Hybridization of Evolutionary Operators with Elitist Iterated Racing for the Simulation Optimization of Traffic Lights Programs

Christian Cintrano cintrano@lcc.uma.es
ITIS Software, University of Malaga, Bulevar Louis Pasteur 35, 29010 Malaga, Spain.

Javier Ferrer ferrer@lcc.uma.es
ITIS Software, University of Malaga, Bulevar Louis Pasteur 35, 29010 Malaga, Spain.

Manuel López-Ibáñez manuel.lopez-ibanez@uma.es
ITIS Software, University of Malaga, Bulevar Louis Pasteur 35, 29010 Malaga, Spain.

Enrique Alba eat@lcc.uma.es
ITIS Software, University of Malaga, Bulevar Louis Pasteur 35, 29010 Malaga, Spain.

Abstract

In the traffic light scheduling problem the evaluation of candidate solutions requires the simulation of a process under various (traffic) 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 shown that combining IRACE with evolutionary operators is effective for this task due to the power of evolutionary operators in numerical optimization. In this paper, we further explore the hybridization of evolutionary operators and the elitist iterated racing of IRACE for the simulation-optimization of traffic light programs. We review previous works from the literature to find the evolutionary operators performing the best when facing this problem to propose new hybrid algorithms. We evaluate our approach over a realistic case study derived from the traffic network of Málaga (Spain) with 275 traffic lights that should be scheduled optimally. The experimental analysis reveals that the hybrid algorithm comprising IRACE plus differential evolution offers statistically better results than the other algorithms when the budget of simulations is low. In contrast, IRACE performs better than the hybrids for high simulations budget, although the optimization time is much longer.

Keywords

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

1 Introduction

The increase in traffic flow has become a severe problem in most cities of the world, leading to traffic jams, accidents and air pollution. Modern cities regulate traffic prominently using traffic lights. The larger the metropolitan area, the higher the number of traffic lights needed to regulate the traffic flow. Optimal management of traffic minimizes journey times, reduce fuel consumption and harmful emissions. In some cities, legal and technical limitations preclude real-time traffic light control and mandate an optimal planning of traffic light programs (TLPs) given the particular traffic flows of that city (Teklu et al., 2007; Sánchez et al., 2008; Teo et al., 2010; Sánchez-Medina et al.,

2010; García-Nieto et al., 2012; Putha et al., 2012; García-Nieto et al., 2013; Stolfi and Alba, 2014, 2015; Bravo et al., 2016; Ferrer et al., 2016; Péres et al., 2018; Ferrer et al., 2019; Cintrano et al., 2021).

Simulation-optimization is a common approach for finding optimized TLPs that consists in simulating each candidate TLP under a number of scenarios generated from real traffic data. These traffic scenarios represent uncertainty about the real-time traffic conditions, and good solutions should not only achieve good fitness value but also show reliability, i.e., low variance across scenarios. Given the computational effort of such simulations, there is a trade-off between the number of scenarios used to estimate the fitness of each candidate solution and the total number of candidate solutions evaluated. Higher number of scenarios lead to more reliable solutions, while evaluating higher number of solutions may allow finding even better ones.

Previous work (Ferrer et al., 2019) has shown that IRACE (López-Ibáñez et al., 2016) is able to find high-quality and low-variance TLPs 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 algorithmic parameters in optimization and machine learning, where each configuration must be evaluated on a number of training problem instances or datasets and the algorithm themselves are often stochastic. In IRACE, new solutions are generated from a probabilistic distribution, thus resembling a univariate estimation of distribution algorithm (Krejca, 2019). This solution generation approach works well for the fitness landscapes typically found in algorithm configuration (Pushak and Hoos, 2018), which consist of a mix of categorical and numerical parameters with dependencies and constraints among them. However, it is expected that other approaches perform better for other problem types, specially for problem landscapes consisting only of numerical decision variables, as in the case of the optimization of traffic lights programs.

Following this intuition, Cintrano et al. (2021) carried out an initial comparison between IRACE, a genetic algorithm, using uniform crossover (Syswerda, 1989) and integer polynomial mutation (Deb and Agrawal, 1999), a differential evolution (DE) algorithm, using the "DE/best/1/bin" strategy (Price et al., 2005), and variants (*hybrids*) of IRACE replacing its solution generation approach with the evolutionary operators used by the GA and the DE under comparison. Experimental results indicated that, although the original IRACE outperforms the GA and DE algorithms, both hybrids of IRACE+GA and IRACE+DE outperformed IRACE.

In this paper, we further explore the hybridization of evolutionary operators from evolutionary algorithms (EAs) and the elitist iterated racing of IRACE for the simulation-optimization of traffic light programs. First, we review previous works from the literature to find the evolutionary operators performing the best when facing the optimization of traffic light programs. Most previous works used an integer vector as solution encoding (Bravo et al., 2016; Ferrer et al., 2016, 2019; García-Nieto et al., 2012, 2013; Péres et al., 2018; Teo et al., 2010). A few other works used binary Gray code encoding for solution representation (Putha et al., 2012; Sánchez et al., 2008; Sánchez-Medina et al., 2010; Teklu et al., 2007). The solution representation restricts the evolutionary operators that can be used for searching the optimal solution, and these in turn determine how the search is carried out. As a result of our review, we consider in this paper hybrids of IRACE with the following operators: Integer Uniform Crossover (Bravo et al., 2016; Ferrer et al., 2016, 2019), Binary Uniform Crossover (Teklu et al., 2007; Putha et al., 2012), Simulated Binary Crossover (SBX) (Péres et al., 2018), Binary Two-points Crossover (Sánchez et al., 2008; Sánchez-Medina et al., 2010), Integer

Polynomial Mutation (Bravo et al., 2016; Ferrer et al., 2016, 2019; Péres et al., 2018), Binary Uniform Mutation (bit-flip) (Sánchez et al., 2008; Sánchez-Medina et al., 2010; Teklu et al., 2007; Putha et al., 2012), and Differential Mutation (Ferrer et al., 2016; García-Nieto et al., 2012, 2013). These hybrids of IRACE and evolutionary operators differ conceptually from previous approaches hybridizing EAs and racing (Heidrich-Meisner and Igel, 2009), which perform an independent race to carry out the evaluation and environmental selection step at each generation of the EA. The hybrids of IRACE studied here replace the solution generation mechanism of IRACE with evolutionary operators, but keep other IRACE components, such as the elitist iterated racing (López-Ibáñez et al., 2016), which is not a sequence of independent races.

Our work is motivated by the real-world optimisation of traffic light programs in the city of Malaga, Spain, hence, we use as a case study, a traffic network and 60 traffic flow scenarios derived from real data. Unfortunately, such data is rarely made publicly available, with Ferrer et al. (2019) being a notable exception, and we did not find, among all the works reviewed, any other dataset that could be used for our analysis.

In summary, the main contributions of this work are:

- We study hybrid algorithms that combine evolutionary operators, extracted from the relevant literature, with elitist iterated racing.
- We offer an in-depth analysis of 5 hybrid algorithms with 6 different evolutionary operators and two solution encodings (integer and binary Gray code).
- We optimize the traffic light plan of a real city (Malaga in Spain) using detailed micro-simulations.
- We identify hybrid algorithms that obtain high-quality and reliable solutions, i.e., low-fitness solutions with also low fitness variance across scenarios.

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 lights regulate traffic flow in the intersections between two o more roads and between roads and pedestrian crossings. In most cases, traffic lights are coordinated in phases: green, yellow and red. Within the same intersection, when a traffic light is in green, some others must be in red, and all traffic lights must change color simultaneously following a sequence of colors that repeats over time. The sequence of phases constitutes the program of an intersection. Although the colors of a program are predefined, we can influence the flow of traffic and pedestrians by adjusting the duration of each color phase and the start of program, relative to other intersections (Wei et al., 2019). The collection of programs, their phase durations and start times for all intersections constitute a traffic light program (TLP) of a city.

The large number of valid TLPs that may be possible in large cities require automatic tools to generate an optimal TLP, which motivates the Traffic Light Scheduling Problem (TLSP) (Little, 1966; Sánchez et al., 2008; García-Nieto et al., 2013; Sánchez-Medina et al., 2010). The main objective in this problem is to find a 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.

2.1 The Traffic Light Scheduling Problem (TLSP)

Let us define the TLSP as follows. Let $P = \{To_1, I_1, \ldots, To_n, I_n\}$ be a candidate TLP, where each pair To_i, I_i represent an intersection i, respectively, its offset time and its 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.

By default, the program of all intersections start at the same time. However, we also optimize an offset time at each intersection ($To_i \in [To_{\min}, To_{\max}]$) 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 the 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 green waves, series of coordinated traffic lights that produce a continuous traffic flow over several intersections in one main direction.

The objective is to find a TLP P' that minimizes a fitness function $f\colon \Gamma \to \mathbb{R}$ such that:

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

where Γ is the space of all possible TLPs.

The definition of the fitness function must account for several performance metrics that characterize good TLPs and may be calculated from the information regarding the flow of vehicles provided by a traffic simulator. In the simulator, 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. In addition, a long phase duration may lead to a collapse of the intersection, i.e., traffic stops flowing in some direction. TLPs should prioritize those phases with more green lights on the directions with a high number of vehicles circulating by maximizing 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.

Taking the above criteria into account, 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 several studies (García-Nieto et al., 2012, 2013; Ferrer et al., 2019; Cintrano et al., 2021).

2.2 Repair Procedure

Real-world instances of the TLSP often present additional constraints. 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. Non-fixed phases have a minimum duration of $\varphi_{\min}$.

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 $[Tp_{\min}, Tp_{\max}]$.

To ensure that candidate solutions are valid, we apply a repair procedure (Ferrer et al., 2019) that is used by all the algorithms before simulating a solution. 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.

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 (Blum and Raidl, 2016). 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 algorithms.

3.1 IRACE

IRACE (López-Ibáñez et al., 2016) is a well-known tool for automatic (hyper-)parameter configuration of optimization and machine learning algorithms. However, IRACE

can be seen as an optimization method for mixed-integer black-box problems under uncertainty, and, hence, it may be used to tackle simulation-optimization problems, such as the TLSP (Ferrer et al., 2019).

Algorithm 1 briefly presents IRACE applied to the TLSP. Initially, a set of solutions Θ_1 are sampled uniformly at random. Then a race is performed to identify the best (elite) solutions among the initial set. These elite solutions are used to update a sampling model from which new solutions are generated, in a similar fashion as in univariate estimation of distribution algorithms. New and elite solutions together form a new population that is raced again. This process is iterated until a maximum budget of simulations is exhausted.

Algorithm 1 Pseudocode of IRACE

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

1: $t \leftarrow 1$
2: $\Theta_t \leftarrow$ SampleUniformRandomPopulation
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}

A key component of IRACE is the racing procedure. Within a race, each solution is simulated at least T^{first} times on a sequence of different traffic scenarios. In elitist iterated racing, the sequence of scenarios changes between races as follows. The first scenario of the sequence has not been used in previous races, then the sequence of scenarios from the previous race is randomly shuffled and, finally, if additional scenarios are needed for this race, they are randomly taken from the set of scenarios not used yet. If an elitist solution is evaluated on a scenario seen in a previous race, its fitness is recovered from a cache memory and no simulation is required. After all solutions are evaluated on T^{first} scenarios, an elimination step removes the worst performing solutions from the race. In the TLSP, we identify the current best solution of a race according to its mean performance on the traffic scenarios seen so far within this particular race. Then, we use the pairwise paired Student's t-test to eliminate from the race those solutions that perform significantly worse than the current best one. The use of the t-test implies the optimization of the mean value, however, the test is only used as a heuristic that guides the elimination of solutions equivalent to the calculation of confidence intervals around the estimated mean (Birattari et al., 2002). Its use does not justify any claims of statistical significance, which require a proper statistical analysis of the obtained results. In elitist racing, an elite solution cannot be eliminated from the race until the contender has been evaluated in as many scenarios as the elite solution has been evaluated in this and previous races, thus preventing that a lucky new solution evaluated in few scenarios so far eliminates a robust elite evaluated in many scenarios in the previous race. After the elimination step, the race continues by evaluating every surviving solution in one additional scenario and carrying out a new elimina-

tion step. The race stops once a minimum number of solutions (*minSurv*) remains alive in the race, the budget assigned to the race is exhausted, or multiple elimination tests fail to eliminate any solution. If more solutions than *minSurv* are alive at the end of the race, they are sorted according to decreasing mean value and the first *minSurv* solutions constitute the new elite solutions.

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 expected 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 describe the hybrid IRACE 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 the sampling step with the mating selection and variation steps of an Evolutionary Algorithm, i.e., selection of parents, generation of new individuals from them (crossover), and modification of those new individuals (mutation). Apart from this modification, the other steps in IRACE, in particular, the racing procedure described above, remain intact.

We now describe the evolutionary mating selection and variation steps used in the hybrid IRACE. 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 randomly selected from Θ^{elite}. However, the size of Θ^{elite} varies at each iteration of IRACE, and may be insufficient for the number of parents required by the evolutionary operators, e.g., it is possible that a single solution survives after a race. 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. The number of selected parents depends on the evolutionary operators used. For example, a uniform crossover needs two parents, while the differential evolution mutation needs four.

Once the required number of parents are selected, the crossover operator is applied with a certain probability to generate a new solution. One solution is generated per operator application. If no crossover operator is applied, we simply select the first parent as the new solution for subsequent mutation. The mutation operator is then applied to the new solution with a probability of mutation that determines the strength of the mutation, ensuring that at least one mutation per solution occurs. The resulting solution is added to the set of new solutions Θ^{new}.

3.3 Solution Encoding

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 (To_i) or the duration of a phase (φ_{ij}). The repair mechanism described in Section 2.2 is applied to enforce the total cycle time constraints (Eq. 4). Figure 1 shows an example of the solution representation.

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

Figure 1: Example of integer solution representation in the TLPS.

Instead of the integer representation above, some of the evolutionary operators used in the TLSP literature require a binary representation. A solution in integer representation can be converted to binary representation as follows. First, we shift the integer value of each variable so that its lowest possible value corresponds to zero, that is, phase duration variables $\varphi_{ij} \in [\varphi_{\min}, Tp_{\max}] \subset \mathbb{N}^+$ are shifted to the range $[0, Tp_{\max} - \varphi_{\min}]$ and offset decision variables $To_i \in [To_{\min}, To_{\max}]$ are shifted to $[0, To_{\max} - To_{\min}]$. Finally, the number of bits needed for each decision variable is computed as:

$$N^{\text{bits}}(x) = \lceil \log_2(1 + x_{\max} - x_{\min}) \rceil \tag{7}$$

where $x_{\max} = Tp_{\max}$ and $x_{\min} = \varphi_{\min}$ when x is a variable that represents a phase duration φ_{ij}, whereas $x_{\max} = To_{\max}$ and $x_{\min} = To_{\min}$ when x is a variable that represents an offset To_i. This ensures that we use the minimum number of bits required for the representation of solutions.

Figure 2 shows the binary representation of the solution shown in Fig. 1 when $To_{\min} = -30$ and $To_{\max} = 30$, i.e., the binary values are shifted to the range $[0, 60]$.

Intersection 1				…	Intersection n			
To_1	φ_{11}	…	φ_{1j}	…	To_n	φ_{n1}	…	φ_{nj}
110111	0010100	…	0100011	…	101101	0111100	…	0011110

Figure 2: Equivalent binary representation of the solution shown in Fig. 1.

Before (after) applying an evolutionary operator that requires binary representation we encode (decode) the integer solution to (from) binary. After decoding from binary, decision variables are clipped to their range and the repair procedure is applied.

Additionally, a few works on the TLSP considered a binary Gray code encoding (Sanchez et al., 2005), which is an ordering of the binary numeral system such that two successive integer values differ in only one bit. Therefore, our experiments consider three representations in total: integer, binary and Gray code encodings.

3.4 Evolutionary Operators Used by Hybrid Algorithms

Our main purpose is to assess whether the use of evolutionary operators within IRACE may produce better solutions for the TLSP than the general-purpose sampling procedure of IRACE. For this reason, we have reviewed previous works solving the TLSP with the aim of finding appropriate evolutionary operators to hybridize with IRACE. In total, we have found ten papers applying evolutionary algorithms to the TLSP. Of those, six works have considered an integer representation, four works have considered a binary representation and three have considered a gray code one. Table 1 summarises the evolutionary operators used in the literature for solving the TLSP. Although these evolutionary operators are well-known, we briefly describe them in the following for completeness:

Uniform Crossover (Syswerda, 1989) recombines two parent solutions by selecting a decision variable from either parent according to a probability (typically 0.5). With a binary representation Spears and De Jong (1991), the selection is made per bit.

Two Points Crossover (Holland, 1975)randomly picks two variables (or two bits in binary representation). These variables are called the crossover points. The new solution takes the values of the variables between the crossover points from the

Table 1: Evolutionary operators used for solving TLSP

Operators	**Binary / Gray code**	**Integer**
Uniform Crossover	Teklu et al. (2007); Putha et al. (2012)	Bravo et al. (2016); Ferrer et al. (2016); Ferrer et al. (2019)
Two-points Crossover	Sánchez et al. (2008); Sánchez-Medina et al. (2010)	
Simulated Binary Crossover (SBX)		Péres et al. (2018)
Uniform Mutation	Sánchez et al. (2008); Sánchez-Medina et al. (2010); Teklu et al. (2007); Putha et al. (2012)	
Polynomial Mutation		Bravo et al. (2016); Ferrer et al. (2016, 2019); Péres et al. (2018)
Differential Mutation		Ferrer et al. (2016); García-Nieto et al. (2012, 2013)

second parent and the values of the other variables from the first parent. (In our hybrid algorithm, all crossover operators generate a single solution).

Simulated Binary Crossover (Deb and Agrawal, 1995) SBX uses two parents and apply the blending operator variable by variable to create a child solution. The operator involves a parameter, called the distribution index (η_i), which is kept fixed to a non-negative value throughout a simulation run. If a large value of η_i is chosen, the resulting offspring solutions are close to the parent solutions. On the other hand, for a small value of η_i, solutions away from parents are likely to be created.

Uniform Mutation (Bit-flip) (Garnier et al., 1999) visits every binary decision variable of a solution and flips its value with a given probability (typically small). This operator is not applied when using integer representation.

Polynomial Mutation (Deb and Agrawal, 1999) In this operator, a polynomial probability distribution is used to perturb a solution in a solution's vicinity. The probability distribution in both left and right of a variable value is adjusted so that no value outside a specified range is created by the mutation operator. The effect of this operator is to perturb the current decision variable values (parent) to a neighbouring variable values (child).

Differential Mutation consists of the construction of a new solution from an original one, using other three parents which must be distinct from each other. Solutions are moved around in the search-space by using simple mathematical formulae to combine the positions of existing solutions from the population. There are a number of well-known mutation schemes or strategies in differential evolution, in this

work we use "DE/best/1" (Price et al., 2005). Although traditionally called a mutation, it is actually a combination of crossover and mutation.

With the aim of evaluating the best hybrid algorithms for the TLSP, we have designed five hybrid variants by combining the evolutionary operators found in the TLSP literature:

IRACE+DE uses differential mutation ("DE/best/1/bin").

IRACE+GA uses uniform crossover and integer polynomial mutation.

IRACE+SBX uses SBX and integer polynomial mutation.

IRACE+UNIFORM uses binary uniform crossover and binary uniform mutation.

IRACE+2PC uses binary two-points crossover and binary uniform mutation.

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 (Stolfi and Alba, 2015), which encompasses an area of about 3 km^2 with 58 intersections controlled by 275 traffic lights (Fig. 3). Our network model was created from real data about traffic rules, traffic element locations, road directions, streets, intersections, etc.

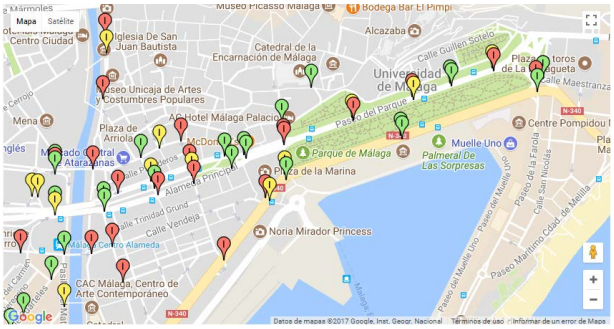

Figure 3: Locations of traffic lights considered in the case of study. The colors show large (red), medium (yellow) and small (green) differences between two different solutions.

We consider 60 different traffic scenarios, which contain the routes and speeds of the vehicles circulating, with an average of 4,827 vehicles (or different vehicle routes) per scenario. These traffic scenarios were generated by applying the Flow Generator Algorithm (FGA) (Stolfi and Alba, 2015) to data extracted from sensors placed at several streets measuring traffic density at various time intervals. This network and the corresponding traffic scenarios have been used in several previous studies (Stolfi and Alba, 2015; Ferrer et al., 2019; Cintrano et al., 2021) and the data is publicly available[1].

[1]Datasets: doi:10.5281/zenodo.6542005

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 final solutions found after the optimization.

For the constraints of the TLSP, we use the values recommended by the City Council of Malaga (Spain). In particular, the pedestrian crosses (Y) last for a fixed time of $4 \times$ *number of lanes* seconds; non-fixed phases have a minimum duration of $\varphi_{\min} = 15$ seconds; the total program time (Tp_i) within each intersection I_i is constrained within $[Tp_{\min}, Tp_{\max}] = [60, 120]$ seconds; and, lastly, offset values are constrained within the time interval $To_i \in [To_{\min}, To_{\max}] = [-30, 30]$ seconds.

4.2 Simulator: SUMO

The quality of a candidate solution (traffic light program) is evaluated through the Simulator of Urban Mobility (SUMO) version 0.22 (Behrisch et al., 2011; Krajzewicz et al., 2012), which is a microscopic road traffic simulator that provides detailed information about vehicles like velocity, fuel consumption, emissions, journey time, waiting time, etc. SUMO is widely used in the TSLP (Tsai et al., 2021; Mahto and Malik, 2021). 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. 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.

4.3 Algorithms

In our experiments, we compare IRACE with the five hybrid variants described above, namely, IRACE+GA, IRACE+DE, IRACE+SBX, IRACE+UNIFORM and IRACE+2PC. Ferrer et al. (2019) already compared non-hybrid IRACE with conventional DE, GA and particle swarm optimization, and concluded that IRACE obtains statistically better results than those conventional algorithms for the TLSP. Similarly, Cintrano et al. (2021) already compared IRACE, IRACE+GA and IRACE+DE with classic DE and GA and concluded that the IRACE variants outperform the classic DE and GA. Therefore, we focus here in the comparison of IRACE and the hybrid algorithms.

A previous analysis of IRACE for the TLSP (Ferrer et al., 2019) concluded that the variable population size of IRACE was not well-suited for this problem and fixed the population size to 10 individuals. The value of *minSurv* is set automatically by IRACE (López-Ibáñez et al., 2016) and in our experiments was between 3 and 4.Moreover, the minimum number of traffic scenarios simulated per candidate solution before the first elimination test was set to $T^{\text{first}} = 2$ instead of the default $T^{\text{first}} = 5$ since many solutions perform very poorly and can be eliminated with just two simulations. Finally, we enable the *deterministic* option that tells IRACE that the only source of uncertainty are the different scenarios and not the simulations themselves.

The hybrid algorithms use the same parameter settings as IRACE except for probability of crossover and mutation. The parameters of each operator are presented in Table 2. IRACE and the hybrids are implemented in R.[2]. We used IRACE version 3.4 as the baseline.[3]

[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

Table 2: Parameters used for each evolutionary operator

Algorithm	Operator	Probability	Operator param.
IRACE+DE	"DE/best/1/bin"	0.5	0.5
IRACE+GA	Uniform crossover	0.5	0.5
	Integer polynomial mutation	0.1	20
IRACE+SBX	SBX	0.5	0.5
	Integer polynomial mutation	0.1	20
IRACE+UNIFORM	Binary uniform crossover	0.5	0.5
	Binary uniform mutation	1/30	20
IRACE+2PC	Binary two-points crossover	0.5	0.5
	Binary uniform mutation	1/30	20

4.4 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 algorithms analyzed in this paper.

We expect that some operators show a faster convergence while others encourage exploration of new solutions. Therefore, we conduct experiments with short and long runs of the algorithms, i.e., stopping each run of an algorithm after executing either $1\,000$ or $10\,000$ calls to the SUMO simulator. Given that each solution is simulated on a variable number of different scenarios, which is dynamically determined by IRACE, the actual number of solutions evaluated per run is often much lower.

During optimization, each call to SUMO simulates traffic until 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.

The algorithms presented in this paper are stochastic, so we perform 30 independent runs with different random seeds for a fair comparison between them. Each run is given the same 30 training scenarios but which scenarios are actually simulated and in which order is randomized by each run of IRACE. After the runs, we applied the non-parametric Wilcoxon 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.

All 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 we will analyze them 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. As indicated above, we have performed two types of experiments, changing the maximum number of calls

to the simulator. In doing so, we want to test not only the quality of the solutions, but also how the different algorithms converge under different number of evaluations.

5.1 Training Set

First, we want to analyse how the algorithms behave in the training phase, i.e., during their execution. Figure 4 shows the average fitness of the best solutions found so far up to a maximum of 1 000 evaluations. We can see that IRACE+DE is ahead of the others algorithms after a few evaluations. The second one is IRACE+SBX. It seems that these two crossover strategies are quite powerful for this problem. On the other hand, IRACE and IRACE+GA have a very similar behaviour. The algorithms using gray code encoding lag behind those using integer encoding. This difference increases as we increase the number of evaluations to 10 000 (see Figure 5). Adding a larger budget shows that IRACE+GA lags behind the other integer encoding algorithms, as was already apparent in the experiments for 1 000.

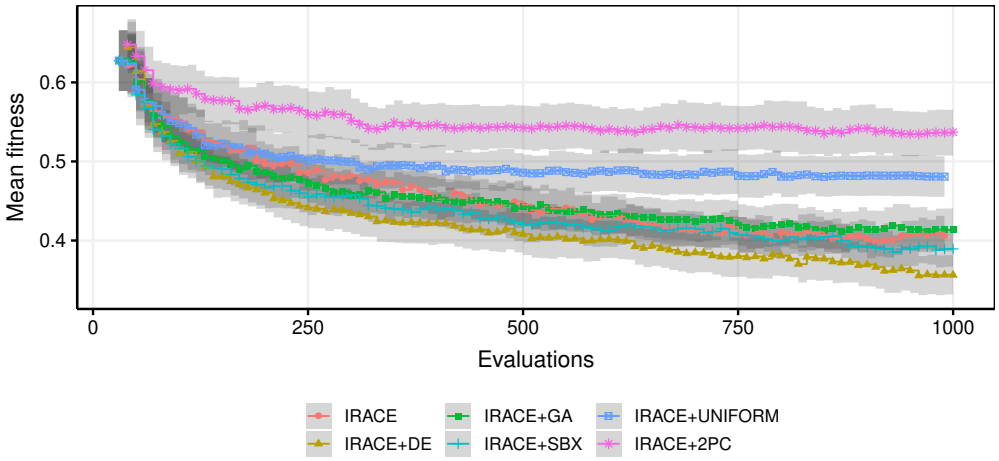

Figure 4: Mean fitness of the best solutions found so far within each run, as estimated by each algorithm at each moment of its execution on traffic scenarios from the training set and a budget of 1 000.

An interesting detail is that, from about 4 000 evaluations onwards, IRACE seems to be in the lead. IRACE's way of improving the population seems to require a larger number of evaluations compared to IRACE+DE. The number of simulations we can perform seems to be the limiting factor in choosing one algorithm or the other. However, these are only training results. The convergence of the algorithm for this set does not imply that it is the best for working with new data sets. To do so, we are going to analyse the results obtained by the best solutions found on the test set.

5.2 Testing Set

The above reported statistics were obtained after evaluating the 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 6 shows the fitness obtained after the re-evaluation of the solutions on the testing scenarios for the algorithm with a budget of 1 000 simulations for each independent run. As already denoted in the training, IRACE+UNIFORM, and IRACE+2PC obtain worse solutions than the other algo-

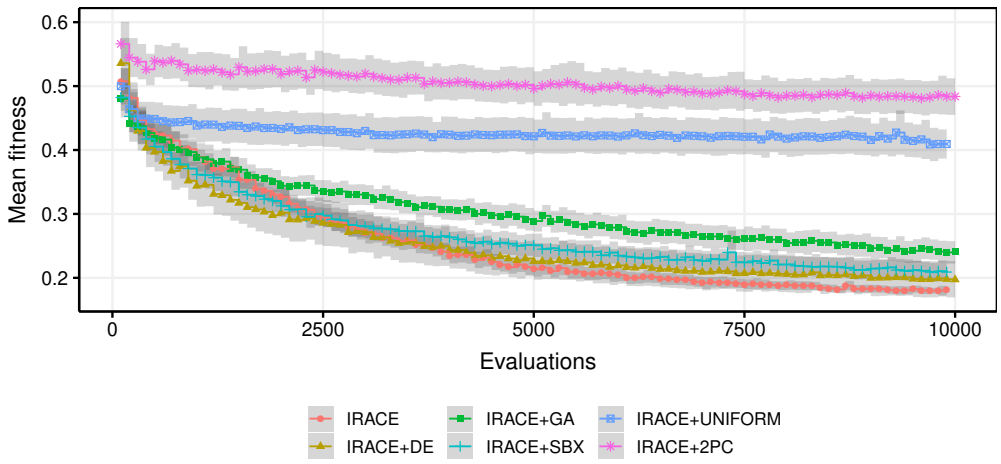

Figure 5: Mean fitness of the best solutions found so far within each run, as estimated by each algorithm at each moment of its execution on traffic scenarios from the training set and a budget of 10 000.

rithms. However, they do not seem to have such differences as we saw in the previous section. On the other hand, we can observe that IRACE+DE and IRACE+SBX have some runs in which they got very robust solutions (small boxplots) and with lower fitness than the rest of the algorithms. These are two very desirable characteristics when working with real-world problems. It is not only interesting to have good solutions (small fitness values) for a specific scenario, but maintaining this quality in different scenarios or situations is also relevant.

On the other hand, if we perform this same study but in the case of 10 000 simulations budget, as we can see in Figure 7, the results seem to change quite a lot. While IRACE+DE still seems to have some very promising results, for some independent runs, the solutions found are worse (both in quality and robustness) than even some of those found by the Gray code algorithms. In general, IRACE seems to be more consistent in quality and robustness than the other algorithms. IRACE+DE has a fairly high variability between its runs, while IRACE+GA and IRACE+SBX have a more even mix of better and worse quality iterations between them.

After analysing the results graphically, we will compare the results numerically.

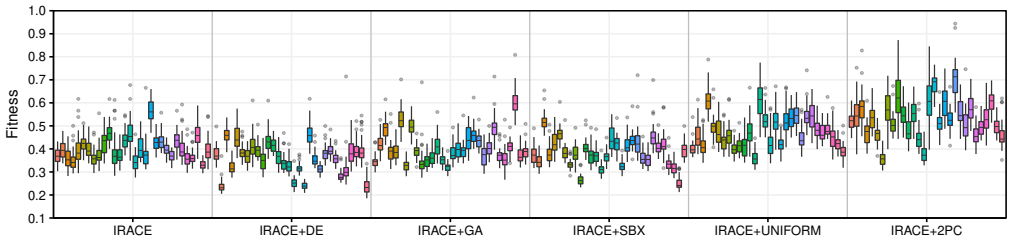

Figure 6: Fitness of the solutions obtained by the algorithms with a budget of 1 000 simulations. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the test set.

Table 3: Wilcoxon Test p-value of the testing set with Holm correction.

Budget of 1 000 simulations					
	IRACE	IRACE+DE	IRACE+GA	IRACE+SBX	IRACE+UNIFORM
IRACE+DE	$< 2e{-16}$	—	—	—	—
IRACE+GA	0.48	$< 2e{-16}$	—	—	—
IRACE+SBX	$4.5e{-7}$	$2.7e{-11}$	$2.3e{-8}$	—	—
IRACE+UNIFORM	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	—
IRACE+2PC	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$
Budget of 10 000 simulations					
	IRACE	IRACE+DE	IRACE+GA	IRACE+SBX	IRACE+UNIFORM
IRACE+DE	$4.5e{-11}$	—	—	—	—
IRACE+GA	$< 2e{-16}$	$< 2e{-16}$	—	—	—
IRACE+SBX	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	—	—
IRACE+UNIFORM	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	—
IRACE+2PC	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$	$< 2e{-16}$

First, we check whether there are significant differences between the different algorithms or not. To do so, we perform a non-parametric Wilcoxon rank-sum test between the algorithms to analyse whether such differences exist. In general, we always obtain p-values< 0.05, so we can conclude that there are significant differences (see Table 3). The only case where we do not have such differences is between IRACE and IRACE+GA for $1\,000$ evaluations.

Once the statistical tests have been carried out, we can focus on Table 4 where we are presented with some statistics to compare the general behaviour of the algorithms in our two case studies. On the one hand, as we have already mentioned, IRACE+DE seems to be the best option when we have a limited number of possible evaluations. We would like to highlight that for $1\,000$ evaluations, IRACE+SBX is in second place in both mean and median. The power of its crossover operator offers good results without giving up good robustness (it is the second with the lowest standard deviation). On the other hand, in the case of $10\,000$ simulations, IRACE seems to be the best option. It is the algorithm that obtains the best results, except in the case of the median, where IRACE+DE is the best. IRACE is also the best in terms of robustness. In all cases, algorithms with binary operators come last, not performing well for the TLSP.

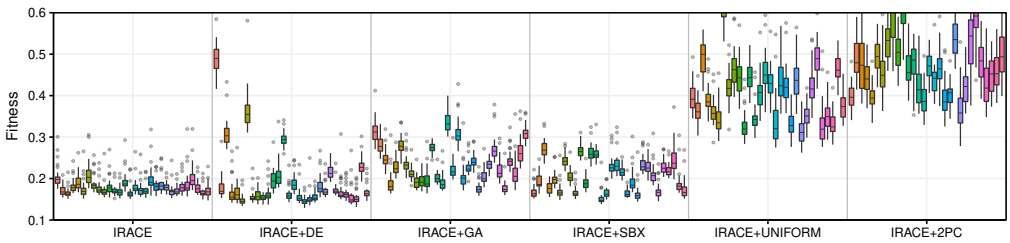

Figure 7: Fitness of the solutions obtained by the algorithms with a budget of $10\,000$ simulations. Each boxplot shows the distribution of fitness values of one solution on the 30 traffic scenarios in the test set.

Table 4: Statistics of each algorithm from the best solutions obtained in the testing phase. We mark in bold the lower value of each metric.

	1 000			10 000		
Algorithm	Mean±Conf.Int.	Median	STD Dev	Mean±Conf.Int.	Median	STD Dev
IRACE	0.403 ± **0.0044**	0.3907	**0.0669**	**0.1810** ± **0.0015**	0.1751	**0.0235**
IRACE+DE	**0.359** ± 0.0049	**0.3582**	0.0747	0.1970 ± 0.0051	**0.1657**	0.0776
IRACE+GA	0.408 ± 0.0050	0.3937	0.0763	0.2370 ± 0.0032	0.2289	0.0491
IRACE+SBX	0.384 ± 0.0046	0.3782	0.0700	0.2070 ± 0.0027	0.2042	0.0408
IRACE+UNIFORM	0.472 ± 0.0052	0.4610	0.0802	0.4060 ± 0.0053	0.3960	0.0811
IRACE+2PC	0.531 ± 0.0063	0.5231	0.0969	0.4730 ± 0.0056	0.4601	0.0855

5.3 Impact in Real World

We have previously analysed the results from an algorithmic point of view, focusing on fitness values. However, we are solving a real problem in the real world. It is relevant to compare the solutions as a domain expert would. For this reason, it is necessary to use the quality metrics that are commonly used in this type of problem. Therefore in this section, we study the main traffic and environmental indicators which give the domain's expert more information about the real quality of the solutions.

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:

1. 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,
2. we order upwards these mean fitness for each algorithm,
3. 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 testing scenarios but we do not stop the simulation this time, e.g., we allow 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 26 different traffic and environmental measures of the 30 testing scenarios.

Figure 8 (for solutions obtained in the $1\,000$ evaluations experimentation) and 9 (for the $10\,000$ evaluations experimentation) show some of the most important measures for each algorithm. An interesting outcome of this experiment is that, contrary to the previous experiments, the fitness values (without penalty) are lower in the solutions obtained in the $1\,000$ evaluations experiment. This is a very interesting result, as the number of calls to a simulator is a limiting factor when implementing these algorithms in a end-user solution. Moreover, in general, the solutions obtained at $10\,000$ obtain metrics with higher variance. In general, IRACE+2PC obtains the highest variability, except in the minimum values where IRACE+UNIFORM is the winner. These results confirm once again that these algorithms do not offer any improvement over the others. Looking at the different metrics, we see that, in the maximum values of CO, CO2, Fuel, HC, NOx, PMx, Travel times, and Waiting times; IRACE seems to have the best results. However, IRACE has lower robustness than the challenge algorithms, especially in the case of solutions obtained after $10\,000$ evaluations. On the other hand, in the mean case, IRACE+DE is the winner over the others, with lower mean and variability. As for the best minimum values, these are distributed between IRACE+SBX

(in the 1 000 evaluations experiment) and IRACE+GA (in the 10 000 evaluations experiment). Finally, we are analyzing the worst-case scenarios of the algorithms. There are several scenarios in which IRACE+DE has worse values than IRACE, even though it has better values in most of the scenarios. IRACE+2PC has the highest variability and worst results in 1 000 assessments. In 10 000 evaluations, IRACE+DE presents the best results even in the worst cases. While in minimum values of the different metrics, IRACE+GA is the best. IRACE+2PC and IRACE+UNIFORM obtain the worst results. Analyzing the worst cases is necessary when working on real-world problems. Having solutions that can cause trouble under some circumstances is not feasible when we apply these solutions in an actual city.

With all this, we can conclude that IRACE and IRACE+DE were the best algorithms according to the previous experimentation and they are still competitive in this last one. However, when we check the impact they could have in the real world, the best results are not always obtained by them. However, in average and worst cases, they are the ones that offer the best metrics. It might be interesting for a domain expert to implement solutions obtained by IRACE+SBX or IRACE+GA according to how common a particular scenario is.

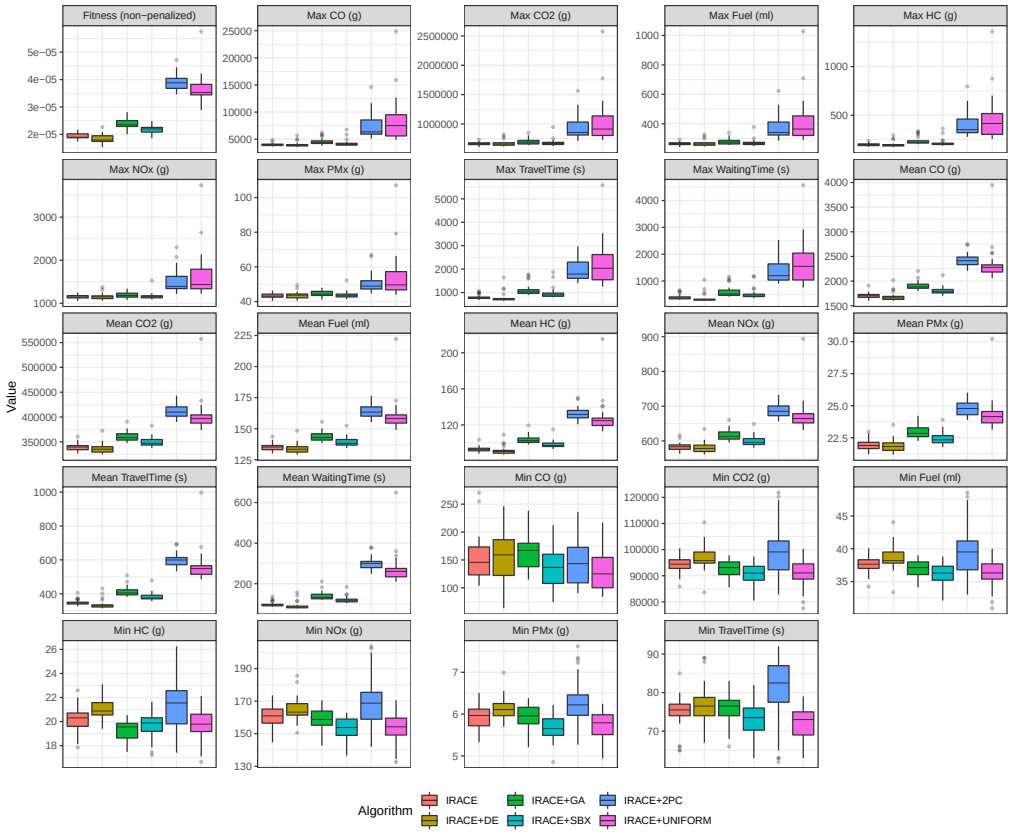

Figure 8: Traffic measures per vehicle. Mean values (and standard deviation) over 30 test traffic scenarios of the median solutions for the algorithms and 1 000 evaluations.

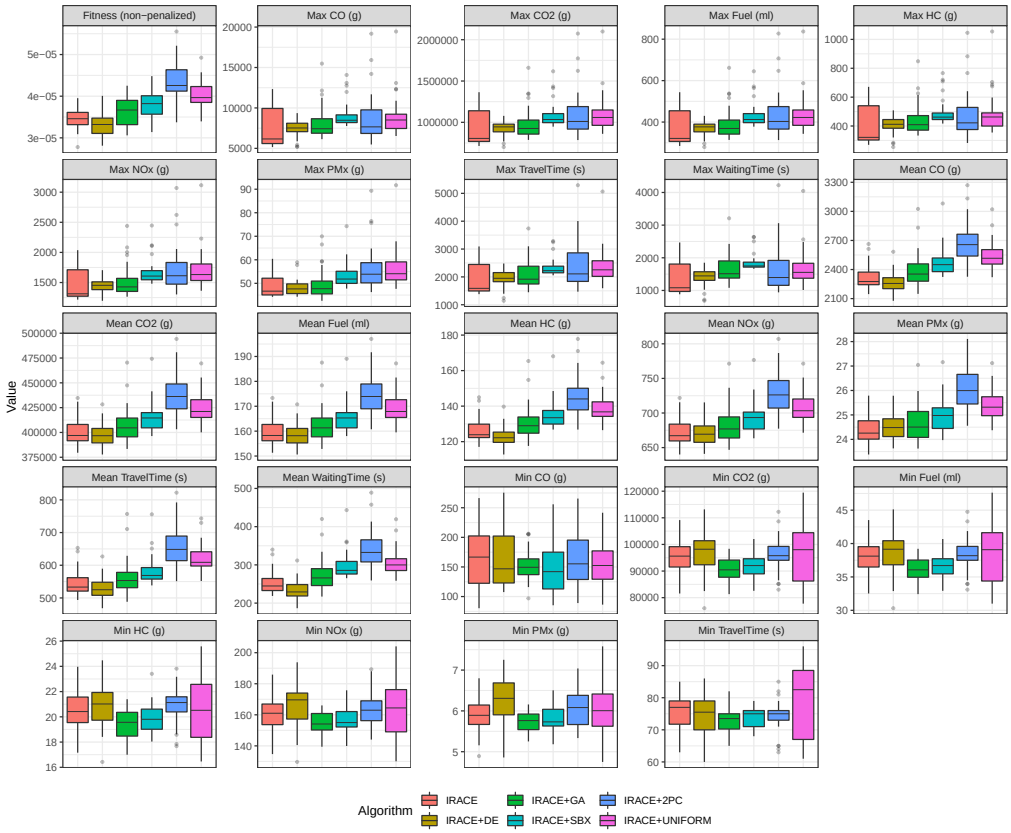

Figure 9: Traffic measures per vehicle. Mean values (and standard deviation) over 30 test traffic scenarios of the median solutions for the algorithms and 10 000 evaluations.

6 Conclusions

In this article, we have extended the idea proposed in Cintrano et al. (2021) by testing new operators used in the state-of-the-art TLSP. We have analysed the main publications about TLSP and extracted the evolutionary operators used. With them, we have created hybrids between IRACE and evolutionary algorithms: IRACE+DE, IRACE+GA, IRACE+SBX, IRACE+2PC, and IRACE+UNIFORM. We also studied the behaviour of operators that use different types of encoding for the solution: numerical and binary. After testing the algorithms in the real scenario of Malaga, Spain, and varying the maximum number of calls to the simulator, we have found that IRACE+DE returns the best results both in convergence and quality of the solution when the number of iterations is more limited. On the other hand, IRACE performed better for longer run times. However, if we study the solutions from a real-world point of view, we have found that by using other operators, such as DE or SBX, we can improve some important metrics such as pollutant gas emissions more than IRACE even as we increase the number of simulations the algorithm can perform.

As future work, we would like to evaluate other real-world problems and analyse these algorithms' performance. Besides, we want to continue adding more operators to these hybrid versions of IRACE, allowing more configuration for user needs.

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

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D. (2011). SUMO - Simulation of Urban MObility: An overview. In *SIMUL 2011, The Third International Conference on Advances in System Simulation*, pages 63–68, Barcelona, Spain. ThinkMind.

Birattari, M., Stützle, T., Paquete, L., and Varrentrapp, K. (2002). A racing algorithm for configuring metaheuristics. In Langdon, W. B. et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2002*, pages 11–18. Morgan Kaufmann Publishers, San Francisco, CA.

Blum, C. and Raidl, G. R. (2016). *Hybrid Metaheuristics—Powerful Tools for Optimization*. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer, Berlin, Germany.

Bravo, Y., Ferrer, J., Luque, G. J., and Alba, E. (2016). Smart mobility by optimizing the traffic lights: A new tool for traffic control centers. In Alba, E., Chicano, F., and Luque, G. J., editors, *Smart Cities (Smart-CT 2016)*, LNCS, pages 147–156. Springer, Cham, Switzerland.

Cintrano, C., Ferrer, J., López-Ibáñez, M., and Alba, E. (2021). Hybridization of racing methods with evolutionary operators for simulation optimization of traffic lights programs. In Zarges, C. and Verel, S., editors, *Proceedings of EvoCOP 2021 – 21th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 12692 of *LNCS*, pages 17–33. Springer, Cham, Switzerland.

Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search spaces. *Complex Systems*, 9(2):115–148.

Deb, K. and Agrawal, S. (1999). A niched-penalty approach for constraint handling in genetic algorithms. In Dobnikar, A., Steele, N. C., Pearson, D. W., and Albrecht, R. F., editors, *Artificial Neural Nets and Genetic Algorithms (ICANNGA-99)*, pages 235–243. Springer Verlag.

Ferrer, J., García-Nieto, J., Alba, E., and Chicano, F. (2016). Intelligent testing of traffic light programs: Validation in smart mobility scenarios. *Mathematical Problems in Engineering*, 2016:1–19.

Ferrer, J., López-Ibáñez, M., and Alba, E. (2019). Reliable simulation-optimization of traffic lights in a real-world city. *Applied Soft Computing*, 78:697–711.

García-Nieto, J., Alba, E., and Olivera, A. C. (2012). Swarm intelligence for traffic light scheduling: Application to real urban areas. *Engineering Applications of Artificial Intelligence*, 25(2):274–283.

García-Nieto, J., Olivera, A. C., and Alba, E. (2013). Optimal cycle program of traffic lights with particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 17(6):823–839.

Garnier, J., Kallel, L., and Schoenauer, M. (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7(2):173–203.

Heidrich-Meisner, V. and Igel, C. (2009). Hoeffding and Bernstein races for selecting policies in evolutionary direct policy search. In Danyluk, A. P., Bottou, L., and Littman, M. L., editors, *Proceedings of the 26th International Conference on Machine Learning, ICML 2009*, pages 401–408, New York, NY. ACM Press.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Krajzewicz, D., Erdmann, J., Behrisch, M., and Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3-4):128–138.

Krejca, M. S. (2019). *Theoretical analyses of univariate estimation-of-distribution algorithms*. doctoralthesis, Universität Potsdam.

Little, J. D. (1966). The synchronization of traffic signals by mixed-integer linear programming. *Operations Research*, 14(4):568–594.

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Stützle, T., and Birattari, M. (2016). The irace package: Iterated racing for automatic algorithm configuration. *Operations Research Perspectives*, 3:43–58.

Mahto, T. and Malik, H. (2021). Traffic signal control to optimize run time for energy saving: a smart city paradigm. In *Metaheuristic and Evolutionary Computation: Algorithms and Applications*, pages 491–497. Springer.

Péres, M., Ruiz, G., Nesmachnow, S., and Olivera, A. C. (2018). Multiobjective evolutionary optimization of traffic flow and pollution in Montevideo, Uruguay. *Applied Soft Computing*, 70:472–485.

Price, K., Storn, R. M., and Lampinen, J. A. (2005). *Differential Evolution: A Practical Approach to Global Optimization*. Springer, New York, NY.

Pushak, Y. and Hoos, H. H. (2018). Algorithm configuration landscapes: More benign than expected? In Auger, A., Fonseca, C. M., Lourenço, N., Machado, P., Paquete, L., and Whitley, D., editors, *Parallel Problem Solving from Nature - PPSN XV*, volume 11101 of *LNCS*, pages 271–283. Springer, Cham, Switzerland.

Putha, R., Quadrifoglio, L., and Zechman, E. (2012). Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions. *Computer-Aided Civil and Infrastructure Engineering*, 27(1):14–28.

Sanchez, J., Galan, M., and Rubio, E. (2005). Bit level versus gene level crossover in a traffic modeling environment. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, pages 1190–1195.

Sánchez, J., Galán, M., and Rubio, E. (2008). 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.

Sánchez-Medina, J. J., Galán-Moreno, M. J., and Rubio-Royo, E. (2010). 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.

Spears, V. M. and De Jong, K. A. (1991). On the virtues of parameterized uniform crossover. In Belew, R. K. and Booker, L. B., editors, *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 230–236. Morgan Kaufmann Publishers, San Mateo, CA.

Stolfi, D. H. and Alba, E. (2014). Red swarm: Reducing travel times in smart cities by using bio-inspired algorithms. *Applied Soft Computing*, 24:181–195.

Stolfi, D. H. and Alba, E. (2015). An evolutionary algorithm to generate real urban traffic flows. In Puerta, J. M., Gámez, J. A., Dorronsoro, B., Barrenechea, E., Troncoso, A., Baruque, B., and Galar, M., editors, *Advances in Artificial Intelligence, CAEPIA 2015*, volume 9422 of *LNCS*, pages 332–343. Springer, Heidelberg.

Syswerda, G. (1989). Uniform crossover in genetic algorithms. In Schaffer, J. D., editor, *Proc. of the Third Int. Conf. on Genetic Algorithms*, pages 2–9. Morgan Kaufmann Publishers, San Mateo, CA.

Teklu, F., Sumalee, A., and Watling, D. (2007). A genetic algorithm approach for optimizing traffic control signals considering routing. *Computer-Aided Civil and Infrastructure Engineering*, 22(1):31–43.

Teo, K. T. K., Kow, W. Y., and Chin, Y. K. (2010). 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*, pages 172–177. IEEE, IEEE Press.

Tsai, C.-W., Teng, T.-C., Liao, J.-T., and Chiang, M.-C. (2021). An effective hybrid-heuristic algorithm for urban traffic light scheduling. *Neural Computing and Applications*, 33(24):17535–17549.

Wei, H., Zheng, G., Gayah, V., and Li, Z. (2019). A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*.