+00	9.76E−01	**8.96E−01**	1.74E+00	9.93E−01
Alameda Avenue	20	**9.47E−01**	1.68E+00	1.49E+00	**8.44E−01**	1.62E+00	1.29E+00	**4.10E+00**	7.87E+00	2.35E+01
	30	**1.56E+00**	3.55E+00	5.12E+00	**1.74E+00**	4.52E+00	6.00E+00	**7.67E+00**	1.33E+01	3.31E+01
	40	**1.88E+00**	3.98E+00	5.38E+00	**2.87E+00**	7.33E+00	1.83E+01	**9.39E+00**	1.64E+01	1.47E+01

Median fitness values obtained by RANDOM and by SCPG algorithms are also provided. NTL is the number of traffic logics.

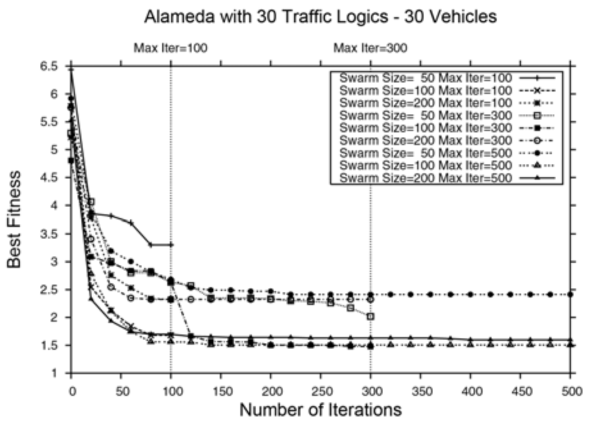


Fig. 4. Traces of progress of the best fitness values (median out of 30 independent runs) of PSO tackling with the Alameda Avenue instance with 30 traffic logics and 300 vehicles. The traces correspond to different configurations of swarm sizes (with 50, 100, and 200 particles) and maximum number of iterations (100, 300, and 500) as the stop condition.

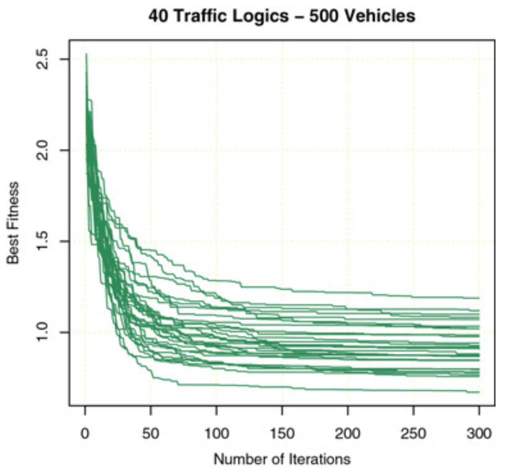


Fig. 5. Trace progress of the best fitness values in 30 independent runs of PSO tackling with the Rivadavia Square instance with 40 traffic logics and 500 vehicles.

(SS=100 and MaxIt=500), in contrast with 30 000 ones in the case of SS=100 and MaxIt=300. Therefore, we opted to set 100 particles in the swarm and a maximum of 300 iteration steps in our experimentation.

From another viewpoint, Fig. 5 plots the trace progress of the best fitness values obtained in 30 independent runs of PSO

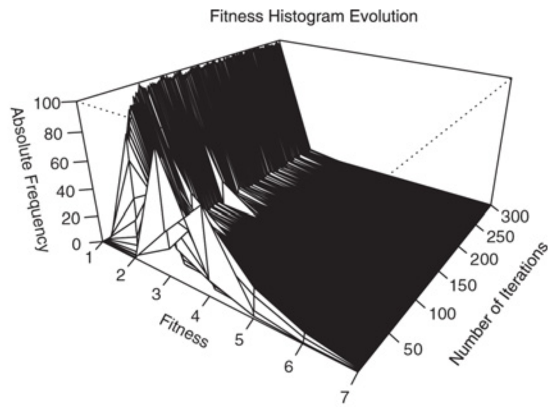


Fig. 6. Swarm fitness histogram through 300 iterations in the optimization of the Rivadavia Square scenario with 40 traffic logics and 500 vehicles.

when solving the Rivadavia Square instance with 40 traffic logics and 500 vehicles. In this figure, we can observe that for all executions, our algorithm practically converged after the first 150 iterations, using the remaining time to only slightly refine solutions. In addition, all the computed solutions are close to each other in quality, but different between each other. These are desirable features in terms of convergence and robustness, since we can offer an expert, a varied set of accurate cycle programs in a reduced time.

To better explain this, Fig. 6 plots the absolute frequency of the fitness distribution of the entire swarm through the optimization process of one typical execution. Specifically, it illustrates one of the 30 independent runs of our PSO tackling the Rivadavia Square scenario with 40 traffic logics and 500 vehicles. We can see that the initial particles are diverse and with high cost values ($\simeq$ 7), although they were able to converge in a low fitness region (≤ 1) during the second half of the execution process. In this specific run, 475 vehicles out of 500 reached their particular destinations (95%) in a simulation time lower than 500 s (the complete number of microsimulation steps). This accurate behavior is also found in all executions and for all instances, and it represents another interesting feature of our approach.

Table III contains the median fitness values obtained by the proposed PSO for all the scenario instances. Additionally, the median fitness values obtained by the RANDOM algorithm, and the results of the SCPG, are also provided in order to permit comparisons. In this table, we can easily check

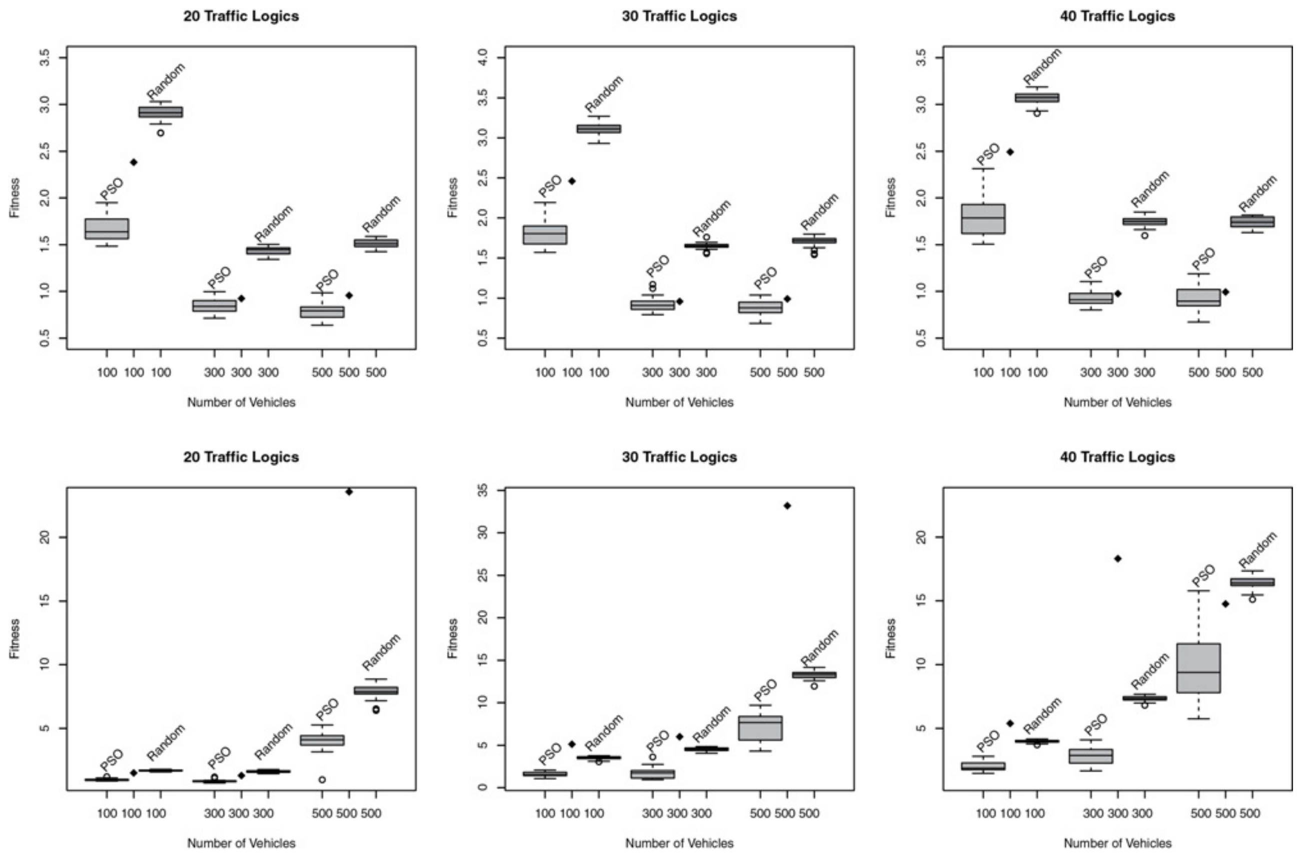


Fig. 7. Boxplot representation of distribution results of Rivadavia Square (three at the top) and Alameda Avenue (three at the bottom) instances with 20, 30, and 40 traffic logics, and 100, 300, and 500 vehicles. The results of SCPG are represented with a ♦ point since this technique always returns the same deterministic result for a given instance.

that PSO obtained the best median fitness (marked in bold) independently of both the number of vehicles and the number of traffic logics in each scenario instance.

In order to provide statistically meaningful comparisons, we have applied a signed ranked (Wilcoxon)[2] test [50] to the numerical distributions of the results. We have used this nonparametric test as the resultant distributions usually violate the condition of normality required to apply parametric tests (Z Kolmogorov–Smirnov = 0.04).[3] Another implication of the violation of the normality condition is the use of median values (as shown in Table III) instead of other measures such as the mean and the standard deviation [51]. The confidence level was set to 95% (α=0.05), which allows us to ensure that these results are statistically different if they result in a p-value < 0.05.

In effect, for all the instances, the differences between the distributions out of 30 independent runs resulted with p-values < 0.05. In general, the differences in the distributions of the medians (see Table III) resulted in a global p-value of 5.73E−7 when comparing PSO with RANDOM, and a global p-value of 6.33E−5 when comparing PSO with SUMO. Therefore, we can claim that our PSO obtained statistically better results than the other two algorithms compared: 1) RANDOM (stochastic search) and 2) SCPG (deterministic). This also means that our algorithm is intelligent and competent when compared to greedy information and human knowledge.

A summary of these results can be seen in Fig. 7, where the boxplots of the distribution fitness of PSO and RANDOM are plotted. The results of SCPG are represented with a point since this technique always returns the same deterministic result. As expected, the distributions of PSO show better lower quartiles, medians, and upper quartiles than RANDOM for all the instances. Regarding SCPG, we can see that the median values of PSO are generally better than the results of SCPG. Only in the case of Rivadavia Square with high densities of traffic (300 and 500 vehicles) do the SUMO results get close to the upper quartiles of our PSO distributions.

Concerning the two scenario instances, the resulting fitness values in Rivadavia Square are generally better than the ones obtained in Alameda Avenue. This difference in the results is more noticeable when a large number of vehicles is circulating (500), where the median fitness values differ in two orders of magnitude (from 7.93E−01 to 3.31E+01). We suspect that the regular structure of Rivadavia Square (see Fig. 3) makes the

[2]The null hypothesis in Wilcoxon test is that the median difference between pairs of observations is zero, with a confidence level of 95% (α=0.05) in our case. This means that if resultant p-value is lower than 0.05, then the compared distributions are different.

[3]Kolmogorov–Smirnov compares the accumulated distribution of observed data with the accumulated distribution expected for a Gaussian distribution, obtaining the p-value based on both discrepancies. Therefore, it measures the quality of a normal fitting to the data and then can be used to test the hypothesis of normality in the population distribution.

TABLE IV

MEDIAN FITNESS VALUES OBTAINED BY OUR PSO, DE, AND STANDARD PSO 2011 FOR ALL OF THE SCENARIO INSTANCES

Instance	NTL	Number of Vehicles								
		100			300			500		
		PSO	DE	SPSO2011	PSO	DE	SPSO2011	PSO	DE	SPSO2011
Rivadavia Square	20	**1.64E+00**	2.18E+00	1.87E+00	**8.40E−01**	9.94E−01	9.82E−01	**7.93E−01**	9.80E−01	1.22E+00
	30	**1.80E+00**	2.25E+00	2.33E+00	**9.09E−01**	1.11E+00	1.28E+00	**8.79E−01**	1.02E+00	1.44E+00
	40	**1.79E+00**	2.23E+00	2.50E+00	**9.11E−01**	1.13E+00	1.25E+00	**8.96E−01**	1.10E+00	1.40E+00
Alameda Avenue	20	**9.47E−01**	1.22E+00	1.11E+00	**8.44E−01**	1.07E+00	9.12E−01	**4.10E+00**	4.98E+00	4.71E+00
	30	**1.56E+00**	2.19E+00	2.49E+00	**1.74E+00**	2.54E+00	3.47E+00	**7.67E+00**	8.57E+00	1.11E+01
	40	**1.88E+00**	2.54E+00	3.21E+00	**2.87E+00**	4.06E+00	5.32E+00	**9.39E+00**	1.17E+01	1.30E+01

NTL is the number of traffic logics.

traffic more fluid in this scenario than in Alameda Avenua (with irregular European design), which could lead the PSO to obtain different ranges of results in similar conditions.

B. Comparison With Other Metaheuristic Algorithms: DE and Standard PSO 2011

For a further comparison, we have studied the performance of two other metaheuristic algorithms for the same experimental procedure as our proposal. A first comparison concerns a DE algorithm (as described in Section V-B), by means of which we expect to better justify the use of PSO on the traffic light cycle program. Second, we compare our PSO with the standard PSO 2011, which performs a different velocity calculation and discretization method.

The median fitness values (out of 30 independent runs) resulted in the experimentation of DE and SPSO2011 are included in Table IV together with the ones of our PSO for the two scenario instances: 1) Rivadavia Square and 2) Alameda Avenue. Again, we confirm that the PSO obtained the best median fitness for all the combinations of number of vehicles and number of traffic logics in each scenario instance. In general, using a Wilcoxon signed rank test with α=0.05, the differences in the distributions of the medians (see Table IV) resulted in a global p-value of 1.94E−4 when comparing PSO with DE, and a global p-value of 1.96E−4 when comparing PSO with SPSO2011. In the first case, the different learning procedures that our PSO and DE perform is the main factor that influences the statistical differences in results, since these two algorithms used the same discretization method. In the second case, the different velocity calculation methods influence the algorithms' performances of our PSO and SPSO2011, indicating that our proposal is better than the last standard PSO for the problem under consideration.

In a further comparison, SPSO2011 showed better fitness values than DE, resulting in a global p-value of 1.47E−2. If we take into account that DE uses a similar discretization method as our PSO, the last results lead us to suspect that the different discretization of vectors marginally influences the global algorithm's performance.

Therefore, within the scope of the experimental framework adopted in this approach, we can claim that our PSO also obtained statistically better results than the other metaheuristic approaches (DE and SPSO2011) used to solve the optimal cycle program of traffic lights.

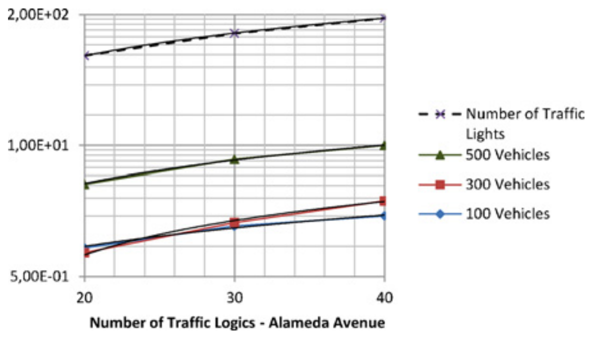


Fig. 8. Increment of the median fitness with regard to the number of traffic lights for the Alameda Avenue scenario. The values are in logarithmic scale.

C. Scalability Analysis

To study the scalability of our proposal, we now focus on the influence of the two main factors defining the complexity of the instances: 1) the number of traffic logics (20, 30, and 40) and 2) the number of vehicles circulating (100, 300, and 500).

The first observation concerns the number of traffic logics (and hence, the number of traffic lights), since it determines the dimensionality of the problem. In Fig. 8, we can observe that the mean fitness values increase with the number of traffic logics, as expected, although this increment is moderate with regard to the number of traffic lights (dotted lines).

A second interesting observation can be obtained from Fig. 7, where the distribution of results concerning the number of vehicles is completely different for both scenarios. Thus, in Alameda Avenue (three boxplots at the top), the distribution of results gets worse with an increase in the number of vehicles. This seems logical since a high number of cars increases the possibility of traffic jams being generated. In addition, we must take into account that the number of vehicles that arrive at their destination directly influences the fitness function. To the contrary, in Rivadavia Square (three boxplots at the bottom), the distribution of results improves as the number of vehicles increases. In this case, we suspect that the particular shape of this scenario, with parallel streets and thus organized flow, could influence in the number of vehicles that quickly reach their destinations and leave the scenario, hence introducing great benefits to the fitness calculation.

TABLE V

MEAN TIME AND STANDARD DEVIATION IN SECONDS OF OUR PSO TO COMPUTE ALL THE EXPERIMENTS

Instance	Number of Traffic Logics	Number of Vehicles		
		100	300	500
Rivadavia Square	20	4.14E+02±6.74E+01	5.94E+02±6.83E+01	7.25E+02±6.53E+01
	30	4.09E+02±6.11E+01	5.04E+02±5.54E+01	7.44E+02±6.51E+01
	40	3.56E+02±5.42E+01	4.43E+02±4.60E+01	6.66E+02±5.66E+01
Alameda Avenue	20	4.30E+02±4.55E+01	1.20E+03±7.58E+01	1.59E+03±9.50E+01
	30	5.46E+02±5.48E+01	1.14E+03±7.43E+01	1.51E+03±8.59E+01
	40	5.12E+02±5.12E+01	1.23E+03±8.03E+01	1.48E+03±8.51E+01

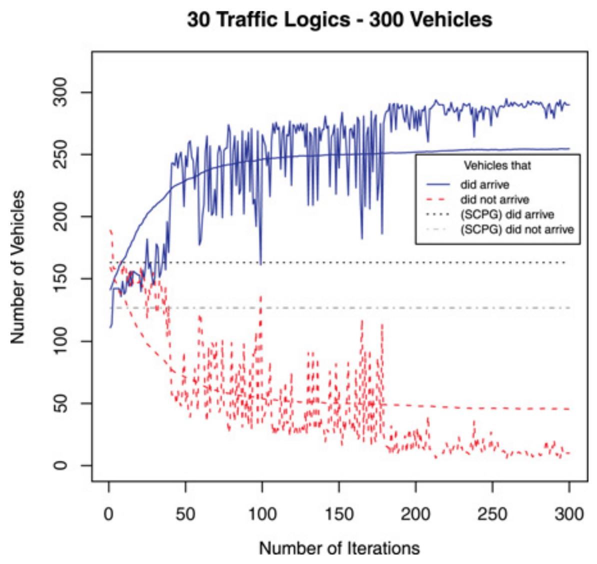


Fig. 9. Number of vehicles that did reach their destination (continuous lines) versus vehicles that did not reach their destination (dotted lines) in the studied time frame. Overlapped curves show the mean number of vehicles (out of 30 independent runs) that did arrive and did not arrive at their destination. SCPG results are also shown with dotted straight lines.

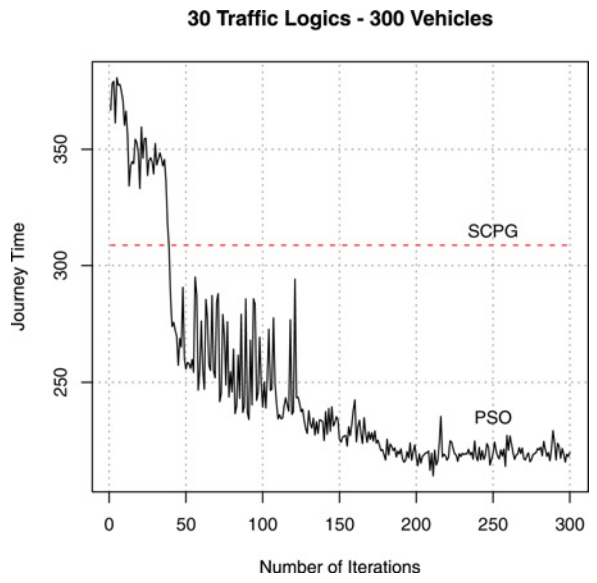


Fig. 10. Mean journey time of vehicles calculated for each one of the simulations performed through a representative run of PSO. SCPG results are also shown with a dotted straight line. The *y*-axis represents the journey time in seconds.

D. Computational Effort

Table V contains the mean times (and standard deviations) in seconds required by our PSO to compute all the experiments. We must state that these times are averaged, since they were calculated in the scope of a Condor [47] middleware with a pool of machines with different specifications.

The lowest execution time (3.56E+02 s) was required for solving the Rivadavia Square scenario with 40 traffic logics and 100 vehicles. The highest time (1.59E+03 s) was used in the resolution of the Alameda Avenue scenario with 20 traffic lights and 500 vehicles. All these times are in a range from 6.33 to 26.5 min, which is acceptable for the human experts in civil engineering, designing and taking decisions on the traffic network.

We stress that the computing time increases with the number of vehicles (common sense), although it decreases with the number of traffic lights (counterintuitive). This fact could be due to the optimized cycle programs that control a great number of traffic lights. These optimized traffic lights enhance the traffic flow, meaning the cars get to their destinations quickly, thereby reducing the computing load of the simulation.

E. Analysis of Solutions

Finally, in this section, we focus on the cycle programs obtained as solutions by our PSO, and the possible benefits they can offer to the actual users in this field. So we show the broad impact of using our strategy, able to compute realistic and comprehensive traffic light cycle programs.

In this context, for each iteration step of the PSO and for each particle in the swarm, we have saved the information obtained from each simulation (solution evaluation) about both the number of vehicles that reached their destination and the average duration of their journeys. In this way, we can distinguish the progressive improvement in the traffic flow obtained from the initial solutions to the final ones, throughout the complete optimization procedure.

A representative example can be observed in the optimization process of the Alameda Avenue scenario with 30 traffic logics (130 traffic lights in the cycle program) and 300 vehicles. First, in Fig. 9 we can see the trace of the number of vehicles that did reach their destination (upper continuous line) versus the number of vehicles that did not reach their destinations (lower dotted line) for each iteration step in a run of PSO. The overlapped curves show the mean number of

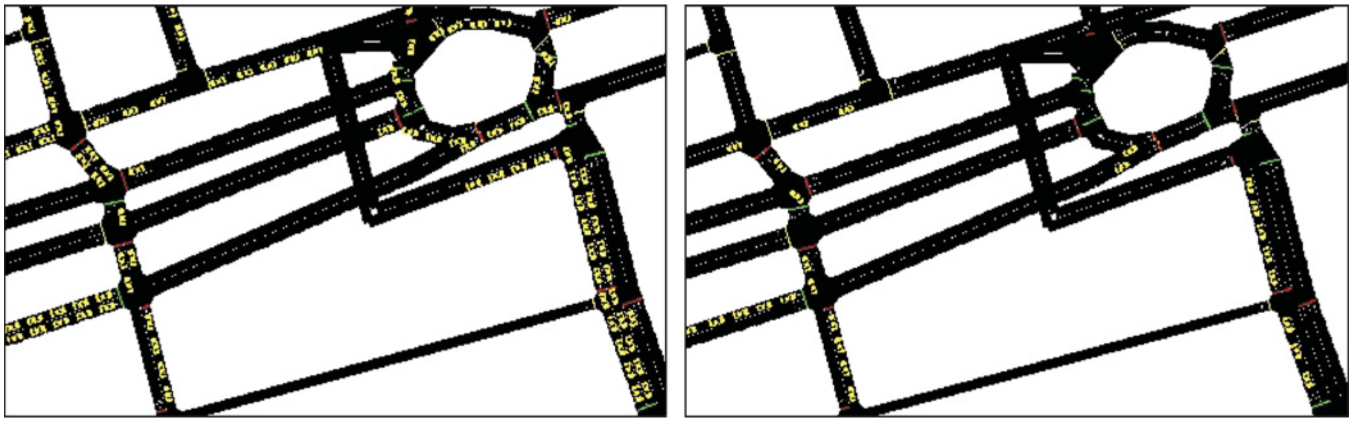


Fig. 11. Simulation traces of the traffic flow (cars in white) resulting from the cycle programs generated by both (left) SCPG and (right) PSO. The pictures show snapshots at the end of the simulation time. The reader can see that the SCPG leaves a dense traffic, while PSO has cleared the routes and the traffic is very fluid and smooth.

vehicles (out of 30 independent runs) that did arrive and did not arrive at their destination. In contrast, this figure also shows the results (in dotted straight lines) of the SCPG (SUMO algorithm) for this same instance.

We can easily see in Fig. 9 how the number of vehicles that did arrive (did not arrive) at their destination increases (decreases) as the algorithm reaches the stop condition of 300 iterations. In fact, at the initial steps of the optimization process, the number of vehicles that reached their destination was lower than the ones resulting in the cycle program generated by SCPG. However, in the final steps of the PSO procedure, the solutions obtained show a high quality in terms of the traffic flow, since 295 vehicles of the initial 300 (98.33%) finalized their trips successfully. Moreover, a mean number of 255 vehicles completed their journeys in the final solutions of PSO (average of 30 runs). This contrasts with the 160 vehicles that reached their destination in the SUMO cycle program. The improvement obtained by our PSO over SCPG is 31.66%.

Another interesting behavior that can be observed in Fig. 9 is the alternating peaks and valleys that appear in the curves of the single run of PSO. These peaks represent solutions with an accurate fitness but with a low number of cars reaching their destinations. This can be due to the fact that the fitness function [see (3)] promotes cycle programs with large durations of phases in which the proportion of traffic lights in green is higher than in red. For certain intersections with several secondary streets and only one big avenue, the traffic lights controlling this avenue could extend their states in red, thus resulting in a traffic jam that could delay the traffic in other adjacent intersections/streets. A string influence on the successful journeys in the fitness function (promoting the number of vehicles that arrive and penalizing it when vehicles do not arrive) leads the PSO to avoid these kind of solutions.

From another viewpoint, Fig. 10 plots the trace of the average journey time employed by the vehicles in the resulting solutions of PSO through all the iterations of an example run. In this case, the journey time becomes shorter as the algorithm approaches the stop condition. We must note that in the calculation of the journey time, the vehicles that did not arrive at their destinations took 500 s, the complete simulation time. For this reason, SCPG solutions showed an average journey time of 308.75 s, while PSO solutions obtained a journey length of 78 s, which means an improvement of 74% with respect to the SCPG solution. In this specific case, 295 vehicles (of 300) completed their journey during the simulation time with an average journey time of 78 s to complete the urban scenario of 650 $\times$ 650 m. In the worst case, the remaining five vehicles will complete their trips in at most 500+78 s, that is, the complete simulation time plus the average journey time.

Finally, in order to clarify the final implications of using (or not using) an optimized cycle program, Fig. 11 shows the simulation traces of the traffic flow resulting from solutions generated by both (left) SCPG and (right) PSO. The pictures were captured at the end of the simulation time (500 s), and correspond to two simulation procedures of the scenario instance Alameda Avenue with 40 traffic logics (184 traffic lights) and 500 vehicles. As we can observe, the traffic density of the SCPG cycle program is clearly higher than that of PSO, even showing the former several intersections with traffic jams. As to the PSO cycle program, all intersections are unblocked at the end of the study.

VII. Conclusion

In this paper, we proposed an optimization technique based on a particle swarm optimization algorithm that can find successful traffic light programs. For the evaluation of solutions, we used SUMO, a well-known microscopic traffic simulator. For this paper, we tested two extensive and heterogeneous metropolitan areas located in Bahía Blanca and Málaga.

From two scenarios, a total number of 18 different numerical instances were generated depending on the number of vehicles circulating and the number of traffic lights operating. A series of analyses was carried out from different viewpoints: 1) the performance of the optimization technique; 2) the scalability; 3) the computational effort; and 4) the quality of solutions. From these, the following conclusions can be extracted.

1) Our PSO solver performs successfully in the generation of optimized cycle programs for big realistic traffic scenarios. For all the instances, our proposal obtained robust results statistically better than the other two algorithms compared: 1) the SCPG and 2) a RANDOM algorithm.
2) In comparison with the DE and standard PSO 2011 algorithms, our PSO also showed a better performance.
3) In the scope of the scenario instances studied here, we can claim that our PSO scales adequately in terms of the number of traffic lights. Concerning an increase in the number of vehicles, we have characterized how the scenario topology can influence the scalability power of our proposal, showing accurate results especially with regular route designs.
4) The complete optimization process required a computational mean time in the range from 6.33 to 26.5 min, which is completely acceptable for use by human experts in civil engineering. Furthermore, these values suggest that we can still work with larger scenario instances in future experiments.
5) The final solutions obtained by our PSO can improve the number of vehicles that reach their destination and the mean journey time, for all the instances. In particular, for the Alameda Avenue instance with 30 traffic logics and 300 vehicles, the improvement obtained is around 31.66% in the number of completed journeys and 74% in the journey time, regarding SCPG. All this means a real improvement in city traffic.

In future work, we will be tackling the optimal cycle program with other optimization techniques, and in particular other metaheuristics. We are also interested in using other traffic simulators and creating new larger dimension instances, as close as possible to real scenarios of an entire city.

REFERENCES

[1] J. McCrea and S. Moutari, "A hybrid macroscopic-based model for traffic flow in road networks," *Eur. J. Oper. Res.*, vol. 207, no. 1, pp. 676–684, 2010.

[2] J. Sánchez-Medina, M. Galán, and E. Rubio, "Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 25–40, Feb. 2008.

[3] J. C. Spall and D. C. Chin, "Traffic-responsive signal timing for system-wide traffic control," *Transp. Res. C, Emerging Technol.*, vol. 5, nos. 3–4, pp. 153–163, 1997.

[4] R. Bretherton, N. Hounsell, and B. Radia. (1996). Public transport priority in SCOOT [Online]. Available: http://eprints.soton.ac.uk/75299/

[5] D. A. Hensher and K. J. Button, *Handbook of Transport Systems and Traffic Control*. Amsterdam, The Netherlands: Elsevier, 2001.

[6] D. Zhao, Y. Dai, and Z. Zhang, "Computational intelligence in urban traffic signal control: A survey," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 4, pp. 485–494, Jul. 2012.

[7] E. Bingham, "Reinforcement learning in neurofuzzy traffic signal control," *Eur. J. Oper. Res.*, vol. 131, no. 2, pp. 232–241, 2001.

[8] K. N. Hewage and J. Y. Ruwanpura, "Optimization of traffic signal light timing using simulation," in *Proc. 36th Winter Simul. Conf.*, Dec. 2004, pp. 1428–1436.

[9] C. Karakuzu and O. Demirci, "Fuzzy logic based smart traffic light simulator design and hardware implementation," *Appl. Soft Comput.*, vol. 10, no. 1, pp. 66–73, 2010.

[10] G. Lim, J. J. Kang, and Y. Hong, "The optimization of traffic signal light using artificial intelligence," in *Proc. 10th IEEE Int. Conf. Fuzzy Syst.*, Dec. 2001, pp. 1279–1282.

[11] C. Tolba, D. Lefebvre, P. Thomas, and A. E. Moudni, "Continuous and timed Petri nets for the macroscopic and microscopic traffic flow modelling," *Simul. Model. Practice Theory*, vol. 13, no. 5, pp. 407–436, 2005.

[12] T. Nagatani, "Effect of speed fluctuation on green-light path in 2D traffic network controlled by signals," *Phys. A, Statist. Mech. Appl.*, vol. 389, no. 19, pp. 4105–4115, 2010.

[13] E. Angulo, F. P. Romero, R. García, J. Serrano-Guerrero, and J. A. Olivas, "A methodology for the automatic regulation of intersections in real time using soft-computing techniques," in *Modelling, Computation and Optimization in Information Systems and Management Sciences*. Berlin, Germany: Springer, 2008, pp. 379–388.

[14] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, "Optimizing traffic lights in a cellular automaton model for city traffic," *Phys. Rev. E*, vol. 64, no. 5, pp. 056132-1–056132-12, Oct 2001.

[15] M. Kutz, *Handbook of Transportation Engineering*. New York, NY, USA: McGraw-Hill, 2004.

[16] M. Clerc (2011). Standard PSO 2011, Particle Swarm Central [Online]. Available: http://www.particleswarm.info

[17] M. A. M. de Oca, T. Stützle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: A composite particle swarm optimization algorithm," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1120–1132, Oct. 2009.

[18] J. Kennedy and R. C. Eberhart, *Swarm Intelligence*. San Francisco, CA, USA: Morgan Kaufmann, 2001.

[19] D. Krajzewicz, M. Bonert, and P. Wagner, "The open source traffic simulation package SUMO," in *Proc. RoboCup Infrastruct. Simul. Competition*, 2006, pp. 1–5.

[20] M. Clerc and J. Kennedy, "The particle swarm—Explosion, stability, and convergence in a multidimensional complex space," *IEEE Trans. Evol. Comput.*, vol. 6, no. 1, pp. 58–73, Feb. 2002.

[21] S. Dipti, C. Min, and L. Ruey, "Neural networks for real-time traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 3, pp. 261–272, Sep. 2006.

[22] B. P. Gokulan and D. Srinivasan, "Distributed geometric fuzzy multiagent urban traffic signal control," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 714–727, Sep. 2010.

[23] K. Wood, "Urban traffic control, systems review," Transport Research Laboratory, Crowthorne, U.K., Tech. Rep. PR41, 1993.

[24] S. M. Rahman and N. T. Ratrout, "Review of the fuzzy logic based approach in traffic signal control: Prospects in Saudi Arabia," *J. Transp. Syst. Eng. Inf. Technol.*, vol. 9, no. 5, pp. 58–70, 2009.

[25] S. Lämmer and D. Helbing, "Self-control of traffic lights and vehicle flows in urban road networks," *J. Statist. Mech.: Theory Exp.*, vol. 2008, no. 4, p. P04019, 2008.

[26] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surveys*, vol. 35, no. 3, pp. 268–308, 2003.

[27] N. M. Rouphail, B. B. Park, and J. Sacks, "Direct signal timing optimization: Strategy development and results," in *Proc. XI Pan American Conf. Traffic Transp. Eng.*, 2000.

[28] P. Holm, D. Tomich, J. Sloboden, and C. Lowrance, "Traffic analysis toolbox volume IV: Guidelines for applying CORSIM microsimulation modeling software," Nat. Tech. Information Service, Springfield, VA, USA, Tech. Rep. FHWAHOP-07-079, 2007.

[29] F. Teklu, A. Sumalee, and D. Watling, "A genetic algorithm approach for optimizing traffic control signals considering routing," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 22, no. 1, pp. 31–43, 2007.

[30] D. Van Vliet, "SATURN–A modern assignment model," *Traffic Eng. Control*, vol. 23, no. 12, pp. 578–581, 1982.

[31] A. M. Turky, M. S. Ahmad, M. Z. Yusoff, and B. T. Hammad, "Using genetic algorithm for traffic light control system with a pedestrian crossing," in *Proc. 4th Int. Conf. Rough Sets Knowl. Technol.*, 2009, pp. 512–519.

[32] J. Chen and L. Xu, "Road-junction traffic signal timing optimization by an adaptive particle swarm algorithm," in *Proc. 9th Int. Conf. Control, Autom., Robot. Vis.*, Dec. 2006, pp. 1–7.

[33] L. Peng, M.-H. Wang, J.-P. Du, and G. Luo, "Isolation niches particle swarm optimization applied to traffic lights controlling," in *Proc. 48th IEEE Conf. Decis. Control/28th Chin. Control Conf.* Dec. 2009, pp. 3318–3322.

[34] S. Kachroudi and N. Bhouri, "A multimodal traffic responsive strategy using particle swarm optimization," in *Proc. 12th Int. Feder. Autom. Control Symp. Transp. Syst.*, Sep. 2009, pp. 531–537.

[35] E. Alba, J. García-Nieto, J. Taheri, and A. Zomaya, “New research in nature inspired algorithms for mobility management in GSM networks,” in *Proc. EvoWorkshops*, 2008, pp. 1–10.

[36] J. García-Nieto and E. Alba, “Automatic parameter tuning with metaheuristics of the AODV routing protocol for vehicular ad-hoc networks,” in *Applications of Evolutionary Computation*, vol. 6025. Berlin, Germany: Springer, 2010, pp. 21–30.

[37] K. E. Parsopoulos and F. M. Vrahatis, “Unified particle swarm optimization for solving constrained engineering optimization problems,” in *Advances in Natural Computation*. Berlin, Germany: Springer, 2005, pp. 582–591.

[38] E. Alba, J. Garcia-Nieto, L. Jourdan, and E.-G. Talbi, “Gene selection in cancer classification using PSO/SVM and GA/SVM hybrid algorithms,” in *Proc. IEEE Congr. Evol. Comput.*, Sep. 2007, pp. 284–290.

[39] J. García-Nieto, J. Toutouh, and E. Alba, “Automatic tuning of communication protocols for vehicular ad hoc networks using metaheuristics,” in *Eng. Appl. Artif. Intell.*, vol. 23, no. 5, pp. 795–805, 2010.

[40] M. Haklay and P. Weber, “OpenStreetMap: User-generated street maps,” *IEEE Pervasive Comput.*, vol. 7, no. 4, pp. 12–18, Oct.–Dec. 2008.

[41] J. Leung, L. Kelly, and J. H. Anderson, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Boca Raton, FL, USA: CRC Press, 2004.

[42] S. Krauß, “Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics,” Ph.D. dissertation, Dept. Comput. Sci., Univ. Cologne, Cologne, Germany, 1998.

[43] N. Hansen, R. Ros, N. Mauny, M. Schoenauer, and A. Auger (2008). PSO facing non-separable and ill-conditioned problems [Online]. Available: http://hal.inria.fr/inria-00250078/en/

[44] J. Sánchez-Medina, M. Galán, M. Royo, and E. Rubio, “Stochastic vs deterministic traffic simulator. Comparative study for its use within a traffic light cycles optimization architecture,” in *Artificial Intelligence and Knowledge Engineering Applications: A Bioinspired Approach* (Lecture Notes in Computer Science Series), vol. 3562. Berlin, Germany: Springer, 2005, pp. 622–631.

[45] D. Krajzewicz, E. Brockfeld, J. Mikat, J. Ringel, C. Rössel, W. Tuchscheerer, P. Wagner, and R. Wösler. (2005). Simulation of modern traffic lights control systems using the open source traffic simulation SUMO [Online]. Available: http://elib.dlr.de/21012

[46] E. Alba, G. Luque, J. García-Nieto, G. Ordonez, and G. Leguizamón, “MALLBA: A software library to design efficient optimisation algorithms,” *Int. J. Innov. Comput. Appl.*, vol. 1, no. 1, pp. 74–85, 2007.

[47] D. Thain, T. Tannenbaum, and M. Livny, “Distributed computing in practice: The Condor experience,” *Concurrency—Practice Exp.*, vol. 17, nos. 2–4, pp. 323–356, 2005.

[48] R. Eberhart and Y. Shi, “Comparing inertia weights and constriction factors in particle swarm optimization,” in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Jul. 2000, pp. 84–88.

[49] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. London, U.K.: Springer-Verlag, 2005.

[50] R. Wilcox, *New Statistical Procedures for the Social Sciences*. Hillsdale, NJ, USA: Lawrence Erlbaum Associates, 1987.

[51] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*. London, U.K.: Chapman & Hall, 2007.