Applied Soft Computing Journal 78 (2019) 697–711

Contents lists available at ScienceDirect

Applied Soft Computing Journal

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

Reliable simulation-optimization of traffic lights in a real-world city

Javier Ferrer [a,*], Manuel López-Ibáñez [b], Enrique Alba [a]

[a] Dept. Lenguajes y Ciencias de la Computación, Universidad de Málaga, Andalucía Tech, Spain
[b] Alliance Manchester Business School, University of Manchester, UK

HIGHLIGHTS

- Adaptation of an elitist iterated racing with the goal of obtaining reliable traffic light programs.
- Thorough analysis of solutions generated by means of 5 algorithms: IRACE, a GA, a DE, a PSO and a RS.
- Use of a real-world case study which comes from sensors at the street level.
- Future algorithms should consider the reliability of solutions over multiple traffic scenarios.
- The hybridization of iterated racing strategies with evolutionary algorithms is one of the most promising directions.

ARTICLE INFO

Article history:
Received 20 June 2018
Received in revised form 5 March 2019
Accepted 6 March 2019
Available online 14 March 2019

Keywords:
Simulation
Optimization
Metaheuristics
Uncertainty
Traffic-light planning

ABSTRACT

In smart cities, when the real-time control of traffic lights is not possible, the global optimization of traffic-light programs (TLPs) requires the simulation of a traffic scenario (traffic flows across the whole city) that is estimated after collecting data from sensors at the street level. However, the highly dynamic traffic of a city means that no single traffic scenario is a precise representation of the real system, and the fitness of any candidate solution (traffic-light program) will vary when deployed on the city. Thus, ideal TLPs should not only have an optimized fitness, but also a high reliability, i.e., low fitness variance, against the uncertainties of the real-world. Earlier traffic-light optimization methods, e.g., based on genetic algorithms, often simulate a single traffic scenario, which neglects variance in the real-world, leading to TLPs not optimized for reliability.

Our main contributions in this work are the following: (a) the analysis of the importance of reliable solutions for TLP optimization, even when all traffic scenarios are consistent with the real-world data and highly correlated; (b) the adaptation of IRACE, an iterated racing algorithm that is able to dynamically adjust the number of traffic scenarios required to evaluate the fitness of TLPs and their reliability; (c) the use of a large real-world case study for which real-time control is not possible and where data was obtained from sensors at the street level; and (d) a thorough analysis of solutions generated by means of IRACE, a Genetic Algorithm, a Differential Evolution, a Particle Swarm Optimization and a Random Search. This analysis shows that simple strategies that simulate multiple traffic scenarios are able to obtain optimized solutions with improved reliability; however, the best results are obtained by IRACE, among the algorithms evaluated.

1. Introduction

Complex real-world systems may be optimized by combining heuristic optimization algorithms, evolutionary or otherwise, with dynamic models [1] or simulations [2]. In such *simheuristics* [3], the quality (*fitness*) of a candidate solution is evaluated by means of simulation. However, any particular simulation is not only an imperfect approximation of the real-world system, but also the state of the real-world system is often dynamic and never completely known before the solution is deployed on it [2]. A recurring question is how to address the *reliability* of the solutions generated, such that the variance of their fitness in the dynamic real-world system is as small as possible [4].

The ultimate goal of our research is to improve the traffic-light planning of the City of Málaga (Spain). As in many real-world cities, the real-time control of traffic-lights [5] is not feasible because of various reasons (legal, technical, etc.), and we must instead find a highly-reliable global schedule of traffic-lights that works well in the dynamic and uncertain traffic system [6–16]. When optimizing the light cycle programs of traffic signals within a city in order to improve traffic flow and reduce pollution,

* Corresponding author.
E-mail addresses: ferrer@lcc.uma.es (J. Ferrer), manuel.lopez-ibanez@manchester.ac.uk (M. López-Ibáñez), eat@lcc.uma.es (E. Alba).

https://doi.org/10.1016/j.asoc.2019.03.016
1568-4946/

the fitness of a candidate traffic-light program is evaluated by simulating vehicle routes and velocities over a given traffic network [8,9,11,15,17]. The simulation of traffic flows on a specific city requires collecting *network data* (topology of the area and information about traffic lights), which is usually precise and static, and *traffic data* (number of vehicles, their journeys and velocities), which is estimated from highly dynamic real-world data. Appropriate sensor placement is a fundamental requirement in the control of any junction [18], especially when traffic light recognition is needed for autonomous vehicles [19].

Given the inherent uncertainty of this estimation, it is possible to generate distinct traffic scenarios that are all consistent with the real-world system [13]. One way to take into account this uncertainty is to compute an aggregated fitness value by simulating the same candidate solution multiple times by using different traffic scenarios [7,20]. An alternative (or additional) way is to make each simulation itself stochastic by introducing random changes to the traffic scenario during simulation [10]. The fitness of a *reliable* traffic-light program should not present a high variance when evaluated on the dynamic traffic of the real city.

Following previous works [4], we distinguish between *reliability against noise* and *robustness*. The latter measures uncertainty caused by imprecision of the decision variables of the solution, which is not relevant for traffic-light optimization because solutions can always be implemented precisely. The former measures uncertainty in estimating fitness given a precise solution, which is highly relevant because the fitness of a given traffic-light program on the real-world system is never exactly the one obtained in simulation.

Despite the inherent uncertainty of simulated traffic scenarios, the literature on traffic-light optimization often relies on deterministic simulation on a single traffic scenario [8–10,16,17]. When multiple scenarios are considered, they are actually used for evaluating the flexibility of the optimization algorithm by optimizing each scenario separately [10]. More recently, Ferrer et al. [7] validated the reliability of candidate solutions on multiple scenarios after the optimization phase, however, each solution is still optimized with respect to a single scenario.

In this paper, we advocate the simulation of multiple traffic scenarios within traffic-light optimization to increase reliability, and we demonstrate that it is possible to generate highly optimized traffic-light programs with low variance across traffic scenarios derived from real-world data of the City of Málaga. Our scenarios are larger (58 intersections, 275 traffic-lights, 4827 vehicles) than other recent studies on traffic-light scheduling (40 intersections, 184 traffic-lights, 500 vehicles [9]) and much larger than those solved with real-time control strategies (16–30 traffic-lights, 800 vehicles [5]).

Moreover, even when the fitness of random candidate solutions is highly correlated for different scenarios generated from the same real-world data, we show that an optimized solution may have a high fitness variance across the scenarios. Motivated by this result, we compare several simulation strategies employing more than one traffic scenario per optimization run. In particular, we incorporate classical resampling and iterative resampling strategies [4] into a differential evolution (DE), genetic algorithm (GA), particle swarm optimization (PSO), and a random search (RS), the latter used as a baseline. Our results demonstrate that resampling strategies using more than one traffic scenario significantly improve the reliability of the generated solutions.

Finally, we evaluate a fifth approach, namely, *elitist iterated racing* [21], as implemented by the irace software, with the goal of obtaining highly optimized and reliable traffic-light programs. irace combines heuristic optimization and a racing method, based on F-race [22], to iteratively optimize candidate solutions for noisy black-box problems. In irace, subsequent races re-use information from previous races and the best (elites) solutions found can only be replaced by strictly better solutions. Although racing is far less studied than other methods for handling uncertainty in the evolutionary optimization literature [4], its use is becoming increasingly widespread [23–27]. Our results show that irace is able to obtain optimized traffic-light programs with much lower fitness and variance than those obtained by the GA with any of the resampling strategies.

The paper is structured as follows. In Section 2, we define the problem of optimizing the phases of traffic lights. Section 3 describes the traffic simulator SUMO and the algorithms compared in this paper. Our experimental setup is presented in Section 4, where we describe the real-world case study that motivated this research, the simulation strategies tested and other experimental details. Results are analyzed in Section 5 and we present our conclusions in Section 6.

2. Description of the problem

In a metropolitan area, traffic jams and congestion are one of the biggest sources of greenhouse gas emissions, and therefore contributors to global climate change. Urban traffic planning is a way to improve mobility efficiency and safety, thus producing a positive impact on the traffic flow. Particularly, the computation of optimal traffic light plans is mandatory, specially for large urban areas, which motivates the Traffic Light Scheduling Problem (TLSP) [9–11]. The goal in the TLSP is to find an optimized traffic light program (TLP), that is, a combination of phase durations for all traffic lights, in all intersections of a given urban area, with the aim of improving the global flow of vehicles by reducing the global journey time, emissions, and fuel consumption.

Fig. 1 illustrates an intersection showing one phase out of a cycle of six valid phases, where each phase has eleven signals (colors). The phase shown in the figure contains the state *"rr ggy rr yggg"* meaning that five traffic lights are green (g), two are yellow (y), and the other four are red (r). The yellow signals (y) indicate that each driver must prepare to stop if necessary to let a driver on another direction proceed. The next phase changes the state of the traffic lights to another valid combination, for example, *"rr yyg rr gyyy"* where the signals that were green turn into yellow, and the signals that were yellow turn into green. The last phase is followed by the first one, and this cycle is repeated for the entire simulation. All the intersections in the complete scenario perform their own cycles of phases at the same time, thereby comprising the global schedule of signal lights.

A formal definition of the TLSP is as follows. Let $P = \{I_1, \ldots, I_n\}$ be a candidate TLP, where each I_i corresponds to a different intersection defined as a set of phases $I_i = \{\varphi_{i1}, \ldots, \varphi_{im_i}\}$, where $m_i = |I_i|$ and each φ_{ij} represents the duration of phase j in intersection I_i, that is, the duration of each valid phase of light colors (e.g., "*rr yyg rr gyyy*"). Typically, durations are defined as integral seconds, that is, $\varphi_{ij} \in \mathbb{N}^+$, as done in real traffic lights. The objective is to find a TLP P' that minimizes a fitness function $f: \Gamma \to \mathbb{R}$ such that:

$$P' = \operatorname*{argmin}_{P \in \Gamma} \{f(P)\} \tag{1}$$

where Γ is the space of all possible TLPs. The fitness function used in this work is explained next.

2.1. Fitness function

The evaluation of a candidate solution to the TLSP, that is, of a TLP $P \in \Gamma$, is typically performed by means of a traffic simulator that provides information regarding the flow of vehicles. Vehicles travel from a starting position (a point in the city map or network) to a destination position. The simulation time t^{sim} defines the

Fig. 1. Example of one phase of traffic lights within an intersection. Clock-wise from left-most: "*rr ggy rr yggg*".

time steps available for the vehicles to complete their journeys. The travel time (t_v) of a vehicle v is the number of simulation steps (1 s per simulation step) in which its speed was above 0.1 m/s, while its waiting time (w_v) is the number of simulation steps in which its speed was below 0.1 m/s.

An optimal TLP should maximize the number of vehicles that reach their destination $V(P)$ during the simulation time t^{sim} and, in consequence, minimize the vehicles that remain circulating $V^{\text{rem}}(P)$ after the simulation time ends. Another desirable behavior for a TLP is the minimization of the travel time t_v and waiting time w_v for the $V(P)$ vehicles that complete their journey.

Moreover, an optimal TLP should avoid long phase durations (φ_{ij}) because the phase duration determines the length of time that all traffic lights within the same intersection maintain their current state, which typically contains both green and red signals. Usually, a traffic light in green signal regulating one direction means a red signal in other directions. Thus, a long phase duration may lead to a collapse of the intersection due to traffic accumulating in all directions but one. Besides, a good TLP should also prioritize those phases with more traffic lights in green on the directions with a high number of vehicles circulating, and traffic lights in red on the directions with a low number of vehicles. We capture all these preferences in the following ratio measure that should be maximized:

$$GR(P) = \sum_{i=1}^{n} \sum_{j=1}^{|I_i|} \varphi_{ij} \cdot \frac{G_{ij}}{R_{ij}} \tag{2}$$

where G_{ij} is the number of traffic lights in green, and R_{ij} is the number of traffic lights in red in phase j of intersection i and φ_{ij} is the duration of the phase. The minimum value of R_{ij} is 1 in order to avoid division by 0.

Given the above criteria, we formulate the following fitness function that should be minimized:

$$f(P) = \frac{V^{\text{rem}}(P) \cdot t^{\text{sim}} + \sum_{v=1}^{V(P)} t_v(P) + w_v(P)}{V(P)^2 + GR(P)} \tag{3}$$

where the presence of vehicles with incomplete journeys $V^{\text{rem}}(P)$ penalizes the fitness of a candidate solution P proportionally to the simulation time t^{sim} and the number of vehicles that arrive at their destinations is squared ($V(P)^2$) in order to prioritize this criterion over the rest. This fitness function has been successfully used in previous works [8,9].

2.2. Constraints

Real-world instances of the TLSP often present additional constraints. In our case, we consider the following constraints recommended by the Mobility Department of the City Council of Málaga (Spain).

Phases containing any yellow signals are called *fixed phases* because they have a predetermined duration and the set of such phases will be denoted by Y. There are two types of fixed phases: (1) phases not corresponding to a pedestrian cross, which last for a fixed time of 4 s, and (2) phases corresponding to a pedestrian cross, which last for a fixed time of $4 \times$ *number of lanes* seconds. Non-fixed phases have a minimum duration of $\varphi_{\min} = 15$ s. Moreover, the total cycle time (Tp_i) within each intersection I_i, which is computed as the sum of its phase durations:

$$Tp_i = \sum_{\varphi_{ij} \in I_i, j=1}^{|I_i|} \varphi_{ij} \tag{4}$$

is constrained within $[Tp_{\min}, Tp_{\max}]$. For the City Council of Málaga (Spain), $Tp_{\min} = 60$ and $Tp_{\max} = 120$ s.

By default, the first cycles of all intersections start at the same time. However, we also optimize an offset time at each intersection (To_i) that represents a shift in seconds of the starting time of the cycle at the start of the simulation. If the offset value of an intersection is negative, then cycle start time is shifted back that number of seconds and the cycle actually starts on a phase before the first one; whereas if the offset is positive, the cycle begins as if that number of seconds has already passed, i.e., skipping those seconds from the duration of the first phase and, maybe, of later phases. Offset times enable the emergence of *green waves*, a series of coordinated traffic lights that produce a continuous traffic flow over several intersections in one main direction. Offset values are constrained within the time interval $To_i \in [To_{\min}, To_{\max}] = [-30, 30]$.

2.3. Solution representation

In this work, each candidate solution to the TLSP is encoded as a vector of integers, where values represent either the offset of each intersection or the duration of a phase, adjusting the minimum and maximum possible ranges given the constraints described before. A repair mechanism, described later in Section 2.4, is applied to enforce the total cycle time constraints (Eq. (4)). Fig. 2 shows an example of the solution representation.

Taking into account that the duration of each phase (before repair) and the offset values are constrained by $\varphi_j \in [\varphi_{\min}, Tp_{\max}] \subset$

$\mathbb{N}^+$ and $To_i \in [To_{\min}, To_{\max}] \subset \mathbb{Z}$, respectively; then, given a problem instance with n intersections and a total of $M = \sum_{i=1}^{n} |I_i|$ non-fixed phases, the size of the solution space Γ can be estimated as $|Tp_{\max} - \varphi_{\min} + 1|^M + |To_{\max} - To_{\min} + 1|^n$. In the network model of the city of Málaga studied here, we have 58 intersections and a total of 198 non-fixed phases ($\varphi_{\min} = 15$, $Tp_{\max} = 120$, $To_{\min} = -30$, $To_{\max} = 30$), thus the solution space contains $106^{198} + 61^{58} \approx 1.025 \cdot 10^{401}$ possible TLPs, that is, candidate solutions. Therefore, efficient optimization algorithms are required to tackle this problem.

2.4. Repair procedure

We propose the following repair procedure to ensure that candidate solutions are valid. The value of each phase duration φ_{ij} is already constrained within a range that is larger than the minimum phase duration ($\varphi_{\min}$). However, we need to ensure that the total cycle time Tp_i is within $[Tp_{\min}, Tp_{\max}]$. Here we can distinguish two different cases.

In the first case, if the total cycle time for intersection I_i is smaller than $Tp_{\min}$, then we replace each non-fixed phase (those that do not contain a yellow signal, i.e., $\varphi_{ij} \notin Y$) with

$$\varphi_{ij} = \left\lceil \varphi_{ij} \cdot \frac{Tp_{\min} - Tp_i^Y}{Tp_i - Tp_i^Y} \right\rceil \tag{5}$$

where $Tp_i^Y = \sum_{\varphi_{ij} \in I_i \cap Y} \varphi_{ij}$ is the sum of the fixed phase durations within intersection I_i.

For example, let us imagine an intersection with a pedestrian crossing composed of the following phases: "*gg*" and "*yy*" with durations 40 and 8, respectively. Since the sum of the complete cycle is $Tp_i = 48$, which is less than $Tp_{\min} = 60$, we need to repair the decision variables of this intersection. The phase "*yy*" has a fixed duration of 8. Thus, we apply Eq. (5) to the duration of the phase "*gg*" as follows

$$\varphi_{\text{“}gg\text{”}} = \left\lceil \frac{40 \cdot (60 - 8)}{48 - 8} \right\rceil = 52$$

In the second case, if the total cycle time is larger than $Tp_{\max}$, then we replace each non-fixed phase ($\varphi_{ij} \notin Y$) with

$$\varphi_{ij} = \varphi_{\min} + \left\lfloor (\varphi_{ij} - \varphi_{\min}) \cdot \frac{Tp_{\max} - Tp_i^Y - \varphi_{\min} \cdot |I_i \setminus Y|}{Tp_i - Tp_i^Y - \varphi_{\min} \cdot |I_i \setminus Y|} \right\rfloor \tag{6}$$

where $|I_i \setminus Y|$ is the number of non-fixed phases within intersection I_i and Tp_i^Y is the total duration of the fixed phases (those that contain a yellow signal) within intersection I_i.

Following the example above, let us assume the duration of phase "*gg*" becomes 120. The sum of the total cycle becomes $Tp_i = 128$, that is, larger than $Tp_{\max} = 120$, hence it needs to be repaired using Eq. (6). The repaired duration of phase "*gg*" is calculated as follows

$$\varphi_{\text{“}gg\text{”}} = 15 + \left\lfloor \frac{(120 - 15) \cdot (120 - (15 + 8))}{128 - (15 + 8)} \right\rfloor = 15 + 97 = 112 \tag{7}$$

3. Algorithms

In this section we present different approaches to tackle the TLSP. First, we introduce the SUMO simulator and the SUMO Cycle Program Generator (SCPG) algorithm provided with the SUMO package. Second, we describe IRACE and explain how we adapt the elitist IRACE to the TLSP. Third, we describe a genetic algorithm that has shown to be very effective in optimizing the traffic system in various applications. We also describe the adaptation of differential evolution and particle swarm optimization to this problem, for the sake of comparison. Finally, we describe a basic random search as a baseline optimizer for our comparison.

3.1. SUMO simulator and SCPG algorithm

SUMO (Simulator of Urban Mobility) [28,29] is a microscopic road traffic simulator that provides detailed information about vehicles' velocity, fuel consumption, emissions, journey time, etc., enabling the study of realistic scenarios according to real patterns of mobility of the target city. We need a microscopic model and simulator due to the need for fine-grained realistic simulations.

SUMO requires several input files that describe the static *network data*, that is, the topology of the area to be simulated and the valid combinations of phases of the traffic lights; the dynamic *traffic scenario*, that is, the number of vehicles, journeys, traffic flow and vehicle characteristics; and a complete traffic light program (TLP). When the simulation ends, SUMO returns information about vehicles journeys (each vehicle's departure and arrival time, its travel time, and its time waiting in traffic lights) that may be used to evaluate the quality of a traffic light program. SUMO also provides estimates of emission traces in vehicles and fuel consumption.

The SUMO package [28] includes a greedy deterministic algorithm based on human expert rules and knowledge called SCPG. The SCPG algorithm generates a complete TLP by assigning values to the phase durations according to three different factors: the proportion of green signals in the phases, the number of incoming lanes into the intersection, and the braking time of the vehicles approaching the traffic lights.

For the traffic scenarios used in our experiments, the solutions returned by the SCPG algorithm satisfy the constraints on total cycle time, but they sometimes violate the minimum phase duration, thus we apply the repair procedure explained in Section 2.4 to make the solutions feasible.

3.2. IRACE

We have adapted elitist IRACE [21] to solve the TLSP. Originally, IRACE was designed for automatically configuring the parameters of optimization algorithms over a number of training instances of a problem of interest. However, IRACE is essentially an optimization method for mixed-variable black-box noisy problems where solution fitness varies according to a known blocking factor (different training instances, traffic scenarios, etc.) that induces between-blocks variance, i.e., variance caused by evaluating the same solution on different blocks. In addition, fitness may be affected by unspecified within-blocks variance, i.e., variance from re-evaluating the same solution within the same block.

When applying IRACE to the TLSP, IRACE searches for an optimal traffic light program (TLP) for a given network file over a training set of traffic scenarios. The fitness of each candidate TLP, i.e., a potential solution to the TLSP, is calculated by simulating it, by means of SUMO, on the given network file together with one of the traffic scenarios. Evaluating the same solution on the same network data with a different traffic scenario will likely lead to a slightly different fitness. Reliable candidate TLPs should have small fitness variance between different traffic scenarios.

An outline of IRACE is given in Algorithm 1. A run of IRACE starts by creating an initial population of candidate solutions uniformly randomly (line 2). The size of this initial population (Θ_1) is by default dynamically computed, according to the number of decision variables and the computational budget (*totalEvals*). Then, a race is performed (line 3). Within a race (Fig. 3), each candidate solution θ_i is evaluated a number of times. Each evaluation requires a simulation on a different traffic scenario (S_k). By default, each candidate solution is evaluated on at least T^{first} traffic scenarios, then a statistical test (the Friedman test) is performed in order to identify candidate solutions that should be discarded because they are statistically worse than the best solution so far. After

Intersection 1					...	Intersection n				
To_1	φ_{11}	φ_{12}	...	φ_{1j}	...	To_n	φ_{n1}	φ_{n2}	...	φ_{nj}
-25	20	25	...	35	...	15	60	8	...	30

Fig. 2. Example of solution representation in the TLPS.

the first test, the race continues by evaluating surviving solutions on additional traffic scenarios and performing again the test. The race stops when a minimum number of solutions remain alive or a maximum number of simulations for this race have been performed.

Algorithm 1 Pseudocode of IRACE

Input: Network data and training traffic scenarios.
Output: Best solution (TLP) found.

1: $t \leftarrow 1$
2: $\Theta_t \leftarrow$ CreateRandomPopulation()
3: $\Theta^{\text{elite}} \leftarrow$ Race(Θ_t)
4: **while** *evals* < *totalEvals* **do**
5: $t \leftarrow t + 1$
6: $\mathcal{M} \leftarrow$ Update(Θ^{elite})
7: $\Theta^{\text{new}} \leftarrow$ Sample($\mathcal{M}$)
8: $\Theta_t \leftarrow \Theta^{\text{new}} \cup \Theta^{\text{elite}}$
9: $\Theta^{\text{elite}} \leftarrow$ Race(Θ_t)
10: **end while**
11: **Output:** best solution from Θ^{elite}

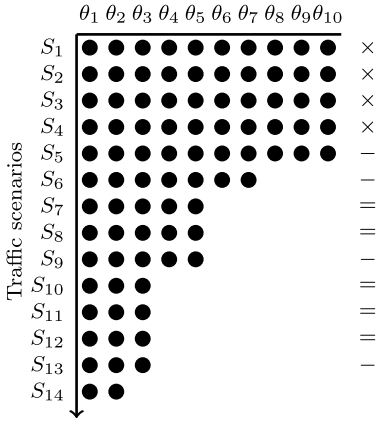

Fig. 3. Illustration of racing for the TLSP. Each node is the simulation of one candidate solution (TLP) θ_i on one training traffic scenario S_k. '×' means that no statistical test is performed, '–' means that the test discarded at least one candidate solution, '=' means that the test did not discard any candidate solution. In this example, $T^{\text{first}} = 5$.

After the race (line 3 in Algorithm 1), the best surviving solutions become the elite population (Θ^{elite}). These elite solutions are used by IRACE to update a sampling distribution $\mathcal{M}$, which is used to create new candidate solutions (line 7). In particular, the value of each decision variable for each new solution is sampled from a Gaussian distribution with mean equal to the value that decision variable has on an elite solution and a variance that decreases with the number of iterations. More details about this sampling can be found in [21]. The number of solutions sampled is dynamically decided by IRACE depending on the number of evaluations left until *totalEvals* and the number of races performed so far. The union of the new solutions and the elite population forms a new population that is subjected again to racing (line 9). Sampling and racing are iterated until reaching a maximum number of evaluations (*totalEvals*), measured as number of calls to the SUMO simulator.

In the elitist version of IRACE, an elite solution cannot be discarded until all competing solution have been evaluated on as many traffic scenarios as the elite one. In addition, information is shared across successive races so that solutions are never re-evaluated on the same traffic scenario. Finally,the sequence of traffic scenarios is reshuffled at each race in order to minimize the potential bias of a particular order of evaluation. More details about elitist IRACE are provided in the original publication [21].

Adapting IRACE to the TLSP required adding the ability of repairing candidate solutions, as explained in Section 2.4. This modified IRACE is publicly available starting from version 2.3. In addition, the input to IRACE is generated by parsing the SUMO input files and the values defining the various constraints of the particular TLSP instance tackled here.

3.3. Genetic algorithm

Genetic algorithms [30] have shown to be very effective in optimizing the traffic system in different ways: for a single intersection [14], for optimizing the traffic flow control [15], or for validating traffic light programs in different scenarios [7].

Algorithm 2 sketches the pseudocode of our GA for computing an optimized TLP. It receives as inputs the SUMO input files for the instance under study, that is, the network data and a set of training traffic scenarios. How and how many of these traffic scenarios are used to evaluate the fitness of a candidate solution defined by a simulation strategy. In this paper, we will compare several possible simulation strategies that are described later in Section 4.2. Initially, the algorithm creates a uniformly random population of individuals (line 2) and enters an inner loop that applies the traditional steps of a generational GA (lines 5–17). That is, some solutions (TLPs) are selected from the population Θ_t, recombined and mutated. If a generated offspring solution is not a feasible TLP (i.e., it violates a constraint of phase or cycle duration), then it is transformed into a valid TLP by applying the Repair operation (line 11) described in Section 2.4. Finally, the offspring solutions are evaluated according to the simulation strategy, and added to the new population Θ^{new}. The algorithm continues until reaching a maximum number of evaluations (*totalEvals*), as in the case of IRACE.

Our implementation of the GA uses a ranking method for parent selection and elitist replacement for the next population, that is, the two best individual of the current population are included in the next one. The operators used are uniform crossover and integer polynomial mutation.

3.4. Differential evolution

Differential evolution is an population-based evolutionary algorithms that we have adapted in order to find optimal (or quasi-optimal) cycle programs for traffic lights. In Algorithm 3 we show the pseudocode of the DE algorithm. Since GA and DE are evolutionary algorithms, their pseudocode is quite similar, except for the absence of a mutation operator and the use of a special crossover we briefly describe here.

Once the algorithm has selected three mutually different individuals from the population, a new individual is generated by the

Algorithm 2 Pseudocode of GA

Input: Network data and training traffic scenarios.
Output: Best solution (TLP) found.

$t \leftarrow 1$
$\Theta_t \leftarrow$CreateRandomPopulation()
Repair(Θ_t)
EvaluateFitness(Θ_t)
while *evals* $<$ *totalEvals* **do**
  $\Theta^{\text{new}} \leftarrow \emptyset$ // auxiliary population
  for $i \leftarrow 1$ **to** *populationSize*/2 **do**
    $\Theta^{\text{parents}} \leftarrow$ Selection(Θ_t)
    $\Theta^{\text{offspring}} \leftarrow$ Recombination(Θ^{parents})
    $\Theta^{\text{offspring}} \leftarrow$ Mutation($\Theta^{\text{offspring}}$)
    Repair($\Theta^{\text{offspring}}$)
    EvaluateFitness($\Theta^{\text{offspring}}$)
    $\Theta^{\text{new}} \leftarrow \Theta^{\text{new}} \cup \Theta^{\text{offspring}}$
  end for
  $\Theta_{t+1} \leftarrow$ Replace(Θ_t, Θ^{new})
  $t \leftarrow t + 1$
end while

following equation:

$$w_{i+1}^{z} = v_i^{r1} + F \cdot (v_i^{r2} - v_i^{r3}) \tag{8}$$

where $r1, r2, r3 \in (1, 2, \ldots, z-1, z+1, \ldots,$ *populationSize*$)$ are random mutually different integers, which are also different from the index z. The parameter $F > 0$ stands for the amplification of the difference between the individuals v_i^{r2}, v_i^{r3}. This operator is applied according to the crossover probability.

Algorithm 3 Pseudocode of DE

Input: Network data and training traffic scenarios.
Output: Best solution (TLP) found.

$t \leftarrow 1$
$\Theta_t \leftarrow$CreateRandomPopulation()
Repair(Θ_t)
EvaluateFitness(Θ_t)
while *evals* $<$ *totalEvals* **do**
  $\Theta^{\text{new}} \leftarrow \emptyset$ // auxiliary population
  for $i \leftarrow 1$ **to** *populationSize* **do**
    $\Theta^{\text{parents}} \leftarrow$ Selection(Θ_t)
    $\Theta^{\text{child}} \leftarrow$ DE-Crossover(Θ^{parents})
    Repair(Θ^{child})
    EvaluateFitness(Θ^{child})
    $\Theta^{\text{new}} \leftarrow \Theta^{\text{new}} \cup \Theta^{\text{child}}$
  end for
  $\Theta_{t+1} \leftarrow$ Replace(Θ_t, Θ^{new})
  $t \leftarrow t + 1$
end while

3.5. Particle swarm optimization

PSO [31] is a population-based metaheuristic with many different variants. We use here the Standard PSO 2011 [32], which has been used previously to find optimal (or quasi-optimal) cycle programs for traffic lights [33]. The pseudo-code of PSO used in this experimentation is introduced in Algorithm 4. The algorithm starts by initializing the swarm, which includes both the positions and velocities of the particles p_i (candidate solutions). Positions correspond to locations in the decision space and velocities are used to modify positions in future iterations.

The position of each particle is randomly initialized, and the leader is computed as the best particle of the swarm. Then, for a maximum number of evaluations, the velocity of each particle is updated. After that, the position is updated according to the velocities, and it is evaluated according to the strategy used. At the end of each iteration, the leader of the swarm is also updated. Finally, the best solution found so far is returned. For further details, we refer to the original publications [31,32].

Algorithm 4 Pseudocode of PSO

Input: Network data and training traffic scenarios.
Output: Best solution (TLP) found.

swarm $\leftarrow$ **initializeSwarm**()
Repair(swarm)
EvaluateFitness(swarm)
updateLeaders(swarm)
while *evals* $<$ totalEvals **do**
  for $i \leftarrow 1$ **to** *SwarmSize*() **do**
    updateVelocity(p_i)
    updatePosition(p_i)
    Repair(p_i)
    EvaluateFitness(p_i)
  end for
  updateLeaders(swarm)
end while

3.6. Random search

We also implemented a basic random search as a baseline for evaluating IRACE and the GA. The pseudocode of the random search (RS) algorithm is presented in Algorithm 5.

The RS receives the same input files as IRACE and the GA. At each iteration of the main loop of the algorithm, a new candidate solution (TLP) is generated by randomly sampling the phase durations φ_{ij} from the discrete uniform distribution $\mathcal{U}\{\varphi_{\min}, \ldots, Tp_{\max}\}$ and the offset values To_i from $\mathcal{U}\{To_{\min}, \ldots, To_{\max}\}$. The resulting solution may be repaired (Section 2.4) to ensure that total cycle times Tp_i are within $[Tp_{\min}, Tp_{\max}]$. As in the case of the GA, the fitness of the new solution is then evaluated according to a simulation strategy by performing one or several simulations using different traffic scenarios. The possible simulation strategies are explained later in Section 4.2. Finally, the new solution replaces the best-so-far solution if the former has better fitness than the latter (lower in this case, because Eq. (3) should be minimized). Like the other algorithms, RS stops when reaching a maximum number of simulations (*totalEvals*).

Algorithm 5 Pseudocode of RS

Input: Network data and training traffic scenarios.
Output: Best solution (TLP) found.

bestSolution $\leftarrow\emptyset$
while *evals* $<$ *totalEvals* **do**
  newSolution $\leftarrow$ CreateRandomSolution()
  Repair(newSolution)
  EvaluateFitness(newSolution)
  bestSolution $\leftarrow$ Replace(bestSolution, newSolution)
end while

4. Experimental setup

In this section, we describe the experimental protocol followed in the rest of the paper. First, we describe the real-world case study of TLSP optimization that is the main motivation of our research. Next, we introduce the various simulation strategies analyzed in our experiments. Finally, we provide details about the experiments carried out, which are analyzed in the next section.

4.1. Real-world case study

We consider a realistic scenario derived from the traffic network of Málaga [13]. The selected urban area is delimited to the north by San Bartolomé Street and Ferrándiz Street, to the west by the Guadalmedina River, to the east by Keromnes Street, and to the south by the Mediterranean Sea, which encompasses an area of about 3 km^2 with 58 intersections controlled by 275 traffic lights (Fig. 4). This urban area encompasses several traffic zones whose signals are currently scheduled independently, but we will optimize simultaneously. Our network model was created from real data about traffic rules, traffic element locations, buildings, road directions, streets, intersections, etc.

The static network of a city is not enough to define a realistic scenario and we precise information about the dynamic traffic data. Hence, we defined the number of vehicles circulating and their speeds by following current specifications available from the Mobility Department of the Málaga's City Council. This information was collected from sensorized points in certain streets measuring traffic density at various time intervals. Fig. 5 shows the exact position of the sensorized points in the area under study. From the sensed data extracted, we have applied the Flow Generator Algorithm (FGA) [13] to generate 60 different traffic scenarios with an average of 4827 vehicles (or different vehicle routes) per scenario. FGA is an evolutionary algorithm that creates vehicle routes minimizing the differences between the real data measured by the sensors and the values obtained after the simulation of the map. The resulting traffic scenarios are composed by different traffic flows with different number of vehicles passing through different streets, but with the particularity that the generated traffic is similar to the real traffic of a weekday.

In principle, any of the generated traffic scenarios is as realistic as the others, since all of them are generated from the same sensor data. Moreover, we must assume that the *real* traffic scenario for a given day in the future is unlikely to precisely match any of the traffic scenarios generated because of the implicit uncertainty of a real traffic system. Thus, good traffic-light programs should be reliable, i.e., they should work well despite small differences in traffic conditions, which are modeled by the different traffic scenarios. In order to evaluate the reliability of a candidate solution, we split the generated traffic scenarios into two equal sets of 30 scenarios each. One (training) set is exclusively used for optimization, that is, for identifying optimal TLSP solutions. The other (testing) set of scenarios is used for comparing the solutions found during optimization.

The number of traffic scenarios from the training set actually used during optimization (and in which manner) depends on a particular simulation strategy. In principle, evaluating a candidate solution over many traffic scenarios should lead to identifying reliable solutions. However, given a fixed budget of simulations, there is a trade-off between the number of different candidate solutions evaluated and the number of traffic scenarios simulated per solution. Different strategies distribute the effort devoted to evaluating solutions in different ways. Strategies that use fewer simulations to evaluate the fitness and reliability of each solution are able to evaluate a higher number of different candidate solutions. On the other hand, strategies that use more simulations per solution will explore fewer candidate solutions. In the following, we describe the various strategies compared in our study.

4.2. Simulation strategies

When dealing with real systems like traffic, it is not enough to assume that one single simulation could capture the main characteristics of the system under study. Therefore, several simulations with random variations are necessary to properly evaluate a solution. Reliable solutions, that is, those that result in similar fitness despite variations in traffic conditions, are highly desirable for the traffic system, which is considered to be critical due to its importance for the city operation [12].

We evaluate here the ability of racing, as implemented in IRACE, for dealing with this uncertainty, and, as a baseline for further comparison, we study here two additional strategies, which we call *All* and *Rand*.

4.2.1. Racing

The evaluation strategy of IRACE is specially designed for problems where at least part of the variation can be treated as blocks and candidate solutions can be compared within each block, which is sometimes called *blocking* or *pairing* in the design of experiments literature. A basic assumption in IRACE is that some variability between blocks is expected, i.e., the relative quality of a candidate solution with respect to other solutions depends on the actual block being evaluated; but the variability is not completely random, i.e., over the whole set of blocks, some candidate solutions are better than others. In the context of the TLSP, a block is characterized by simulating on a specific traffic scenario. It is not expected that a single TLSP solution will be better for all possible traffic scenarios, however, some TLSP solutions should have a better *expected* fitness over the whole set of typical traffic scenarios for the city of Málaga.

In order to account for this type of variance, IRACE simulates each candidate solution by means of racing on at least T^{first} different traffic scenarios, performs a statistical test to assess whether there is enough evidence to discard this candidate solution as worse than the best so far and, if not, performs additional simulations with new scenarios, applying the statistical test after each simulation. The motivation for this approach is that solutions that are clearly worse with low variance are discarded as quickly as possible, whereas additional simulations are performed only in cases where variance is higher and the best solution is not immediately clear. Moreover, in elitist IRACE, the best solution found can only be replaced by another candidate solution simulated on at least as many traffic scenarios, which avoids elite solutions being replaced by solutions with a few "lucky" simulations that actually are less reliable. The disadvantage of such an approach for problems with little or no variance is that, in the best case, IRACE requires T^{first} (by default five) times more simulations per candidate solution than a method not accounting for fitness variance.

4.2.2. All-N

The strategy *All-N* uses only N training traffic scenarios during each optimization run. Each solution is simulated on all N scenarios and its fitness is computed as the mean fitness over the N simulations. For example, the strategy *All-1* would use only one traffic scenario per run, i.e., one simulation per solution and always the same traffic scenario for every simulation. This is the strategy adopted for problems without uncertainty, where there is no noise due to simulation and no between-blocks variance.

4.2.3. Rand-N

The strategy *Rand-N* uses the whole training set of traffic scenarios during each optimization run, from which it randomly selects a subset of N scenarios at each iteration to evaluate solutions. The fitness of each solution is computed as the mean fitness over the N simulations. For example, the strategy *Rand-1* would randomly select one traffic scenario from the whole training set at the first iteration, and use that traffic scenario to compute the fitness of all solutions in the population; in the next iteration, the fitness of new solutions is computed by simulating them on new randomly selected scenario. This strategy aims at creating reliable

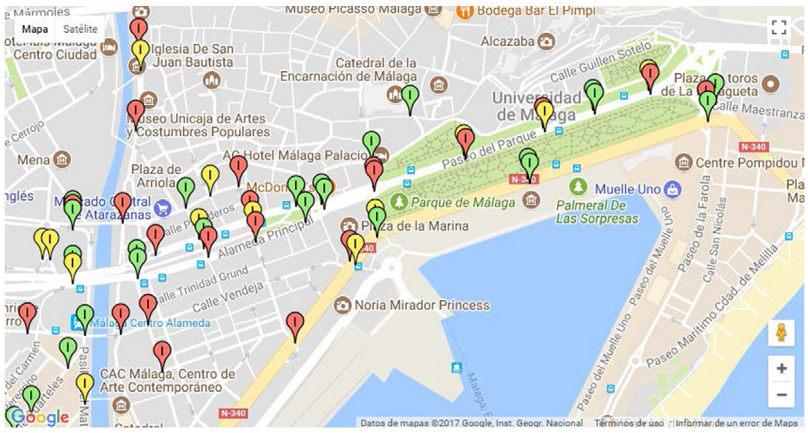

Fig. 4. Locations of traffic lights considered in the case of study. In this figure the colors indicate large (red), medium (yellow) and small (green) differences between two different solutions.

(a) Open Street Map (b) SUMO map

Fig. 5. Málaga case study: geographical area with sensor locations.

solutions that remain good over different traffic scenarios even for low values of N.

Increasing the value of N in either *All-N* or *Rand-N* strategy allows more precise estimation of the expected fitness of a solution over many traffic scenarios, hopefully leading to more reliable solutions. However, it also requires additional simulations, which are typically time-consuming. Given a fixed budget of simulations, the *Rand-N* strategy explores the same number of solutions as the *All-N* strategy, but larger values of N will reduce this number, possibly resulting in solutions with worse fitness being found.

4.3. Experimental details

As mentioned above, we generated 60 traffic scenarios from real sensor data and we split these scenarios into two sets of size 30. One set (training set) is used when running the optimizers (IRACE, GA, DE, PSO and RS) to find TLSP solutions, while the other set (testing set) is used for evaluating the fitness and reliability of these solutions and comparing the various strategies analyzed in this paper. During optimization, the traffic is simulated up to a predefined time horizon (1 h plus 10 min of warm-up, in our case) in order to simulate the peak period in our real-world case study.

When running the optimizers, we perform independent replications with different random seeds of each run of an algorithm. In Fig. 6, we show a summary of the experiments carried out in this article. Both IRACE and the *Rand-N* strategy may use the whole training set on each run, and we performed 30 independent replications of each of their runs. The *All-N* strategy, however, only uses a subset N of the training set and the actual subset may introduce a bias on the solutions found, e.g., if the subset contains a traffic scenario that is quite different from the ones in the test set. To take into account this potential bias, we repeat each run of the *All-N* strategy using a different subset of the training set. In particular, for *All-1*, we perform one run for each of the 30 training scenarios. For larger values of N, exploring all possible subset combinations is not feasible. For example, even without replications, the *All-5* strategy would require $\binom{n}{k} = {}^{30}C_5 = \frac{30!}{5!(30-5)!} = 142{,}506$. Instead, we constructed 30 subsets of five traffic scenarios, where each scenario appears in at least five different subsets, and we perform one replication with each subset. In both *All-1* and *All-5*, if the particular subsets of the training scenarios do not influence the result, then this setup is equivalent to performing 30 independent replications. In total, each algorithmic variant generates 30 independent solutions.

After running the optimizers, the TLSP solutions generated are simulated on all traffic scenarios from the testing set. These are the results presented and analyzed in the next section.

ALGORITHM	STRATEGY	POPULATION	SIMULATIONS	SUBSETS	REP.	TOTAL
IRACE	Elitist F-Race	Dynamic	[2,30]	1	30	30
			[5,30]	1	30	30
		10	[2,30]	1	30	30
			[5,30]	1	30	30
DE, GA and PSO	All	10	1	30	1	30
			5	30	1	30
	Rand	10	1	1	30	30
			5	1	30	30
RS	All	1	1	30	1	30
			5	30	1	30
	Rand	1	1	1	30	30
			5	1	30	30

Fig. 6. Summary of the experiments performed. Different algorithms use different simulation strategies. The number of simulations with different traffic scenarios per solution is either $N = 1$ or $N = 5$ for both strategies *All* and *Rand*, while this value is dynamically adjusted by IRACE within $[T^{\text{first}}, 30]$. We experimented with $T^{\text{first}} = 5$ (default) and $T^{\text{first}} = 2$. In order to avoid biases when using only a subset of the training traffic scenarios, we evaluated different SUBSETS. A single subset means that the whole training set is available to that strategy. Finally, we performed independent replications (REP), i.e., algorithm runs with different random seeds, of each experiment. The column TOTAL gives the number of different final TLSP solutions generated for each variant tested.

In all experiments, we stop each run of an algorithm after executing 30 000 calls to the SUMO simulator. We use default settings for IRACE, unless noted otherwise. In particular, the population size of IRACE is dynamically adjusted by default and not fixed. The GA uses a population size of 10, the probability of crossover is 1.0 and the probability of mutation is 0.1. These parameter settings were found by additional experiments carried out in previous studies [6] to produce a search behavior that is more exploitative rather than explorative, which is more appropriate for our problem. For the sake of a fair comparison, the DE and the PSO also use a population of 10 individuals. The DE strategy is "best/1/bin", the difference factor is $F = 0.5$ and the probability of crossover is 0.5. The PSO acceleration coefficient is $0.5 + \log(2)$, the inertia $1/(2\log(2))$, and the velocity truncation factor is -0.5. These are the default parameter values set by jMetal [34]. The random search does not have any parameters.

All simulations were performed with SUMO version 0.22. Since we already introduce variability by means of the different traffic scenarios, we fix the random seed used by SUMO to zero in all simulations. This means that, given a traffic scenario and a candidate solution, the simulation is deterministic. Thus, we enable the *deterministic* option in IRACE, which assumes that the problem does not have within-blocks variance, but there is still between-blocks variance.

IRACE is implemented in R and we used version 2.3, which is available at https://cran.r-project.org/package=irace. The GA, DE, PSO and RS are implemented in Java using jMetal 5.0 [34]. The source code of the standard operators used here is publicly available at https://github.com/jMetal/jMetal.

The experiments were run on a cluster of 16 machines with Intel Core2 Quad processors Q9400 (4 cores per processor) at 2.66 GHz and 4 GB memory and 2 nodes (96 cores) equipped with two Intel Xeon CPU (E5-2670 v3) at 2.30 GHz and 64 GB memory. The cluster was managed by HTCondor 8.2.7, which allowed us to perform parallel independent executions to reduce the overall experimentation time.

5. Experimental analysis

One goal of our research is to identify the best optimization method and simulation strategy for the real-world Traffic Light Scheduling Problem (TLSP) of the city of Málaga (Spain), but we believe that our methods, findings and conclusions are applicable to other cities. Our initial starting point was that, due to the variability in traffic conditions, more complex simulation strategies would be necessary to reach reliable and near-optimal TLSP solutions. On the other hand, previous studies have shown or assumed that the fitness of different simulations under normal traffic conditions, i.e., on similar (homogeneous) traffic scenarios, is highly correlated [8–10,17]. Their conclusion was that the resulting fitness variance would be small and no worth considering during optimization.

As a first step, we study the heterogeneity of the traffic scenarios by sampling 10 000 random solutions of our case study and evaluating them on the 30 training traffic scenarios. Fig. 7 shows the Spearman's ρ correlation between the traffic scenarios with respect to the fitness obtained by the random solutions. A positive correlation means that the ranking of solutions is not affected by the traffic scenarios whereas a negative correlation means that solutions better on one traffic scenario would perform worse on the other. As shown by the plot, almost all traffic scenarios are very highly correlated with no pair of traffic scenarios showing any negative correlation, which is expected since all scenarios are generated from the same sensor data. Thus, a premature conclusion would be that solutions optimized on one or a few traffic scenarios should be highly reliable and no complex simulation strategies are necessary. As we will show next, this is not the case.

5.1. Comparison of simulation strategies

The simplest and most common strategy corresponds to *All-1*, i.e., use a single traffic scenario per optimization run. This is the strategy followed by methods that do not expect any noise or variance in the fitness function. A simple extension of the *All-1* strategy for handling noise is to evaluate a solution on N traffic scenarios, then use the mean fitness as the fitness of the solution (*All-N*). Since the default minimum number of traffic scenarios simulated per solution by IRACE (T^{first}) is five, we run experiments with the GA, DE, PSO and RS using $N = 5$.

An alternative simulation strategy is *Rand-1*, which uses a different random traffic scenario from the training set at each iteration, i.e., new solutions in the population are evaluated at

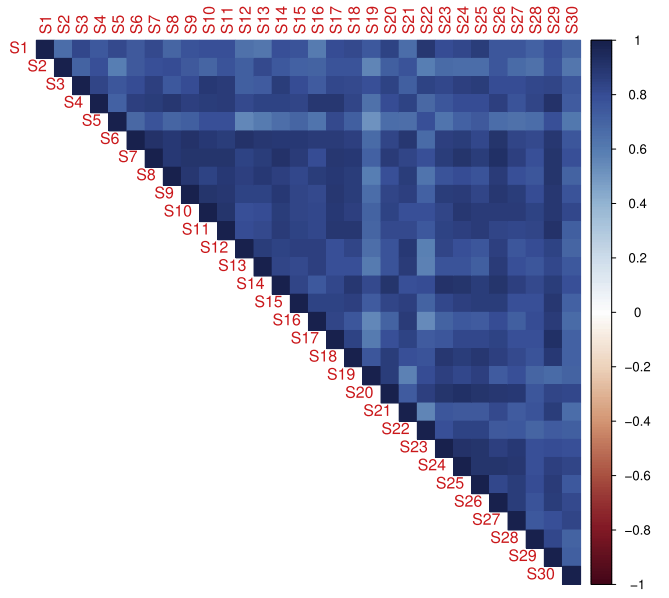

Fig. 7. Spearman's ρ correlation between the results obtained by 10 000 randomly generated solutions (TLPs) on all training traffic scenarios.

each iteration on a new traffic scenario. The actual behavior is that good solutions survive across iterations, whereas poor solutions are quickly discarded. The downside is that solutions that are "lucky" at a late iteration may replace solutions that are generally reliable but "unlucky" on the same iteration. Therefore, *Rand-1* can be seen as an intermediate strategy between *All-1* and racing. For completeness, we also evaluated the *Rand-5* strategy, where the fitness of a solution is computed as the mean fitness over five traffic scenarios and these five traffic scenarios are randomly selected from the training set at each iteration. Such strategy is expected to be more reliable against last-minute "lucky"/"unlucky" simulations at the cost of evaluating five times fewer solutions.

We ran the GA, DE, PSO and RS using the four simulation strategies described above and results are presented in Figs. 8–11, respectively. Each boxplot shows the distribution of fitness of one TLSP solution on the 30 traffic scenarios of the testing set. Thus, the spread of the boxplot, including the outliers shown as points, indicates the reliability of one particular solution.

In the case of the GA, strategy *All-5* is statistically better in mean fitness over all replications (0.1793) than the rest (pairwise Wilcoxon rank-sum test at 0.05 significance level with Holm's *p*-value correction). In the case of the DE, strategy *All-5* is the best in mean fitness over all replications (0.1768), although the difference in means with the rest of strategies is not statistically significant (using the same pairwise Wilcoxon test). In the case of the PSO, strategy *All-5* is statistically better in mean fitness over all replications (0.3797) than the rest (pairwise Wilcoxon rank-sum test at 0.05 significance level with Holm's *p*-value correction). Finally, in the case of the RS, the strategy *Rand-1* obtains the lowest overall mean (0.3767), the difference in means with *All-5* is not statistically significant (using the same pairwise Wilcoxon test as above).

In terms of reliability, the plots show that the GA and DE produce much more reliable solutions than the PSO and RS. In the case of the RS (Fig. 11), the worse strategy is *All-1*, which sometimes produces fitness values outside the range shown in the plot. Regarding PSO (Fig. 10), the worse strategy is *Rand-1*. Although PSO is better than RS, the results obtained with PSO in this context are far in quality from the other metaheuristics (GA and DE).

Increasing N from 1 to 5 slightly improves the reliability of solutions (smaller boxes) for both simulation strategies *All-N* and *Rand-N*. In the case of the GA (Fig. 8) and DE (Fig. 9), the strategy *All-1* sometimes generates reliable solutions (small boxes) with few outliers, but other runs generate both poor (high fitness) and unreliable solutions (large variance). The number of reliable solutions increases for *All-5* with only a few slightly larger boxplots, however, there are still a few poor quality (high fitness) solutions. The *Rand-N* variants produce much more diverse solutions: sometimes very good and reliable, other times very poor and unreliable.

5.2. Comparison with IRACE using default settings

We perform 30 runs of IRACE with default settings, that is, the minimum number of traffic scenarios simulated per solution (T^{first}) is five and the population size is dynamically adjusted. We also run SUMO's greedy algorithm (SCPG), which is deterministic and, thus, returns a single solution.

Fig. 12 compares the fitness, on the testing set, of the solutions obtained by IRACE and SCPG with those obtained by the GA and DE when using their best simulation strategies. The results of SCPG are not only very poor (several values are outside the worst range shown in the plots), but also very sensitive to the traffic scenario. Additionally, in Fig. 12 we can also appreciate that both GA and DE compute reliable solutions (small boxes), however, the median fitness of the solutions computed by the GA is slightly better than the DE (pairwise Wilcoxon rank-sum test at 0.05 significance level).

The results obtained by IRACE show a remarkable improvement over SCPG, yet they are significantly worse than those obtained by the GA and DE, in terms of both fitness value and variance. One possible explanation is that IRACE is sometimes using many more than five simulations for evaluating each solution, in fact, sometimes using all 30 training traffic scenarios, which leads to fewer solutions being explored. This lack of exploration would explain the poor results if the heterogeneity between traffic scenarios is sufficiently low to not require simulating the same solution on a large number of traffic scenarios, as suggested by the previous correlation analysis. We study this explanation in the following section.

5.3. Effect of decreasing the number of simulations and of a fixed population size in IRACE

We examine how much reliability is lost and how much solution quality may be improved if we decrease the minimum number of traffic scenarios simulated per candidate solution from five to two ($T^{\text{first}} = 2$). Decreasing below two is not possible at the moment, since this is the minimum required by the elimination test used by IRACE.

Given the poor statistical power of testing with just two samples, one may expect that a setting of $T^{\text{first}} = 2$ would produce even worse results. On the other hand, statistical tests on IRACE are merely used as a heuristic for choosing between candidate solutions. If the variance between traffic scenarios is lower, a lower-power test simply ends up choosing between solutions that are not statistically different, which makes the search more greedy. As shown by the results in Fig. 13, this appears to be the case here and the setting of $T^{\text{first}} = 2$ improves the performance of IRACE with respect to the results obtained with $T^{\text{first}} = 5$. In particular, the results become even more reliable than before. However, in terms of solution quality, the results obtained by IRACE remain significantly worse than the ones obtained by *GA-all-5*.

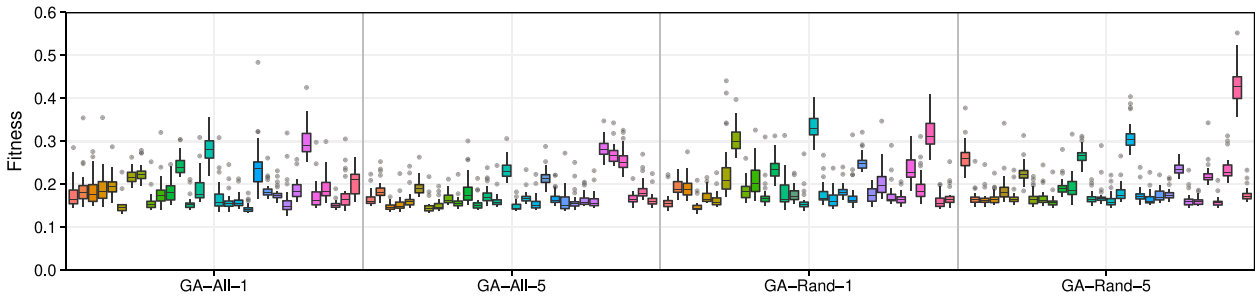

Fig. 8. Fitness of TLSP solutions generated by the GA using various simulation strategies. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the testing set. For each GA variant, there are 30 solutions obtained by independent replications.

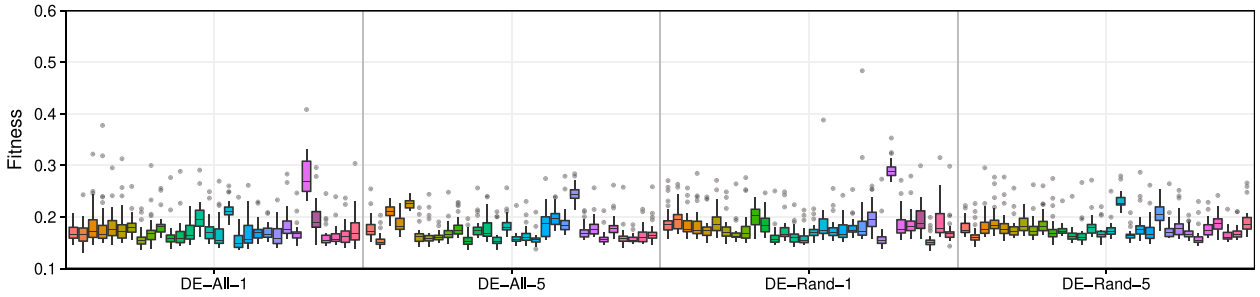

Fig. 9. Fitness of TLSP solutions generated by the DE using various simulation strategies. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the testing set. For each DE variant, there are 30 solutions obtained by independent replications.

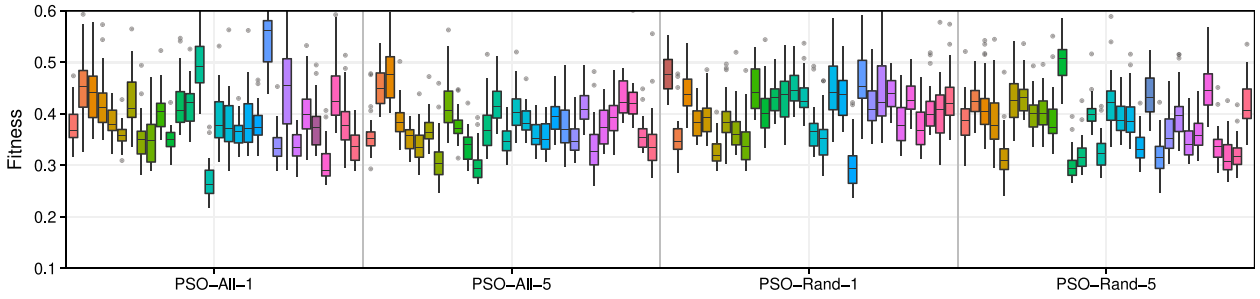

Fig. 10. Fitness of TLSP solutions generated by the PSO using various simulation strategies. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the testing set. For each PSO variant, there are 30 solutions obtained by independent replications.

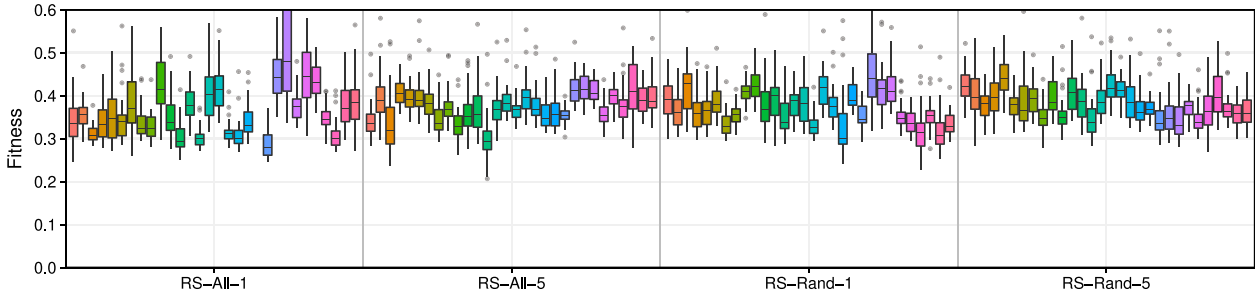

Fig. 11. Fitness of TLSP solutions generated by the RS using various simulation strategies. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the testing set. For each RS variant, there are 30 solutions obtained by independent replications.

This difference in solution fitness between IRACE with dynamic population size and the GA is due to the slow convergence of IRACE, as shown by the plot of fitness over number of evaluations in Fig. 14. Each line in the plot is the fitness value of the best-so-far solution, as estimated by each algorithm at each moment of its execution, averaged over all independent repetitions for each algorithm. The gray shaded area around each line corresponds to the 95% confidence interval around the mean based on Student's t-distribution. The plot shows that IRACE with dynamic population (IRACE $T^{\text{first}} = 5$ and IRACE $T^{\text{first}} = 2$) continues improving its best solution until the maximum number of evaluations is reached, however, its convergence is much slower than the GA and it stops at much worse solutions. When the population size of IRACE is fixed to 10, the convergence speeds up substantially (IRACE $T^{\text{first}} = 5$, $|\Theta_t| = 10$ in Fig. 14). Note that this convergence plot shows the perspective of the optimization algorithms, that is, mean fitness over training scenarios. Results are slightly different

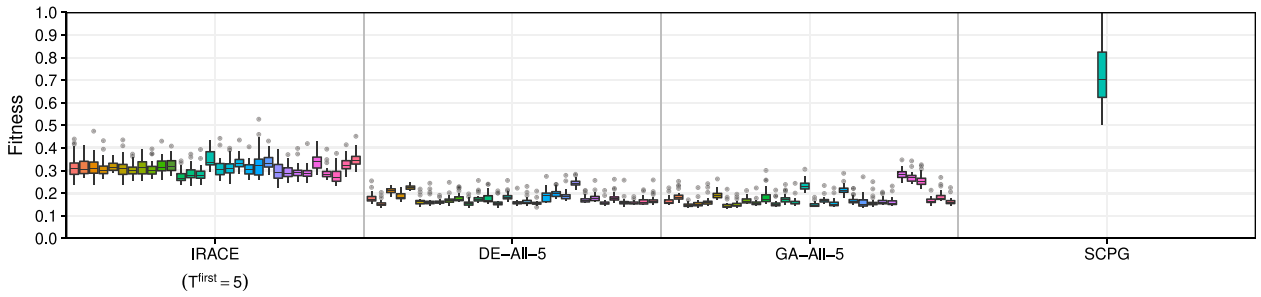

Fig. 12. Fitness of TLSP solutions obtained by IRACE with default settings, the DE and GA using their best simulation strategies, and SUMO's SCPG algorithm. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the testing set. For IRACE, DE and GA, there are 30 solutions obtained by independent replications. SCPG produces a single solution. The range of the *y*-axis has been extended to include all SCPG values.

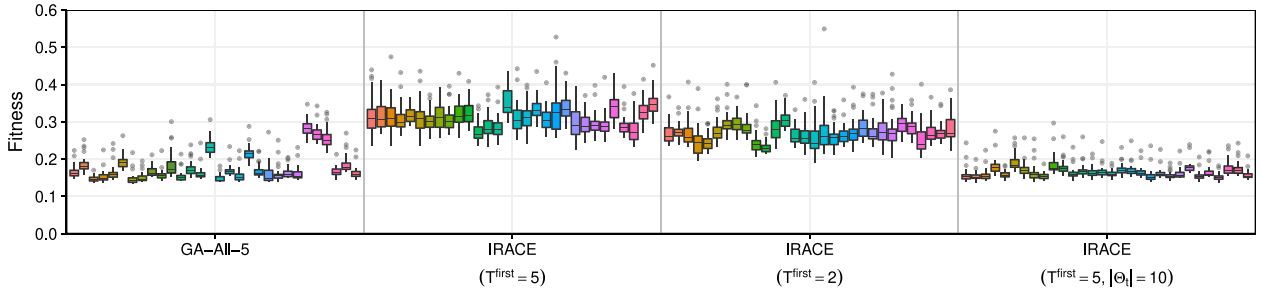

Fig. 13. Fitness of TLSP solutions generated by the best GA strategy (*All-5*), and IRACE using default settings ($T^{\text{first}} = 5$, dynamic population size), using a smaller value of T^{first} ($T^{\text{first}} = 2$) and using a fixed population size ($T^{\text{first}} = 5$, $|\Theta_t| = 10$). Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the testing set. For each algorithm variant, there are 30 solutions obtained by independent replications.

when evaluating the solution returned at the end by the algorithms on the testing scenarios. Also, we only change here one parameter of IRACE at a time. As the next section shows, IRACE with both $T^{\text{first}} = 2$ and $|\Theta_t| = 10$ further improves the final results.

The conclusion from the above analysis is that IRACE with dynamic population is not converging fast enough and a more aggressive search behavior could be beneficial. In fact, previous studies [6] have shown that the TLSP requires a rather aggressive search behavior. This different convergence rate is caused by the population of just 10 solutions, whereas the default dynamic population size mechanism in IRACE uses much larger sizes leading to slow convergence.

When we run IRACE with the same fixed population size of 10 as the GA and DE, there is a massive improvement in both solution quality and reliability, as shown in Fig. 13 (IRACE with $T^{\text{first}} = 5$, $|\Theta_t| = 10$). This latest variant of IRACE not only produces fitness values as good as those obtained by *GA-All-5*, but also it improves solution reliability over different traffic conditions (as shown by the smaller size of the boxes) and over different algorithm runs (IRACE does not produce the poor solutions sometimes generated by *GA-All-5*).

5.4. Comparison of best algorithm variants

Finally, we compare the best variants of the GA, DE, and IRACE, according to the previous analysis. In particular, the GA (*GA-All-5*) and the DE (*DE-All-5*) simulate five traffic training scenarios per solution; and IRACE uses same fixed population size of 10 as the GA and DE, and a minimum number of simulations per solution of 2.

Results are shown in Fig. 15, including SCPG for comparison. As it can be observed, the results of IRACE are clearly better than those of the SCPG in terms of fitness value and low variance. IRACE also obtains a lower mean fitness than the GA and DE (statistically significant according to the Wilcoxon rank-sum test with p-value < 0.01). The fitness variance is also much lower for IRACE than for the GA and DE, as shown by the consistently narrower boxplots, with only a few outliers above 0.2. Moreover, different replications of IRACE obtain roughly the same results, which is another measure of reliability. Thus, we can conclude that IRACE produces the best solutions for our case study in terms of quality and reliability among all the variants analyzed.

5.5. Detailed analysis of median traffic-light programs

In the above analysis, we have focused on the fitness values obtained by the solutions generated by the various algorithms. However, the fitness value is only an approximation of the various traffic and environmental measures that are the actual motivation behind the optimization of traffic-light programs in our case study. Thus, we want to shed light over the practical benefits of the "typical" solutions obtained by the various algorithms in terms of specific measures such as travel time, waiting time, fuel consumption and vehicle emissions.

During optimization, the traffic is simulated up to a predefined time horizon (1 h plus 10 min of warm-up, in our case), in order to avoid large differences in simulation times, and a penalty is added to the fitness value (Eq. (3)) if there are vehicles with incomplete journeys at the end of the simulation. By contrast, in this section, we run the simulation until all vehicles reach their destination, in order to precisely calculate measures such as fuel consumption, travel time, etc. Thus, the fitness value of these simulations is not penalized.

In order to analyze and compare "typical" traffic-light programs (solutions), we compute the mean fitness over the traffic scenarios in the testing set of the 30 solutions generated by each algorithm and we select the solution that obtains the median value. Since there is an even number of values (30), the true median value does not correspond to any solution, thus we selected the solution with the smallest value larger than the median one, i.e., the one in the 16th position when ordered by increasing mean

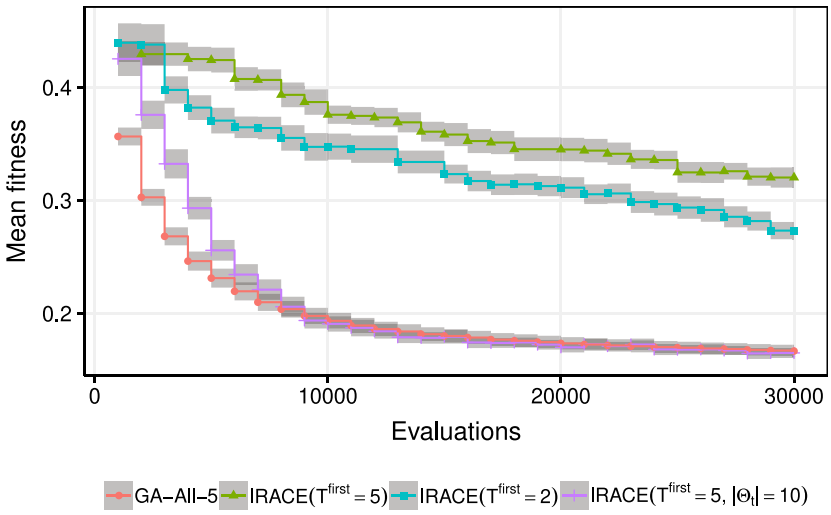

Fig. 14. Mean fitness of the best solution found so far within each run, as estimated by each algorithm at each moment of its execution on traffic scenarios from the training set. The algorithms shown are the best GA strategy (*All-5*), and IRACE using default settings ($T^{\text{first}} = 5$, dynamic population size), using a smaller value of T^{first} ($T^{\text{first}} = 2$) and using a fixed population size ($T^{\text{first}} = 5$, $|\Theta_t| = 10$). The gray shaded area around each line corresponds to the 95% confidence interval around the mean based on Student's t-distribution.

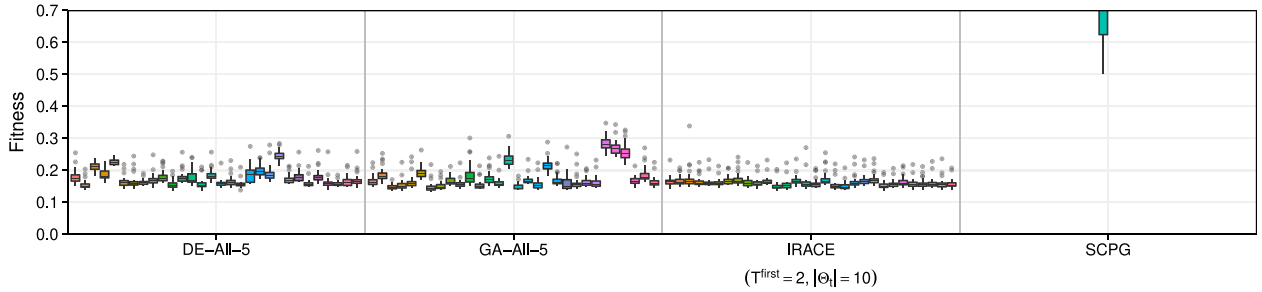

Fig. 15. Fitness of TLSP solutions generated by the best GA strategy (*All-5*), the best RS strategy (*Rand-1*), IRACE using a fixed population size ($T^{\text{first}} = 2$, $|\Theta_t| = 10$) and SCPG. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the testing set. There are 30 solutions obtained by independent replications for each algorithm, except for SCPG that produces a single solution.

fitness. The selected solutions are then simulated again on the 30 test traffic scenarios, each simulation runs until all vehicles reach their destination, and we compute a set of 32 traffic and environmental measures.

Table 1 shows the mean value and standard deviation over 30 traffic scenarios of six traffic measures provided by SUMO. Each measure value is either the mean, maximum or minimum over all vehicles of: *TravelTime*, the time in seconds required by a vehicle to reach their destination; *WaitingTime*; the time spent waiting at traffic lights; and *Fuel* and *CO2*, the amount of fuel consumed and CO_2 emitted by the vehicle in their journey, respectively. The full table with the 32 measures is available as supplementary material at https://goo.gl/PHCPlV. We also show the mean *non-penalized* fitness obtained by each solution. The results show a huge difference in all measures between the TLPs produced by the GA, DE and IRACE and those produced by the PSO, RS and SCPG. A pairwise Wilcoxon test with Holm's *p*-value correction (at 0.05 significance level) indicates that differences are significant between all algorithms in all measures, except in two cases. The first exception is between the GA, DE and IRACE, and the second one between PSO and RS, where the differences are not significant at all. Consequently, we can distinguish here three groups of algorithms, the best algorithms group composed by GA, DE and IRACE, the second group in quality composed by PSO and RS, and finally SCPG which is significantly worse than the rest of algorithms. In general, we can observe that better traffic light programs result in less wasted time, fuel consumption and CO2 emissions. Therefore, the practical benefits of using better algorithms are positive for citizens, institutions, and in the long term, for our world.

Although for the median solution there is no statistical significant differences between IRACE, GA and DE, the boxplots in Fig. 15 show that the GA and DE often generate much worse solutions than IRACE, while IRACE solutions are much more consistent. Moreover, IRACE consistently obtains the best mean values and lowest standard deviation in almost all measures (see Table 1 and the full table available online). In the case of *MeanFuel* and *MeanCO2* for the GA algorithm, these measures are not directly optimized by our fitness function, yet they are positively correlated with lower fitness.

The detailed analysis confirms our expectation that optimizing the fitness taking into account its variance over multiple traffic scenarios also increases the reliability of the solutions for various traffic and environmental measures. In practice, such solutions are more reliable when dealing with the real-world traffic day after day.

6. Conclusions

The research presented in this work deals with a real-world case study of the TLSP and our data comes from specifications provided by the City Council of Málaga and data collected directly from sensors at the street level, which was used to create a number of vehicle traffic scenarios representing typical traffic

Table 1
Traffic measures per vehicle. Mean values (and standard deviation) over 30 test traffic scenarios of the median solutions of the best strategy for IRACE, GA, DE, PSO, RS and SCPG. The best value for each measure is highlighted.

	IRACE $T^{\text{first}} = 2, \lvert\Theta_t\rvert = 10$		GA *All-5*		DE *All-5*		PSO *All-5*		RS *Rand-1*		SCPG	
MeanTravelTime (s)	331.97	(17.42)	334.72	(26.62)	339.48	(20.99)	532.78	(32.73)	545.90	(46.57)	1155.52	(287.05)
MaxTravelTime (s)	772.27	(217.13)	788.70	(238.53)	774.43	(215.12)	1817.87	(252.41)	2790.60	(850.20)	12265.70	(3822.28)
MeanWaitingTime (s)	89.37	(10.31)	90.30	(19.02)	96.09	(14.66)	249.48	(25.22)	253.09	(38.86)	1038.42	(36.74)
MaxWaitingTime (s)	355.67	(149.63)	411.13	(228.37)	393.30	(164.76)	1272.83	(187.65)	2197.93	(827.14)	11174.37	(3593.98)
MeanFuel (ml)	131.87	(3.01)	131.78	(3.94)	132.83	(3.38)	158.83	(4.78)	160.72	(6.97)	246.29	(46.26)
MeanCO2 (mg)	330766	(7568.47)	330545	(9877.95)	333166	(8475.33)	398394.35	(11980.32)	403137	(17477.37)	617752.64	(116019.74)
Fitness (non-penalized)	0.0876	(0.0054)	0.0882	(0.0073)	0.0905	(0.0055)	0.1627	(0.0113)	0.1660	(0.0161)	0.4158	(0.1158)

flows. Due to the inherent variance in traffic flows, it is expected that the fitness of any candidate solution to the problem will vary across traffic scenarios and when deployed in the real-world traffic system. Thus, good solutions should not only produce a near-optimal fitness, but also a small fitness variance, indicating reliability. Previous works have assumed that, under normal traffic conditions, the fitness variance is inherently small, specially if the fitness values across multiple simulations are correlated. As a result, previous optimization algorithms for the TLSP use just one simulation per solution to assess their fitness.

Our results show that, even when generating traffic scenarios consistent from the same sensor data, and even if the fitness of random solutions on those traffic scenarios is highly correlated, the reliability of the solutions generated may be improved by using more than one traffic scenario per optimization run. In particular, increasing the number of simulations per solution from one to five in metaheuristics applied to the TLSP leads to more reliable solutions and better mean fitness. However, the GA and DE, the best metaheuristics in our comparison, sometimes also produced poor and unreliable solutions. We also analyzed a strategy that changes the traffic scenario simulated at each iteration, thus making use of a larger number of traffic scenarios per run. Yet, this strategy did not lead to any improvement over the previous one.

The best results were obtained by the iterated racing strategy as implemented by IRACE, in which the number of simulations performed per solution is decided dynamically within each optimization run. Our analysis identified that the default settings of IRACE, which was originally proposed for the automatic configuration of algorithmic parameters, are not appropriate for the TLSP. In particular, the dynamic population size mechanism of IRACE overestimates the best population size for the TLSP, which is rather small. After using the same fixed population size in IRACE and the metaheuristics, the former was able to outperform all the metaheuristics both in terms of solution quality and reliability.

By analyzing in detail the median solutions produced by the algorithms using longer simulations, we show that the improvements in fitness mean value and variance over multiple traffic scenarios also translate to similar improvements on various traffic and environmental measures of practical interest.

In addition, we tested a PSO algorithm and two baseline algorithms, a classical random search and the greedy algorithm (SCPG) provided by the SUMO simulator. All of them performed significantly worse than the GA, DE and IRACE in terms of all the measures analyzed.

One main conclusion of this work is that future optimization algorithms for the TLSP should consider the reliability of solutions over multiple traffic scenarios and incorporate simulation strategies that improve reliability. Given the results presented here, the hybridization of iterated racing strategies with evolutionary algorithms is one the most promising directions.

A next step would be to re-evaluate previous studies [5,8–10,16,17] in terms of reliability, which will require a significant implementation work since those algorithms were designed for slightly different variants of the traffic-light optimization problem. We also want to compare our obtained solutions with the one currently used on the real system, however, the latter is not available to us at the present moment.

Future work should also study the reliability of simulation–optimization algorithms for the TLSP under unusual traffic scenarios (due to construction works, emergency scenarios, extremely congested periods). Techniques such as iterated racing allow optimizing reliability under such unusual scenarios by adding them to the training set used during optimization. We expect that the benefits reported in this paper will be even greater when considering also those unusual scenarios.

Acknowledgments

This research has been partially funded by the Spanish Ministry of Science and Innovation and FEDER under contract TIN2017-88213-R (6city), the network of smart cities CI-RTI (TIN2016-81766-REDT), the EcoIoT project (RTC-2017-6714-5), and the CELTIC C2017/2-2 project in collaboration with companies EMERGYA and SECMOTIC with contracts #8.06/5.47.4997 and #8.06/5.47.4996. Part of this work was carried out while M. López-Ibáñez was a visiting researcher at the NEO group thanks to the support of a grant ("*Estancias Tipo B, Fondos Propios UMA 2014*") from the University of Málaga. J.Ferrer thanks University of Málaga for his postdoc fellowship. We also thank Daniel Stolfi for generating the traffic scenario files from the sensor data.

References

[1] S. Wang, N.U. Ahmed, T.H. Yeap, Optimum management of urban traffic flow based on a stochastic dynamic model, IEEE Trans. Intell. Transp. Syst. (ISSN: 1524-9050) (2018) 1–13.
[2] J. Ide, A. Schöbel, Robustness for uncertain multi-objective optimization: a survey and analysis of different concepts, OR Spektrum 38 (1) (2016) 235–271, http://dx.doi.org/10.1007/s00291-015-0418-7.
[3] A.A. Juan, J. Faulin, S.E. Grasman, M. Rabe, G. Figueira, A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems, Oper. Res. Perspect. 2 (2015) 62–72, http://dx.doi.org/10.1016/j.orp.2015.03.001.
[4] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments—a survey, IEEE Trans. Evol. Comput. 9 (5) (2005) 303–317.
[5] Z. Cao, S. Jiang, J. Zhang, H. Guo, A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion, IEEE Trans. Intell. Transp. Syst. 18 (7) (2017) 1958–1973.
[6] Y. Bravo, J. Ferrer, G.J. Luque, E. Alba, Smart mobility by optimizing the traffic lights: A new tool for traffic control centers, in: E. Alba, F. Chicano, G. Luque (Eds.), Smart Cities (Smart-CT 2016), in: Lecture Notes in Computer Science, Springer, Cham, Switzerland, 2016, pp. 147–156, http://dx.doi.org/10.1007/978-3-319-39595-1_15.
[7] J. Ferrer, J. García-Nieto, E. Alba, F. Chicano, Intelligent testing of traffic light programs: Validation in smart mobility scenarios, Math. Probl. Eng. 2016 (2016) 1–19, http://dx.doi.org/10.1155/2016/3871046.
[8] J. García-Nieto, E. Alba, A.C. Olivera, Swarm intelligence for traffic light scheduling: Application to real urban areas, Eng. Appl. Artif. Intell. 25 (2) (2012) 274–283.
[9] J. García-Nieto, A.C. Olivera, E. Alba, Optimal cycle program of traffic lights with particle swarm optimization, IEEE Trans. Evol. Comput. 17 (6) (2013) 823–839, http://dx.doi.org/10.1109/TEVC.2013.2260755.

[10] J. Sánchez, M. Galán, 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. 12 (1) (2008) 25–40.
[11] J.J. Sánchez-Medina, M.J. Galán-Moreno, E. Rubio-Royo, Traffic signal optimization in "La Almozara" district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing, IEEE Trans. Intell. Transp. Syst. (ISSN: 1524-9050) 11 (1) (2010) 132–141, http://dx.doi.org/10.1109/TITS.2009.2034383.
[12] D.H. Stolfi, E. Alba, Red swarm: Reducing travel times in smart cities by using bio-inspired algorithms, Appl. Soft Comput. 24 (2014) 181–195, http://dx.doi.org/10.1016/j.asoc.2014.07.014.
[13] D.H. Stolfi, E. Alba, An evolutionary algorithm to generate real urban traffic flows, in: J.M. Puerta, J.A. Gámez, B. Dorronsoro, E. Barrenechea, A. Troncoso, B. Baruque, M. Galar (Eds.), Advances in Artificial Intelligence, in: Lecture Notes in Computer Science, vol. 9422, Springer, Heidelberg, Germany, 2015, pp. 332–343, http://dx.doi.org/10.1007/978-3-319-24598-0_30.
[14] F. Teklu, A. Sumalee, D. Watling, A genetic algorithm approach for optimizing traffic control signals considering routing, Comput.-Aided Civ. Infrastruct. Eng. 22 (1) (2007) 31–43, http://dx.doi.org/10.1111/j.1467-8667.2006.00468.x.
[15] K.T.K. Teo, W.Y. Kow, Y.K. Chin, Optimization of traffic flow within an urban traffic light intersection with genetic algorithm, in: Proceedings - 2nd International Conference on Computational Intelligence, Modelling and Simulation, CIMSim 2010, 2010, pp. 172–177.
[16] M. Péres, G. Ruiz, S. Nesmachnow, A.C. Olivera, Multiobjective evolutionary optimization of traffic flow and pollution in Montevideo, Uruguay, Appl. Soft Comput. J. (ISSN: 15684946) (2018).
[17] Z. Li, M. Shahidehpour, S. Bahramirad, A. Khodaei, Optimizing traffic signal settings in smart cities, IEEE Trans. Smart Grid 3053 (4) (2016) http://dx.doi.org/10.1109/TSG.2016.2526032, 1–1.
[18] L.C. Zammit, S.G. Fabri, K. Scerri, Real-time parametric modeling and estimation of urban traffic junctions, IEEE Trans. Intell. Transp. Syst. (ISSN: 1524-9050) (2019) 1–11.
[19] J. Wang, L. Zhou, Traffic light recognition with high dynamic range imaging and deep learning, IEEE Trans. Intell. Transp. Syst. (ISSN: 1524-9050) (2018) 1–12.
[20] N. Wu, D. Li, Y. Xi, Distributed weighted balanced control of traffic signals for urban traffic congestion, IEEE Trans. Intell. Transp. Syst. (ISSN: 1524-9050) (2018) 1–11.
[21] M. López-Ibáñez, J. Dubois-Lacoste, L. Pérez Cáceres, T. Stützle, M. Birattari, The irace package: Iterated racing for automatic algorithm configuration, Oper. Res. Perspect. 3 (2016) 43–58, http://dx.doi.org/10.1016/j.orp.2016.09.002.
[22] M. Birattari, T. Stützle, L. Paquete, K. Varrentrapp, A racing algorithm for configuring metaheuristics, in: W.B. Langdon, et al. (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002, Morgan Kaufmann Publishers, San Francisco, CA, 2002, pp. 11–18.
[23] S. Becker, J. Gottlieb, T. Stützle, Applications of racing algorithms: An industrial perspective, in: E.-G. Talbi, P. Liardet, P. Collet, E. Lutton, M. Schoenauer (Eds.), Artificial Evolution, in: Lecture Notes in Computer Science, vol. 3871, Springer, Heidelberg, Germany, 2005, pp. 271–283.
[24] E. Haasdijk, A. Atta-ul Qayyum, A.E. Eiben, Racing to improve on-line, on-board evolutionary robotics, in: N. Krasnogor, P.L. Lanzi (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2011, ACM Press, New York, NY, 2011, pp. 187–194.
[25] B. Yuan, M. Gallagher, Statistical racing techniques for improved empirical evaluation of evolutionary algorithms, in: X. Yao, et al. (Eds.), Proceedings of PPSN-VIII, Eigth International Conference on Parallel Problem Solving from Nature, in: Lecture Notes in Computer Science, vol. 3242, Springer, Heidelberg, Germany, 2004, pp. 172–181.
[26] V. Heidrich-Meisner, C. Igel, Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search, in: A.P. Danyluk, L. Bottou, M.L. Littman (Eds.), Proceedings of the 26th Annual International Conference on Machine Learning, ACM Press, New York, NY, 2009, pp. 401–408, http://dx.doi.org/10.1145/1553374.1553426.
[27] W. Hu, L. Yan, H. Wang, B. Du, D. Tao, Real-time traffic jams prediction inspired by Biham, Middleton and Levine (BML) model, Inform. Sci. (ISSN: 00200255) (2017).
[28] M. Behrisch, L. Bieker, J. Erdmann, D. Krajzewicz, SUMO - simulation of urban mobility: An overview, in: SIMUL 2011, The Third International Conference on Advances in System Simulation, Barcelona, Spain, 2011, pp. 63–68.
[29] D. Krajzewicz, J. Erdmann, M. Behrisch, L. Bieker, Recent development and applications of SUMO - simulation of urban mobility, Int. J. Adv. Syst. Meas. 5 (3–4) (2012) 128–138.
[30] T. Bäck, D.B. Fogel, Z. Michalewicz, Handbook of Evolutionary Computation, IOP Publishing, 1997.
[31] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995, pp. 39–43.
[32] M. Clerc, J. Kennedy, Standard PSO 2011. Particle Swarm Central, http://www.particleswarm.info/, 2011. (Online).
[33] W. Hu, H. Wang, Z. Qiu, C. Nie, L. Yan, A quantum particle swarm optimization driven urban traffic light scheduling model, Neural Comput. Appl. (ISSN: 09410643) (2018) http://dx.doi.org/10.1007/s00521-016-2508-0.
[34] A.J. Nebro, J.J. Durillo, M. Vergne, Redesigning the jMetal multi-objective optimization framework, in: J.L.J. Laredo, S. Silva, A.I. Esparcia-Alcázar (Eds.), GECCO (Companion), ACM Press, New York, NY, 2015, pp. 1093–1100.