Hybridization of Racing Methods with Evolutionary Operators for Simulation Optimization of Traffic Lights Programs

Christian Cintrano[0000−0003−2346−2198], Javier Ferrer[00000−0002−1074−0139], Manuel López-Ibáñez[0000−0001−9974−1295], and Enrique Alba[0000−0002−5520−8875]

University of Malaga, Bulevar Louis Pasteur 35, 29010 Malaga, Spain,
{cintrano,ferrer, manuel.lopez-ibanez, eat}@lcc.uma.es

Abstract. In many real-world optimization problems, like the traffic light scheduling problem tackled here, the evaluation of candidate solutions requires the simulation of a process under various scenarios. Thus, good solutions should not only achieve good objective function values, but they must be robust (low variance) across all different scenarios. Previous work has revealed the effectiveness of IRACE for this task. However, the operators used by IRACE to generate new solutions were designed for configuring algorithmic parameters, that have various data types (categorical, numerical, etc.). Meanwhile, evolutionary algorithms have powerful operators for numerical optimization, which could help to sample new solutions from the best ones found in the search. Therefore, in this work, we propose a hybridization of the elitist iterated racing mechanism of IRACE with evolutionary operators from differential evolution and genetic algorithms. We consider a realistic case study derived from the traffic network of Malaga (Spain) with 275 traffic lights that should be scheduled optimally. After a meticulous study, we discovered that the hybrid algorithm comprising IRACE plus differential evolution offers statistically better results than conventional algorithms and also improves travel times and reduces pollution.

Keywords: Hybrid algorithms, Evolutionary algorithms, Simulation optimization, Uncertainty, Traffic light planning

1 Introduction

In many real-world optimization problems, the evaluation of candidate solutions requires the simulation of a process under various scenarios that represent uncertainty about the real-world. Good solutions should not only achieve good

Please cite as: Cintrano C., Ferrer J., López-Ibáñez M., Alba E. (2021) Hybridization of Racing Methods with Evolutionary Operators for Simulation Optimization of Traffic Lights Programs. In: Zarges C., Verel S. (eds) Evolutionary Computation in Combinatorial Optimization. EvoCOP 2021. Lecture Notes in Computer Science, vol 12692. Springer, Cham. The final authenticated publication is available online at https://doi.org/10.1007/978-3-030-72904-2_2.

objective function values but also show robustness, i.e., low variance across scenarios. To assess the robustness of solutions, it is often required to simulate each solution a number of times using different data, starting conditions or random numbers. For example, when planning the traffic light schedules within a city[1], it is desirable to find a schedule that works well under many different traffic conditions [3, 5, 7, 8, 13, 15, 16, 17, 18, 20, 21]. A common approach is to simulate each candidate solution under a number of scenarios generated from real traffic data. However, there is a trade-off between the number of scenarios used for evaluating each solution and the number of candidate solutions evaluated.

Previous work [6] has shown that IRACE [11] is able to find high-quality and low-variance traffic light schedules by dynamically adjusting the number of simulations performed per solution. The elitist iterated racing algorithm implemented by IRACE has been traditionally used for the configuration of parameters in machine learning and optimization algorithms, where each configuration must be evaluated on a number of training instances of a problem and the algorithm themselves are often stochastic. The algorithm implemented in IRACE uses a learning mechanism inspired by reinforcement learning to sample new solutions from the best ones previously found. Although this approach tends to work well for configuring a mix of categorical and numerical parameters with dependencies and constraints among them, other operators may perform better when the problem consists only of numerical decision variables.

In this paper, we propose a hybridization of evolutionary operators from evolutionary algorithms (EAs) and the elitist iterated racing of IRACE. We evaluate its performance on the traffic light optimization problem and compare it with previous results from the literature. The idea of previous approaches to hybridizing EAs and racing was performing independent races to carry out the evaluation step within an EA [9], whereas our proposal replaces the sampling mechanism in IRACE, which is not simply a sequence of independent races, with evolutionary operators.

Besides, in order to add value to our experimentation, we use an instance based on real data from the city of Malaga, Spain. We also use a traffic simulator, SUMO [1, 10], to evaluate each of the traffic light schedules generated by the algorithms. With this, we not only seek to analyze which algorithm is better but also to solve a real problem of the city.

In summary, the main contributions of this work are:

- We propose new hybrid algorithms that combine racing strategies with evolutionary operators. Thus obtaining powerful and robust algorithms.
- We optimize the traffic light plan of a real city like Malaga (Spain) using detailed micro-simulations.
- We offer an in-depth analysis of our hybrid algorithms and compare them with well-known EAs such as a genetic algorithm (GA) and a differential evolution (DE).
- We study which algorithm presents the greatest improvement to the city according to different measures about traffic quality and emission reduction.

[1] Legal and technical limitations may make real-time traffic light control infeasible.

The rest of this article is organized as follows: Section 2 presents a description of the Traffic Light Scheduling Problem. Section 3 describes the main contribution of this work, the hybridization between IRACE and EAs. Section 4 outlines the main aspects of our experimentation. We discuss the results obtained in Section 5. Finally, Section 6 presents conclusions and future work.

2 Problem Description

Traffic flow in large smart cities has become one of the most severe problems that large cities face. In some cases, this problem is further aggravated due to the high amount of traffic jams, traffic accidents, or even injured people or deaths. Therefore traffic must be regulated with some elements such as traffic lights. The larger the metropolitan area, the higher the number of traffic lights needed to regulate the traffic flow. Optimal management of traffic might be beneficial to minimize journey times, reduce fuel consumption and harmful emissions.

Traffic lights are coordinated in phases: green, yellow and red. In this way, when some traffic lights of the same intersection are in green, some others must be in red. Besides, the different pre-defined phases for an intersection are sequences repeated over time, we call traffic light program (TLP) to each of those sequences.

The large number of program combinations that appear in traffic light schedules of large cities require automatic tools to generate optimal TLP, which motivates the Traffic Light Scheduling Problem (TLSP) [8, 15, 16]. The main objective in this problem is to find optimized TLP for all the traffic lights located in the intersections of an urban area with the aim of reducing journey time, emissions, and fuel consumption.

Let us define the TLSP 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 predefined valid phases $I_i = \{\varphi_{i1}, \ldots, \varphi_{im_i}\}$, where $m_i = |I_i|$ and each $\varphi_{ij} \in \mathbb{N}^+$ represents the duration (in seconds) of phase j in intersection I_i, that is, the duration of each valid phase of light colors (e.g., "*rr yyg rr gyyy*"). The objective is to find a TLP P' that minimizes a fitness function $f\colon \Gamma \to \mathbb{R}$ such that:

$$P' = \underset{P \in \Gamma}{\arg\min}\{f(P)\} \tag{1}$$

where Γ is the space of all possible TLPs.

In order to define the fitness function, we need to explain some previous concepts used in the definition. The evaluation of a solution is performed using a traffic simulator that provides information regarding the flow of vehicles. Vehicles travel from a starting position to a destination position, then the travel time (t_v) of a vehicle v is the number of simulation steps (1 second 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.

Long phase duration may lead to a collapse of the intersection. TLPs should prioritize those phases with more green lights on the directions with a high number of vehicles circulating. So, we should maximize the following ratio measure:

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

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 a division by 0.

Finally, we define 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 solution P proportionally to the simulation time t^{sim}. The number of vehicles that arrive at their destinations is squared ($V(P)^2$) to prioritize this criterion over the rest. This fitness function has been successfully used in [7, 8].

3 Hybridization of IRACE and Evolutionary Algorithms

There are many definitions of hybrid algorithms, yet the general idea is to combine components or concepts from different techniques to exploit desirable characteristics of those components to tackle problems with particular features [2]. In this work, we combine the elitist iterated racing strategy from IRACE with evolutionary operators to obtain an algorithm that performs well on numerical optimization problems where the fitness of each solution is uncertain and must be evaluated using multiple simulations. The elitist iterated racing strategy of IRACE decides how many simulations should be performed per solution, how solutions are compared, and which solutions should be discarded at each iteration. The evolutionary operators are responsible for generating new solutions from the surviving population of solutions. Next, we will briefly explain the base algorithm, IRACE, and the different characteristics of the hybrid algorithm.

3.1 IRACE

IRACE [11] is a well-known tool for automatic (hyper-)parameter configuration of optimization and machine learning algorithms. In the context of automatic parameter configuration, decision variables correspond to algorithmic parameters, candidate solutions correspond to potential configurations of an algorithm, and evaluating the fitness of a solution requires running the algorithm with a particular parameter configuration on multiple training data or problem instances. However, IRACE can be seen as an optimization method for mixed-integer black-box problem under uncertainty, and, hence, it may be used to tackle simulation-optimization problems, such as the TLSP [6].

Algorithm 1 Pseudocode of IRACE

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

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

Algorithm 1 briefly presents IRACE applied to the TLSP. Initially, a set of solutions are sampled uniformly at random. Then a race is performed to identify the best solutions among the initial set. Within a race, each solution is simulated multiple times on different traffic scenarios until there is enough evidence to eliminate it because it is performing worse than the best solution found so far. In the TLSP, we use the pairwise paired Student's t-test as the elimination test. The race stops once a minimum number of solutions remains alive in the race, the budget assigned to the race is exhausted, or multiple elimination tests fail to eliminate any solution. The solutions that remain alive after the race are called elite. These elite solutions are used to update a sampling model in a similar fashion as reinforcement learning, from which new solutions are generated. New and elite solutions together form a new population that is raced again. In elitist racing, results from previous races are re-used in subsequent races and elite solutions cannot be eliminated from the race until the contender has been evaluated in as many scenarios as the elite solution. This process is iterated until a maximum budget of simulations is exhausted. The main benefit of the racing strategy is that poor solutions are discarded quickly to avoid wasting simulations, while good solutions are simulated on many scenarios to provide a good estimate of their fitness. Moreover, the elimination test takes into account not only the mean value over multiple simulations but also the variance and the number of simulations performed so far.

3.2 Hybrid Algorithms

Once we have described how IRACE works, let us analyze our hybrid algorithms. In line 7 of Algorithm 1, the function $\mathsf{Sample}(\mathcal{M})$ generates a new set of candidate solutions to the problem. In our hybrid algorithms, we replace that function with operators taken from two EAs: Genetic Algorithm (GA) and Differential Evolution (DE). We call IRACE+GA and IRACE+DE, respectively, to these new hybrid algorithms. These EAs have already demonstrated their effectiveness in solving the TLSP [6], so we consider them to hybridize with IRACE. In this way, the racing step remains intact but the sampling of new solutions is carried out by these EAs. In IRACE, the $\mathsf{Sample}(\mathcal{M})$ procedure is equivalent to the mating selection and variation steps of an EA, i.e., selecting parents, generating new individuals from them (crossover), making some modification to the new solutions (mutation), and returning the new set of solutions. IRACE works with both numerical and categorical parameters. TLSP only has numerical parameters, so, in this paper, we do not have to deal with categorical parameters.

The set of elite solutions Θ^{elite} contains the best solutions found by IRACE after the race performed at each iteration (line 9). In our hybrid algorithm, the parents used by the evolutionary operators are selected from Θ^{elite}. However, the size of Θ^{elite} may vary each iteration and may be insufficient for the number of parents required by the evolutionary operators. We handle this situation by generating additional parents by random uniform sampling (as in line 2). This mechanism also introduces more diversity to the set of parent solutions. Because we use several evolutionary operators, the number of selected parents differs from one algorithm to another. IRACE+GA needs two parents for the operator execution, while IRACE+DE needs four. The restriction in the number of parents is given by the operators used by each algorithm, because each operator requires a different number of solutions to generate a new one.

In this work, we have implemented two variants of the proposed hybrid algorithm with the following operators:

– IRACE+GA: uniform crossover [19] and integer polynomial mutation [4]
– IRACE+DE: the "DE/best/1/bin" strategy [14].

The evaluation of the new solutions returned after the $\mathsf{Sample}(\mathcal{M})$ phase is computed by performing several simulations, as carried out by IRACE. After the evaluation phase, we merge this set of new solutions Θ^{new} with set of elite solutions Θ^{elite} to execute the racing. This returns a new set of elite solutions, which will be used in the next iteration of the hybrid algorithms.

4 Experimental Setup

We describe here the experimental protocol followed in this work. First, we describe the real-world case study of TLSP that is the main motivation of our research. After that, we provide details about the experiments carried out. We will analyze these experiments in the next section.

4.1 Real World Case Study

We consider a realistic scenario derived from the traffic network of Malaga [18], which encompasses an area of about $3\,\text{km}^2$ with 58 intersections controlled by 275 traffic lights (Fig. 1). Our network model was created from real data about traffic rules, traffic element locations, road directions, streets, intersections, etc.

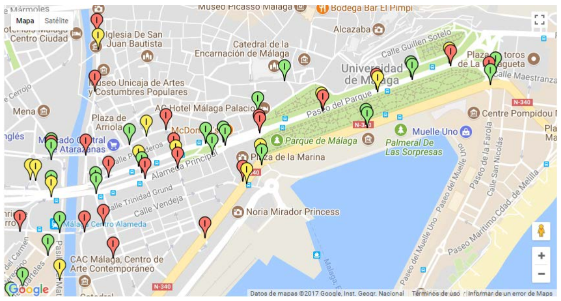

Fig. 1. Locations of traffic lights considered in the case of study. The colors show large (red), medium (yellow) and small (green) differences between two different solutions.

Once we have a realistic traffic network of a city, we need the routes and vehicles circulating and their speeds. This information was collected from sensorized points in certain streets measuring traffic density at various time intervals. From the sensed data extracted, we have applied the Flow Generator Algorithm (FGA) [18] to generate 60 different traffic scenarios with an average of 4,827 vehicles (or different vehicle routes) per scenario. 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.

4.2 Case Study Constraints

Real-world instances of the TLSP often present additional constraints. In our case, we consider the constraints recommended by the City Council of Malaga (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. These fixed phases correspond to pedestrian crosses, which last for a fixed time of $4 \times$ *number of lanes* seconds. Non-fixed phases have a minimum duration of $\varphi_{\min} = 15$ seconds. Moreover, the total program 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 Malaga (Spain), $Tp_{\min} = 60$ and $Tp_{\max} = 120$ seconds.

By default, the first programs 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 program at the start of the simulation. If the offset value of an intersection is negative, then program start time is shifted back that number of seconds and the program actually starts on a phase before the first one; whereas if the offset is positive, the program 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 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 $To_i \in [To_{\min}, To_{\max}] = [-30, 30]$.

4.3 Repair Procedure

The TLSP is subject to some constraints we have just explained in Section 4.2. To ensure that candidate solutions are valid, we propose a repair procedure that is used by all the algorithms before the simulation. 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 program time Tp_i is within $[Tp_{\min}, Tp_{\max}]$. Here we can distinguish two different cases.

In the first case, if the total program 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.

In the second case, if the total program 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 within intersection I_i.

4.4 Simulator: SUMO

The quality of a solution (traffic light program) is evaluated through the Simulator of Urban Mobility (SUMO) [1, 10], which is a microscopic road traffic simulator that provides detailed information about vehicles like velocity, fuel consumption, emissions, journey time, waiting time, etc. The study of realistic scenarios according to real patterns of mobility of the target city is possible due to the fine-grained realistic micro-simulations provided by SUMO.

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. In all experiments, we stop each run of an algorithm, either a variant of IRACE or otherwise, after executing 30 000 calls to the SUMO simulator. Given that each solution is simulated on a number of different scenarios, the number of solutions evaluated per run is often much lower.

4.5 Algorithms

In our experiments we compare IRACE with the two hybrid variants described above, namely, IRACE+GA and IRACE+DE. In addition, to assess the contribution of the elitist racing mechanism, we also evaluate the classical GA and DE algorithms. Here, we describe the implementation details of these algorithms.

Following the conclusions from a previous work on the TLSP [6], we use default settings for IRACE and the hybrids, except the following. The population size is fixed to 10 individuals (also for GA and DE), the minimum number of traffic scenarios simulated per candidate solution is set to two ($T^{\text{first}} = 2$) and we enable the *deterministic* option that tells IRACE that the only source of uncertainty are the different scenarios and not the simulations themselves. The evolutionary algorithms use a fixed number of simulations per candidate solution. Each solution is simulated on five different training scenarios and its fitness is computed as the mean fitness over the five simulations. In [6], the authors already compared IRACE with differential evolution, genetic algorithm, particle swarm optimization, and a random search; and showed that IRACE obtained the best results with GA and DE being a close second, therefore, we focus here in the comparison of IRACE, GA, DE, and the hybrids.

Our GA implementation uses a ranking method for parent selection and elitist replacement for the next population, that is, the two best individuals of the current population are included in the next one. The operators used are uniform crossover and integer polynomial mutation with 1.0 of probability of crossover and 0.1 of probability of mutation. These parameter settings were found by additional experiments carried out in previous studies [3] to produce a search behavior that is more exploitative rather than explorative, which is more appropriate for the TLSP. Our DE implementation uses a “best/1/bin” strategy with difference factor $F = 0.5$ and probability of crossover 0.5. These are the default parameter values in jMetal [12]. Finally, IRACE+GA and IRACE+DE use the same parameter settings as the GA and DE, respectively.

The GA and DE are implemented in Java using jMetal 5.0 [12]. IRACE and the hybrids are implemented in R.[2] We used IRACE version 2.3 as the baseline.[3]

4.6 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 algorithms 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 hour plus 10 minutes of warm-up, in our case) in order to simulate the peak period in our real-world case study. For the constraints of the TLSP, we apply the same repair method as in [6].

[2] The source code is available at https://github.com/NEO-Research-Group/irace-ea
[3] Available at https://cran.r-project.org/package=irace

The algorithms presented in this paper are non-deterministic algorithms, so we performed 30 independent runs for a fair comparison between them. After the executions, we applied the non-parametric Kruskal-Wallis test with a confidence level of 95% (p-value < 0.05) with Holms's p-value correction to check if the observed differences are statistically significant. In the cases where Kruskal-Wallis test rejects the null hypothesis, we run a single factor ANOVA post hoc test for pairwise comparisons. To properly interpret the results of statistical tests, it is always advisable to report effect size measures. For that purpose, we have also used the non-parametric effect size measure $\widehat{A}_{12}$ statistic proposed by Vargha and Delaney [22]. In the case of minimization problems, such as the TLSP, higher $\widehat{A}_{12}$ values suggest that algorithm 2 has a higher probability of obtaining a better result than algorithm 1, e.g., $\widehat{A}_{12} = 0.3$ indicates that algorithm 2 gets better values than algorithm 1 in 30% of the runs.

The experiments were run on a cluster of 16 machines with Intel Core2 Quad processors Q9400 at 2.66 GHz and 4 GB memory and 3 machines equipped with three 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 Results

To give an in-depth view of the performance of our hybrid algorithms against the standard ones, we will analyze their performance in several sets of scenarios (training and testing). With this, we want to present the competitiveness of our proposal and give a solution to the TLSP.

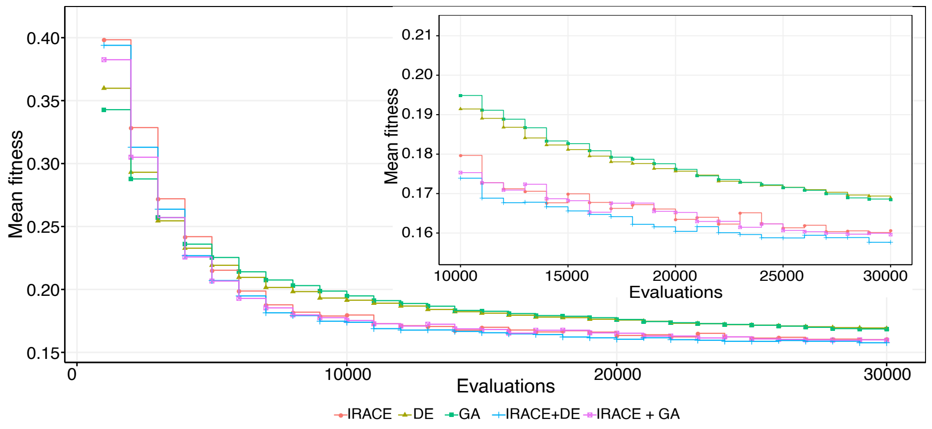

Fig. 2. Mean fitness of the best solutions found so far within each run, as estimated by each algorithm at each moment of its execution on traffic scenarios from the training set. Results in the range [10,000, 30,000] are magnified.

Table 1. Results of the $\widehat{A}_{12}$ test for the evaluation of the last solutions found over training scenarios. Probability that the algorithm (column) is better than another algorithm (row). We highlight in bold the values when the algorithm in the column is better than the algorithm in the row.

	IRACE	IRACE+DE	IRACE+GA	GA	DE
IRACE	—	**0.6711**	**0.6100**	**0.6067**	0.4933
IRACE+DE	0.3289	—	0.3956	**0.5122**	0.3556
IRACE+GA	0.3900	**0.6044**	—	**0.5478**	0.4267
GA	0.3933	0.4878	0.4522	—	0.3811
DE	**0.5067**	**0.6444**	**0.5733**	**0.6189**	—

5.1 Training Set

During the training phase, each algorithm performs a maximum of 30,000 simulations. Figure 2 shows the best fitness obtained, over the number of simulations, averaged over 30 runs of each algorithm. We can see that, in general, up to 10,000 simulations, all the algorithms improve significantly the quality of their solutions, but after this number of simulations, the improvement slows down. Although the GA and DE obtain the best results up to 5,000 simulations, they are quickly overtaken by IRACE and its hybrid variants. Figure 2 also shows in more detail the differences, starting from 10,000 simulations, between IRACE, IRACE+DE and IRACE+GA. We can notice that IRACE+DE consistently obtains the lowest mean fitness, while IRACE and IRACE+GA show a similar result. The plot also shows that the fitness reported by the IRACE hybrids sometimes increases due to the racing procedure performing additional simulations to refine the estimation of the fitness.

We have performed an $\widehat{A}_{12}$ test at the end of the training execution (30,000 simulations) to check if IRACE+DE is indeed better than the other algorithms. Table 1 shows the results of the $\widehat{A}_{12}$ test among the different algorithms, where each value gives the probability of the algorithm in the column returning a better solution than the one in the row. The test indicates that GA is better than the rest of the algorithms. However, we also look at the other statistics shown in Table 2. Although GA has better median than IRACE+DE's only by 10^{-4}, the standard deviation of IRACE+DE is 3.6 times less than GA. Thus, we conclude that IRACE+DE is more robust than GA. EAs obtain lower mean and median, while IRACE reports solutions with smaller variability. These results support our approach to hybridizing IRACE with EAs to obtain good quality robust solutions. Particularly, IRACE+DE looks like a good option if we want to apply these features.

5.2 Testing Set

The above reported statistics were obtained after evaluating the final solutions on the same scenarios used during optimization, but the training scenarios will never arise exactly in the real-world. We evaluate again the solutions on the 30 testing scenarios to properly assess their quality in unseen scenarios. Figure 3 shows the

Table 2. Statistics of each algorithm from the best solutions obtained in the 30,000 simulation of the training. We mark in bold the lower value of each metric.

Algorithm	Mean	Median	STD Dev.
IRACE+DE	**0.1585**	0.1563	0.0101
IRACE+GA	0.1597	0.1590	0.0076
IRACE	0.1621	0.1615	**0.0064**
GA	0.1684	**0.1562**	0.0364
DE	0.1689	0.1596	0.0215

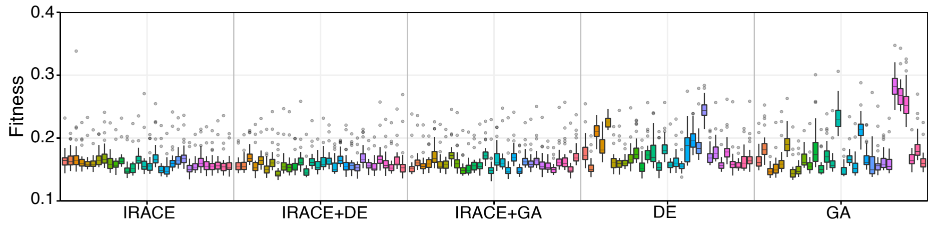

Fig. 3. Fitness of the solutions obtained by the 5 algorithms. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the test set.

boxplots of each independent execution in each algorithm. EAs have the highest variability, while the other three algorithms have more robust boxplots.

To better compare the different algorithms, we summarize the mean, median and standard deviation (see Table 3) between the different independent runs. IRACE+DE gets the best results in each of these metrics, followed by IRACE and IRACE+GA, which are very similar, and the last ones, the EAs. This is a great result for IRACE+DE because, as it was proved in the training results, remarks the competitiveness of the algorithm also in the testing phase.

Anyway, the algorithms using IRACE obtain very similar results. This makes us wonder if there are significant differences between them. To study this, we perform a Wilcoxon rank-sum test between the algorithms to check if there are significant differences. Table 4 shows the p-values reported by the test. As we expected, IRACE+DE has significant differences compared to IRACE and EAs. This result support our working hypothesis: including EAs (specifically a DE) into IRACE can improve the performance. IRACE+GA and IRACE do not offer significant differences between them, which is not a bad result either, since at least the hybrid algorithm reaches a similar performance to IRACE. Lastly, EAs have significant differences with the others algorithms.

Finally, we perform an $\widehat{A}_{12}$ test to see if our hybrid algorithms (especially IRACE+DE) effectively beat the other competitors. Table 5 shows the results for the $\widehat{A}_{12}$ test. We observe that IRACE+DE is better than standard IRACE 53.62% of the time, and 66.17% better than evolutionary ones. While IRACE+GA is 51.33% of the time better than IRACE and 64.34% better than the evolutionary ones. These differences are in favour of our approach. After this experimentation, we can conclude that hybridizing IRACE with evolutionary algorithms is a viable and competitive option. With this idea, we join the best

Table 3. Statistics of each algorithm from the best solutions obtained in the testing phase. We mark in bold the lower value of each metric.

Algorithm	Mean	Median	STD Dev
IRACE+DE	**0.1607**	**0.1571**	**0.0175**
IRACE+GA	0.1620	0.1577	0.0184
IRACE	0.1623	0.1581	0.0184
GA	0.1793	0.1630	0.0407
DE	0.1769	0.1676	0.0275

Table 4. Wilcoxon Test p-value of the testing set with Holm correction.

	IRACE	DE	IRACE+DE	GA
DE	< 2e−16	—	—	—
IRACE+DE	**0.0237**	< 2e−16	—	—
GA	3.5e−11	**0.0011**	< 2e−16	—
IRACE+GA	0.3284	< 2e−16	0.2122	5.3e−13

of both types of algorithms obtaining a powerful and robust algorithm, which allows us to find better solutions for TLSP than the commonly used algorithms.

5.3 Impact in Real World

The previous analysis has focused on the fitness function, an approximation which encompasses some knowledge of traffic flow to guide the search, however, it is quite complex to extract useful information for the domain's expert. Therefore in this section, we study the main traffic and environmental indicators which give the domain's expert more information about the solution.

In a real-world problem, it is desirable to analyze the impact that a representative solution of the different algorithms would have in a real environment. We choose one solution from each algorithm, as a typical traffic light plan as follows: (i) we calculate the mean of the fitness obtained in the 30 scenarios of the testing set by each of the 30 solutions of each algorithm, (ii) we order upwards these mean fitness for each algorithm, (iii) we select, as the representative, the solution whose fitness value is at the 16th position, that is, immediately following the median solution. We cannot select the median because there are an even number of solutions (30).

We simulate again each of the representative solutions in the test scenarios but allowing all the vehicles to reach their destination. This means that the fitness values are not penalized, hence, they are smaller than those reported in the previous boxplots. With these new simulations, we obtain 34 different traffic and environmental measures of the 30 testing scenarios. Figure 4 shows some of the most important measures for each algorithm. In all measures, hybrid algorithms get the best results. IRACE+DE obtains the best average values in *MeanTravelTime* and *MeanWaitingTime*, while IRACE+GA has the lowest *MaxTravelTime* and *MaxWaitingTime*. In practice, if we implement the IRACE+DE solution, citizens would complete the journeys in less time (329.60s) and with less waiting time at intersections (88.84s). If IRACE+GA solution were implemented, the *MeanTravelTime* is higher than in IRACE+DE solution, but in the worst case (*MaxTravelTime* and *MaxWaitingTime*), IRACE+GA obtains

Table 5. Results of the $\widehat{A}_{12}$ test for testing. Probability that the algorithm (column) is better than another algorithm (row). We highlight in bold the values when the algorithm in the column is better than the algorithm in the row.

	IRACE	IRACE+DE	IRACE+GA	GA	DE
IRACE	—	**0.5362**	**0.5133**	0.4066	0.3198
IRACE+DE	0.4638	—	0.4780	0.3820	0.2946
IRACE+GA	0.4867	**0.5220**	—	0.3985	0.3147
GA	**0.5934**	**0.6180**	**0.6015**	—	0.4506
DE	**0.6802**	**0.7054**	**0.6853**	**0.5494**	—

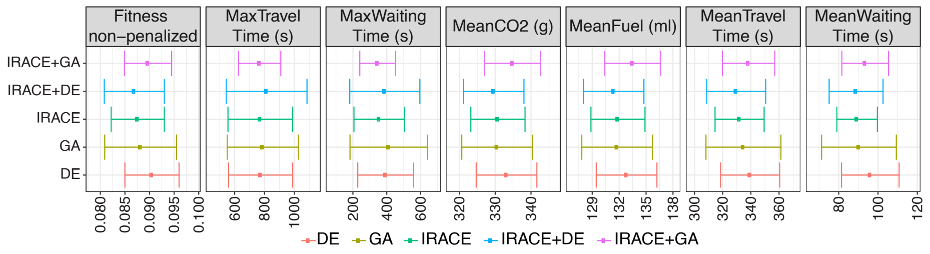

Fig. 4. Traffic measures per vehicle. Mean values (and standard deviation) over 30 test traffic scenarios of the median solutions for the five algorithms.

the minimum values. On the complete opposite side, we have GA and DE, with the worst results of the comparison.

Regarding the environmental impact (fuel consumption and CO2 emissions), IRACE+DE gives the most eco-friendly solutions. Nowadays, pollution is a serious issue in many cities, so offering solutions that reduce emissions and fuel is of vital importance in today's cities.

With all these results, we can confirm that better TLPs result in less CO2 emissions, less fuel consumption, and less journey time for the citizen. Our hybrid proposals, specially IRACE+DE, not only offer competitive solutions from a scientific point of view, but it would also have a positive impact in the city at multiple levels both environmental and for the quality of life of the citizens.

6 Conclusions

In this article, we have proposed new hybrid algorithms combining IRACE with two evolutionary algorithms: GA and DE. These new hybrid algorithms are ideally suited for black-box numerical optimization problems under uncertainty, by using evolutionary operators designed for numerical optimization to generate better solutions, while handling uncertainty by means of the elitist racing strategy in IRACE. We have used these hybrid algorithms (IRACE+DE and IRACE+GA), IRACE, a GA, and a DE, to solve the TLSP using the real instance of Málaga, Spain, and the SUMO traffic simulator to evaluate the solutions. The results obtained in the experiments confirm the competitiveness of the hybridization strategy. Both hybrid algorithms offer better results than GA (60%

of the time) and DE (70% of the time) on realistic traffic scenarios. Particularly, IRACE+DE returns the best results during the testing, being also competitive during the training. Besides, we have seen the impact that the solutions would have on the city. Our hybridization strategies obtain the best results in travel times, fuel consumption, CO2 emissions, etc. These results reinforce our algorithmic proposal and show the efficiency that IRACE+DE and IRACE+GA obtain when solving a real-world problem.

As future work, we will consider other algorithms and operators that have proven to be effective in numerical optimization problems for hybridization with IRACE. Although preliminary experiments hybridizing IRACE with JADE [23], a well-known variant of DE, did not improve the results over the IRACE+DE proposed in this paper, we plan to perform a deeper analysis of IRACE+JADE to extract any insights about the behavior of the new hybrid algorithms. Also, we plan to test our hybrid algorithms on other black-box numerical optimization problems under uncertainty to further validate our results.

Acknowledgements. This research was partially funded by the University of Málaga, Andalucía Tech and the project TAILOR Grant #952215, H2020-ICT-2019-3. C. Cintrano is supported by a FPI grant (BES-2015-074805) from Spanish MINECO. M. López-Ibáñez is a "Beatriz Galindo" Senior Distinguished Researcher (BEAGAL 18/00053) funded by the Ministry of Science and Innovation of the Spanish Government. J. Ferrer is supported by a postdoc grant (DOC/00488) funded by the Andalusian Ministry of Economic Transformation, Industry, Knowledge and Universities.

References

1. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO - Simulation of Urban MObility: An overview. In: SIMUL 2011, The Third International Conference on Advances in System Simulation, pp. 63–68, ThinkMind, Barcelona, Spain (2011)
2. Blum, C., Raidl, G.R.: Hybrid Metaheuristics—Powerful Tools for Optimization. Artificial Intelligence: Foundations, Theory, and Algorithms, Springer, Springer, Berlin, Germany (2016)
3. Bravo, Y., Ferrer, J., Luque, G.J., Alba, E.: Smart mobility by optimizing the traffic lights: A new tool for traffic control centers. In: Alba, E., Chicano, F., Luque, G.J. (eds.) Smart Cities (Smart-CT 2016), pp. 147–156, LNCS, Springer, Cham, Switzerland (2016)
4. Deb, K., Agrawal, S.: A niched-penalty approach for constraint handling in genetic algorithms. In: Dobnikar, A., Steele, N.C., Pearson, D.W., Albrecht, R.F. (eds.) Artificial Neural Nets and Genetic Algorithms (ICANNGA-99), pp. 235–243, Springer Verlag (1999)
5. Ferrer, J., García-Nieto, J., Alba, E., Chicano, F.: Intelligent testing of traffic light programs: Validation in smart mobility scenarios. Mathematical Problems in Engineering **2016**, 1–19 (2016)
6. Ferrer, J., López-Ibáñez, M., Alba, E.: Reliable simulation-optimization of traffic lights in a real-world city. Applied Soft Computing **78**, 697–711 (2019)
7. García-Nieto, J., Alba, E., Olivera, A.C.: Swarm intelligence for traffic light scheduling: Application to real urban areas. Engineering Applications of Artificial Intelligence **25**(2), 274–283 (Mar 2012)

8. García-Nieto, J., Olivera, A.C., Alba, E.: Optimal cycle program of traffic lights with particle swarm optimization. IEEE Transactions on Evolutionary Computation **17**(6), 823–839 (Dec 2013)
9. Heidrich-Meisner, V., Igel, C.: Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In: Danyluk, A.P., Bottou, L., Littman, M.L. (eds.) Proceedings of the 26th International Conference on Machine Learning, ICML 2009, pp. 401–408, ACM Press, New York, NY (2009)
10. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements **5**(3-4), 128–138 (2012)
11. López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., Birattari, M.: The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives **3**, 43–58 (2016)
12. Nebro, A.J., Durillo, J.J., Vergne, M.: Redesigning the jMetal multi-objective optimization framework. In: Laredo, J.L.J., Silva, S., Esparcia-Alcázar, A.I. (eds.) GECCO (Companion), pp. 1093–1100, ACM Press, New York, NY (2015)
13. Péres, M., Ruiz, G., Nesmachnow, S., Olivera, A.C.: Multiobjective evolutionary optimization of traffic flow and pollution in Montevideo, Uruguay. Applied Soft Computing **70**, 472–485 (2018)
14. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Springer, New York, NY (2005)
15. Sánchez, J., Galán, M., Rubio, E.: Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife. IEEE Transactions on Evolutionary Computation **12**(1), 25–40 (2008)
16. Sánchez-Medina, J.J., Galán-Moreno, M.J., Rubio-Royo, E.: Traffic signal optimization in "La Almozara" district in Saragossa under congestion conditions, using genetic algorithms, traffic microsimulation, and cluster computing. IEEE Transactions on Intelligent Transportation Systems **11**(1), 132–141 (Mar 2010), ISSN 1524-9050
17. Stolfi, D.H., Alba, E.: Red swarm: Reducing travel times in smart cities by using bio-inspired algorithms. Applied Soft Computing **24**, 181–195 (2014)
18. Stolfi, D.H., Alba, E.: An evolutionary algorithm to generate real urban traffic flows. In: Puerta, J.M., Gámez, J.A., Dorronsoro, B., Barrenechea, E., Troncoso, A., Baruque, B., Galar, M. (eds.) Advances in Artificial Intelligence, CAEPIA 2015, LNCS, vol. 9422, pp. 332–343, Springer (2015)
19. Syswerda, G.: Uniform crossover in genetic algorithms. In: Schaffer, J.D. (ed.) Proc. of the Third Int. Conf. on Genetic Algorithms, pp. 2–9, Morgan Kaufmann Publishers, San Mateo, CA (1989)
20. Teklu, F., Sumalee, A., Watling, D.: A genetic algorithm approach for optimizing traffic control signals considering routing. Computer-Aided Civil and Infrastructure Engineering **22**(1), 31–43 (Jan 2007)
21. Teo, K.T.K., Kow, W.Y., Chin, Y.K.: 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, pp. 172–177, IEEE, IEEE Press (2010)
22. Vargha, A., Delaney, H.D.: A critique and improvement of the CL common language effect size statistics of McGraw and Wong. Journal of Educational and Behavioral Statistics **25**(2), 101–132 (2000)
23. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation **13**(5), 945–958 (2009)