Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 3871046, 19 pages
http://dx.doi.org/10.1155/2016/3871046

Research Article

Intelligent Testing of Traffic Light Programs: Validation in Smart Mobility Scenarios

Javier Ferrer, José García-Nieto, Enrique Alba, and Francisco Chicano

Departamento de Lenguajes y Ciencias de la Computación, Universidad de Málaga, E.T.S. Ingenieria Informatica, Bulevar Louis Pasteur 35, 29071 Málaga, Spain

Correspondence should be addressed to Javier Ferrer; ferrer@lcc.uma.es

Received 1 July 2015; Revised 24 December 2015; Accepted 12 January 2016

Academic Editor: Dan Simon

In smart cities, the use of intelligent automatic techniques to find efficient cycle programs of traffic lights is becoming an innovative front for traffic flow management. However, this automatic programming of traffic lights requires a validation process of the generated solutions, since they can affect the mobility (and security) of millions of citizens. In this paper, we propose a validation strategy based on genetic algorithms and feature models for the automatic generation of different traffic scenarios checking the robustness of traffic light cycle programs. We have concentrated on an extensive urban area in the city of Malaga (in Spain), in which we validate a set of candidate cycle programs generated by means of four optimization algorithms: Particle Swarm Optimization for Traffic Lights, Differential Evolution for Traffic Lights, random search, and Sumo Cycle Program Generator. We can test the cycles of traffic lights considering the different states of the city, weather, congestion, driver expertise, vehicle's features, and so forth, but prioritizing the most relevant scenarios among a large and varied set of them. The improvement achieved in solution quality is remarkable, especially for CO_2 emissions, in which we have obtained a reduction of 126.99% compared with the experts' solutions.

1. Introduction

Nowadays, all initiatives for the development of the *Smart City* [1–3] focus on public institutions for the impact they have on society in general. A Smart City is a holistic initiative to manage the city where actions and services, based on information technology, are optimally conceived and deployed for sustainable living. Traffic flow management is one of the most important aspects in the context of smart cities due to the large number of vehicles to be managed in the metropolitan area. An optimal management of traffic might be beneficial to minimize journey times and reduce fuel consumption and harmful emissions. For this purpose, the cycle programming of traffic lights constitutes a key task [4]. Nevertheless, the large number of cycle combinations that appear in current traffic light schedules require automatic intelligent tools to be used by the experts in this field.

In this regard, several research studies on automatic traffic light scheduling exist which use different intelligent techniques such as fuzzy logic controllers [5, 6], multiagent systems [7], neural networks [8], and metaheuristic algorithms [9–11]. However, these solutions entail a high dependency on the scenario instances under examination, with specific conditions and limited variability. Furthermore, the variability of the traffic system is due, among other factors of varying importance, to different weather conditions, the daily traffic versus the weekend's, rush hours, changing environment, and so forth. Therefore, our main objective in this paper is to offer a validation strategy for the generated cycle programs of different and variable traffic scenarios, since they could determine the robustness of the proposed cycle programs of traffic lights for the experts.

With the aim of representing high variability systems (like vehicular traffic scenarios), feature models (FMs) [12] have emerged in the last decade as a standard strategy for modeling common and variable features of a system and their relationships. FMs have been applied to test Software Product Lines (SPL) [13–15] with great success, although they have still not been considered for representing features in other domains. Mindful of this, our motivation is therefore to develop a model for traffic management, with the target being to represent all feature variability for the generation

of different scenarios of urban mobility. In this regard, we aim to extract validated information about which traffic lights program (from among a number of them) is best adapted for most traffic scenarios with different feature combinations and which fails to manage the traffic flow with other specific feature combinations.

In fact, among the main features that affect traffic flow, we can enumerate the following: different meteorology conditions, type of vehicle, driver expertise, simulation time, and so forth. However, the analysis of all possible traffic scenarios is inviable due to the large number of feature combinations to take into account. For this reason, we propose an evolutionary approach, called the *Prioritized Genetic Feature Model* (PGFM) algorithm, designed for the generation of a minimal prioritized test suite of traffic scenarios, which meet a coverage criterion (pairwise). The pairwise coverage criterion [16] is the most popular in the literature. Since all scenarios are not equally important, our feature model has been extended to provide priorities for the features. Therefore, in the case there are time or cost requirements, this prioritization allows us to first generate the most important scenarios, in which we test the traffic lights programs.

In this paper, we focus on an extensive urban area of Malaga city (in Spain) for the case study. We use the well-known SUMO (Simulator of Urban Mobility) [17] traffic simulator, which offers a continuous source of information about the vehicle's flow: velocity, fuel consumption, emissions, journey time, and so forth, giving rise to configurations of realistic scenarios according to real patterns of mobility. In particular, we validate a large number of traffic light programs previously generated by four optimization algorithms: two metaheuristic techniques, PSOTL [9] and DETL [9]; a blind stochastic search algorithm, random search (RS); and a deterministic procedure following human experts' rules for traffic configuration, SCPG [17]. Our present work will help determine the suitability of the computed configurations for a wider set of city scenarios for which they have not been trained. The main contributions of this paper are summarized as follows:

(i) Design and application of a prioritized feature model (PFM) for the representation of traffic light management systems: as far as we know, this is the first application of a PFM to traffic light management systems. It constitutes the actual use of cross-fertilizing techniques, since we take advantage of prioritized feature models, typically applied to software testing, to leverage traffic and environmental solutions in current smart cities (like Malaga). This model represents traffic variability through different features weighted according to their probability of occurrence and importance. This testing technique is useful here to identify which cycle programs fail to manage the traffic flow in different scenarios.

(ii) Development of Prioritized Genetic algorithm for feature models (PGFM), for the automatic generation of weighted traffic scenarios: PGFM is shown here to comply with the pairwise covering criterion [16]. In this way, we are able to work with a reduced set of scenarios that cover all the different traffic features considered in our PFM.

(iii) Thorough analysis and comparison of the different traffic light programs for the Malaga city case study: these cycle programs were generated as candidate solutions by means of four optimization algorithms: PSOTL, DETL, RS, and SCPG (human experts' solutions). Although these solutions were obtained for a specific traffic scenario, they have been validated for multiple scenarios covering our PFM.

The remainder of the paper is organized as follows. Section 2 presents an overview of related work in the literature. In Section 3, basic concepts are explained for the sake of a better understanding of the work presented here. After this, Section 4 details the validation strategy. Section 5 describes how traffic scenario features are represented by means of our feature model. Section 6 outlines the PGFM algorithm and presents the generated test suite of traffic scenarios. Then, Section 7 is devoted to presenting the Malaga city case study. In Section 8, the experimental procedure and the analysis of results are addressed. In Section 9, we provide a deeper analysis, focusing on the prioritization and the ranking of the solutions. Section 10 discuses the threats to validity. Finally, Section 11 outlines some concluding remarks and future work.

2. Related Work

This section presents an overview of related work considering a two-pronged approach: combinatorial testing for feature models and optimization of traffic light cycle programs. The former focuses on intelligent techniques for combinatorial interaction testing (CIT), using covering arrays. The latter considers the use of metaheuristics for traffic optimization.

CIT [16] is an effective testing approach for detecting failures caused by certain combinations of components or input values. Generally, this task consists of generating, at least, all possible combinations of the parameters' values (this task is NP-hard [18]). A large number of CIT approaches have already been published. A good overview and classification of approaches can be found in [19, 20] and more recently in [21]. In addition, an extensive survey that focuses on CIT with constraints is also given in [22]. There are only two approaches (to the best of our knowledge) that support prioritized test data generation: the Deterministic Density Algorithm (DDA) presented in [23] and an approach based on Binary Decision Diagrams (BDD) [24]. However, neither of them have been applied to feature models prioritization. So, here we use the priorities of the traffic features to generate prioritized scenarios for testing traffic light programs.

Except for a few efforts, the application of bioinspired techniques to combinatorial interaction testing with feature models remains largely unexplored. Garvin et al. applied simulated annealing to CIT for computing n-wise coverage for feature models [25]. Ensan et al. also propose a genetic algorithm approach for test case generation [26]. In contrast with the proposed work here, they use as fitness function a variation of cyclomatic complexity metric adapted to feature

models; their goal is not n-wise coverage and they do not consider priorities. Henard et al. propose an approach that uses a dissimilarity metric that favors individuals whose n-wise coverage varies the most from the current population thus increasing the chances of wider coverage [27]. A key difference with this work is that the prioritization of them is not based on assigned weights that have a real value. Recent surveys on feature models [28, 29], attest to the increasing relevance of the topic within the community but also confirm that the latent potential of search based testing techniques remains largely untapped.

The generation of test suites from feature models has recently been examined in several studies. Oster et al. proposed MoSo-PoLiTe [30], an approach that translates feature models and their constraints into binary constraint solver problems (CSP), from which they compute pairwise covering arrays. Hervieu et al. developed a tool called PACOGEN that also relies on constraint programming for computing pairwise coverage from feature models [31]. Johansen et al. [15] proposed a greedy approach to generate n-wise test suites that adapts Chvátal's with the aim of solving the set cover problem. In the work presented here, a test suite is generated, although it is composed of the different traffic scenarios where traffic light programs are tested.

In terms of traffic optimization, metaheuristic algorithms [32] have become very popular for solving traffic light staging problems. A first attempt was proposed by Rouphail et al. [33], where a genetic algorithm (GA) was coupled with the CORSIM [34] microsimulator for the timing optimization of nine intersections in the city of Chicago (USA). Following the model proposed in Brockfeld et al. [35], Sánchez et al. [10] designed a GA with the objective of optimizing the cycle programming of traffic lights in a commercial area in the city of Santa Cruz de Tenerife (Spain). In that approach, the computation of valid states was done before the algorithm began, and it highly depended on the scenario instance tackled. A GA was also used by Turky et al. [36] to improve the performance of traffic lights and pedestrian crossing control in a single four-way, two-lane intersection.

Peng et al. [37] presented a particle swarm optimization (PSO) with isolation niches for the scheduling of traffic lights. In that approach, a purely academic instance with a restrictive one-way road with two intersections was used to test the proposal. Kachroudi and Bhouri [38] applied a multiobjective version of PSO for optimizing cycle programs using a predictive control model based on a public transport progression model. In that approach, private and public vehicle models were used to carry out simulations on a virtual urban road network made up of 16 intersections and 51 links. Kesur [39] used the nondominated sorting genetic algorithm (NSGA-II) to evaluate two confront objectives: delay and number of stops. This work focuses on evaluating different modifications of the NSGA-II algorithm. In addition, the traffic model used was the microscopic stochastic traffic network simulation proposed by him. García-Nieto et al. [9, 40] proposed a PSO approach with the SUMO microsimulator for traffic light programming in large realistic urban areas, like Malaga and Seville (Spain) and Bahía Blanca (Argentina). For all the scenarios, this approach considered hundreds of traffic lights and circulating vehicles. In addition, these last works [9, 40] evaluated other tools like SCOOT that uses different traffic light schemes of scheduling on different instances adapted to very specific use cases. Performing comparative studies with these tools is out of the scope of the proposed work here, where the main goal is the validation of traffic light plans in certain urban area under different environmental conditions.

In summary, the aforementioned approaches have focused on different aspects of traffic light programming. However, a common limitation in all of them is that, in the optimization phase, they considered just one scenario with specific traffic conditions. This means that the resulting cycle programs could be strongly biased to the optimized scenario instances, so even a slight change in the traffic conditions (e.g., weather change) might suppose that the cycle program does not work properly.

In contrast, our initiative stands for cross-fertilization of the two domains involved here: Feature Modeling and Traffic Lights Optimization. Therefore, we consider a number of different traffic conditions in scenarios, covering all possible features according to our designed feature model, so, from now on by means of our approach, the cycle programs will be empirically validated, thereby providing the experts in traffic management with robust and generalist solutions.

3. Background

In this work, we do not focus on the simulation model; hence a comparison of simulation algorithms is out of the scope of this study, although we do use one simulator (SUMO simulator) to measure the quality of the traffic light programs. We focus on the optimization and validation models. In this regard, the validation model has been specified as general as possible in order to be used with multiple general purpose optimization algorithms. With this aim, we have tested two metaheuristic algorithms: PSO and DE, a stochastic random search algorithm, and a deterministic traffic scheduler, SCPG.

Before describing our validation strategy, several required concepts are introduced: the SUMO traffic simulator, which is the simulator used to measure the quality of the solution proposed, the cycle program optimization of traffic lights program, the optimization algorithms used to generate traffic lights programs, and the feature models proposed here.

3.1. SUMO Traffic Simulator. SUMO (Simulator of Urban Mobility) [17] is a well-known traffic simulator that provides an open source, highly portable, and microscopic road traffic simulation tool designed to handle large road scenarios. SUMO requires several input files that contain information about the traffic and the streets to be simulated. A *journey* is a vehicle movement from one location to another defined by the starting edge (street), the destination edge, and the departure time. A route is an extended journey, meaning that a route definition contains not only the first and last edges but also all the edges the vehicle will pass through. Additional files can be added to SUMO containing the map or traffic light positions and cycles.

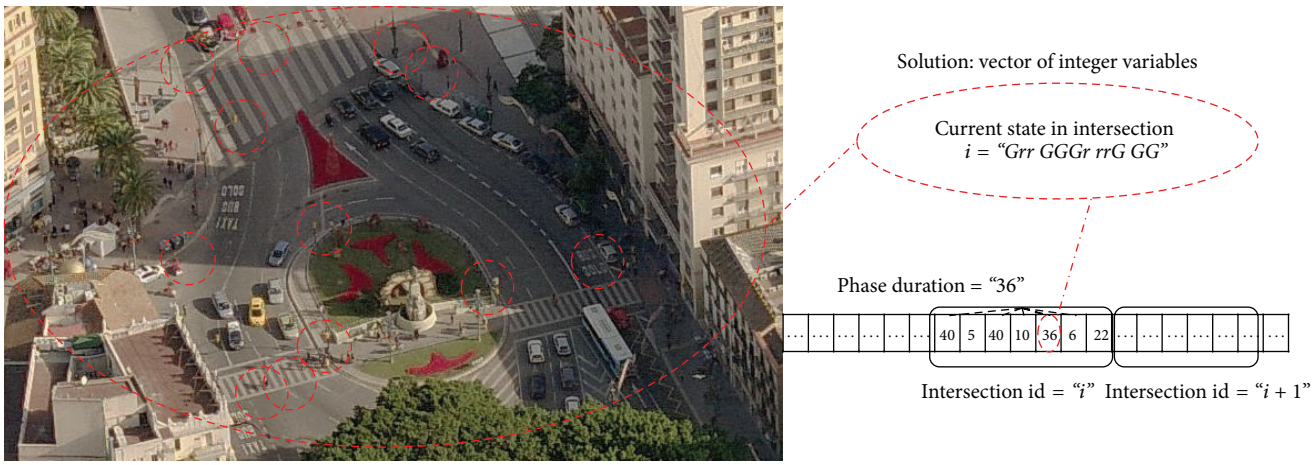

FIGURE 1: Cycle program (phase duration) of traffic lights within intersections. Dashed circles indicate where traffic lights are located in the real intersection.

It is important to note that SUMO by default provides the valid combination of states that the traffic light controller can go through inside the map specification file, and an approximation of interval times for these states [41]. This means that SUMO already incorporates a solver (SCPG) for the cycle program of traffic lights based on greedy and human knowledge.

The output of a SUMO simulation is registered in a journey information file that contains information about each vehicle's departure time, the time the vehicle waited to set off (offset), the time the vehicle arrived, the duration of its journey, and the number of steps in which the vehicle speed was below 0.1 m/s (temporal stops in driving). Other output files gather information about emission traces in vehicles (CO_2, NO_x, PM, etc.) and hydrocarbon consumption. This information is used to evaluate the quality of alternative traffic light cycle programs.

3.2. Cycle Program Optimization of Traffic Lights. The objective in this problem is to find optimized cycle (timing) programs for all the traffic lights located in a given urban area with the aim of reducing the global journey time, emissions, and fuel consumption. Specifically, cycle programs are the time span that a set of traffic lights (at an intersection) keep their color states. This is the first step of our work: obtaining configurations of the city traffic lights optimizing these metrics; the second is to find the best of them in terms of generalization for different city conditions (weather, traffic, driver, and vehicle conditions).

An example of this mechanism can be observed in Figure 1, where the intersection with `id` = "i" contains seven phases with durations of 40, 5, 40, 10, 36, 6, and 22 seconds. In these phases, the states have twelve signals (colors), each one of them corresponding to one of the twelve signal lights located in the intersection being studied (dashed circles indicate where traffic lights are located in the real intersection). These states are the valid ones generated by SUMO [42] obeying real traffic rules. In this instance, the fifth phase contains the state *"Grr GGGr rrG GG"* meaning that seven traffic lights are green (G) and the other five are red (r) for 36 seconds. The following phase changes the state of the traffic lights to another valid combination, for example, *"yGG yyyG GGy yy"* (y means yellow) for 6 seconds. The last phase is followed by the first one, and this cycle is repeated for the entire analysis. All the intersections in the complete scenario perform their own cycles of phases at the same time, thereby comprising the global schedule of signal lights. As mentioned before, computing the Signal Light Timing Program (SLTP) is based on optimizing the combination of phase durations of all traffic lights (in all intersections) with the intention of improving the global flow of vehicles.

A formal definition of the optimal SLTP is as follows.

Let $P = \{I_1, \ldots, I_n\}$ be a set of intersections, where each one has a different set of phases $I_i = \{\varphi_1, \ldots, \varphi_m\}$ and each φ_j represents the timespan that the set of traffic lights in intersection I_i keep one valid state of light colors (e.g., "*Grr GGGr rrG GG*"); find a program P' that minimizes a scoring function $\Theta : \Gamma \rightarrow p$ such that

$$P' = \underset{p \subset \Gamma}{\operatorname{argmin}} \{\Theta(p)\}, \tag{1}$$

where Γ is the space of all possible combinations of cycle programs and p a given program of Γ.

Since the timespans of phase durations are calculated in seconds (as done in real traffic lights), Γ can be represented with a tuple of positive integer numbers $\mathbb{Z}^+$. Then, the number of possible program combinations, that is, the solution space size, can be calculated as $T^{\sum_{i=1}^{n} |I_i|}$, T being the value in the interval of possible timespans. This way, as intervals were set to $T = [5, 60]$ and the worked instances had 304 phases (at least), the problem solution space would consist of $55^{304} = 1.18 \cdot 10^{529}$ combinations. Therefore, efficient automated approaches are required to tackle it. The number of phases (304) is automatically computed according to the traffic model by the SUMO simulator when a problem instance is generated.

Input: A scenario instance $\phi(c)$ of a given city c
Output: Best found solution b encoding a cycle program
(1) $S \leftarrow$ **initializeSwarm**()
(2) **while** $g <$ MAXIMUM$_g$ **do**
(3) **for** each particle x_g^i in S **do**
(4) $x_{g+1}^i \leftarrow$ **update**$(x_g^i, v_g^i, p_g^i, b_g)$ // update velocity and particle's position as in Standard PSO 2011
(5) $q_{g+1}^i \leftarrow$ **Q**(x_{g+1}^i) // Mid-Thread quantisation: adaptation to cycle programs encoding
(6) **evaluate**$(q_{g+1}^i, \phi(c))$ // solution evaluation by micro-simulation on scenario instance $\phi(c)$
(7) **end for**
(8) $b_{g+1} \leftarrow$ **updateLeader**(b_g, q_{g+1}^i) // if better
(9) **end while**

Algorithm 1: Pseudocode of PSOTL.

3.3. Optimization Algorithms for Cycle Programming. In this subsection we briefly describe the algorithms proposed for the optimization of cycle programs of traffic lights.

3.3.1. PSOTL. This is a particle swarm optimization algorithm for finding quasi-optimal cycle programs for traffic lights. In PSOTL, the initial swarm is composed of a number of particles (solutions) initialized with a set of random values representing the phase durations. These values are within the time interval $[5, 60] \subseteq \mathbb{Z}^+$ and constitute the range of possible time spans (in seconds) a traffic light can be kept on a signal color (only green or red, the time for yellow is a constant value). This interval is specified to follow several examples of real traffic light programs provided by Malaga's City Council (Spain).

Since the optimal SLTP requires solutions encoded with a vector of integers (representing phase durations in seconds), we have used the *quantisation* method provided in the standard specification of PSO 2011 [43]. This quantisation is applied to each new generated particle and transforms the continuous values of particles to discrete ones. It consists of a Mid-Thread uniform quantiser as specified in (2). The quantum step is set here to $\Delta = 0.5$. Consider

$$Q(x) = \Delta \cdot \left\lfloor \frac{x}{\Delta} + 0.5 \right\rfloor. \quad (2)$$

Algorithm 1 describes the pseudocode of PSOTL. The algorithm starts by initializing the swarm (Line (1)), which includes both positions and velocities of the particles. The corresponding personal best p^i of each particle is randomly initialized, and the leader b is computed as the best particle of the swarm. Then, for a maximum number of iterations, each particle is updated (Line (4)), quantised (Line (5)), and evaluated (Line (6)), according to a scenario instance ϕ(city). At the end of each iteration, leader b is also updated (Line (8)). Finally, the best solution (cycle program in individual b) found so far is returned.

3.3.2. DETL. This algorithm also performs a population-based search, but following in this case the operation scheme as specified by the Differential Evolution algorithm (version DE/rand/1) [44].

As shown in Algorithm 2, after initializing the population in (Line (1)), the individuals evolve for a number of iterations performing differential operators (Line (4)). After this, solutions are quantised (Line (5)) and evaluated (Line (6)), according to a scenario instance $\phi(c)$. At the end of each iteration, each particle is either selected or not (Line (7)) depending on whether it outperforms the previous one in the evolution procedure. Finally, the best solution (cycle program in particle v) found so far is returned.

3.3.3. RS. Random search is included here not as a serious competitor but as a sanity check to find out whether an intelligent algorithm is actually needed. In RS, at each iteration step, a new solution vector of integer variables is randomly generated (uniformly) in the range of $[5, 60]$. The new individual replaces the previous one if it is better.

3.3.4. SCPG. This is a deterministic algorithm provided by the SUMO [42] package for generating cycle programs. This technique incorporates actual scheduling information used by human experts in the domain. Cycle programs generated by SCPG are actually running our city traffic light systems at present. This algorithm consists of assigning fresh values in the range $[6, 31]$ to the phase durations according to three different factors:

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

3.4. Feature Models. Feature models (FMs) provide a way to make a compact representation for modeling the common and variable features of a system, their relationships, and the constraints between them. FM can be understood as a tree of concepts, where the nodes are the features (which are depicted as labeled boxes), and the edges represent the relationships between them. Thus, an FM denotes the set of feature combinations. In an FM, each feature (except the root) has one parent feature and can have a set of child features. Note here that a child feature can only be included in a feature combination of a valid product if its parent is included as

Input: A scenario instance $\phi(c)$ of a given city c
Output: Best found solution v encoding a cycle program
(1) $P \leftarrow$ initializePopulation()
(2) **while** $g <$ MAXIMUM$_g$ **do**
(3) **for** each individual v_g^i **do**
(4) $w_{g+1}^i \leftarrow$ **differentialOps**(vg, F) // differential mutation and crossover as in DE/rand/1
(5) $q_{g+1}^i \leftarrow \mathbf{Q}(w_{g+1}^i)$ // Mid-Thread quantisation: adaptation to cycle programs encoding
(6) **evaluate**$(q_{q+1}^i, \phi(c))$ // solution evaluation by micro-simulation on scenario instance $\phi(c)$
(7) $v_{g+1}^i \leftarrow$ **selection**(v_g^i, h_{g+1}^i) // differential selection of new vector h_{g+1}^i if better
(8) **end for**
(9) **end while**

Algorithm 2: Pseudocode of DETL.

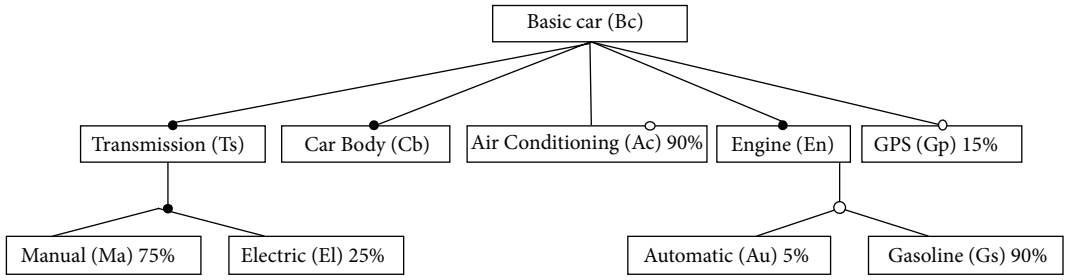

Figure 2: Basic car feature model.

well. The root feature is always included. In order to illustrate these concepts, in Figure 2 we show an instance from the SPLOT repository [45] of a basic car. This instance defines 24 different configurations combinations and four kinds of feature relationships:

(i) *Mandatory features* are depicted as filled circles. A mandatory feature is selected whenever its respective parent feature is selected. Features called `Transmission (Ts)`, `Car Body (Cb)`, and `Engine (Eg)` are mandatory.

(ii) *Optional features* are depicted with an empty circle. An optional feature may or may not be selected if its respective parent feature is selected. Features `Air Conditioning (Ac)` and `GPS (Gp)` are optional.

(iii) *Exclusive-or relationships* are depicted as empty circles crossed by a set of lines connecting a parent feature with its child features. They indicate that exactly one of the features in the exclusive-or group must be selected whenever the parent feature is selected. If feature `Transmission (Ts)` is selected, then either feature `Manual (Ma)` or feature `Automatic (Au)` must be selected.

(iv) *Inclusive-or relationships* are depicted as filled circles crossed by a set of lines connecting a parent feature with its child features. They indicate that at least one of the features in the inclusive-or group must be selected if the parent is selected. If feature `Engine` is selected then at least one of the features `Electric (El)` or `Gasoline (Ga)` must be selected.

Besides the parent-child relationships, features can also relate across different branches of the FM with the so-called *Cross-Tree Constraints* (CTC). These constraints, as well as those implied by the hierarchical relationships between features, are usually expressed and checked using propositional logic [46]. These FMs could be extended to take into account feature priority. In our example, we indicate the weight associated with the optional features as a percentage; the mandatory features should always be present (do not need to be weighted). In the following paragraphs, we summarize the basic terminology that we use in this paper related to feature models.

A configuration is a pair $[\text{sel}, \overline{\text{sel}}]$ where sel and $\overline{\text{sel}}$ are the sets of selected and unselected features, respectively. In this paper, a configuration will be a traffic scenario. Regarding priorities, each feature f has a weight f_w in the range $[0, 1]$. Then, the weight $\overline{f_w}$ is $1 - f_w$. A *prioritized pair pc* is composed of two features f_1 and f_2 and a weight such that the pair weight is calculated as the product of the weight of f_1 and f_2: $pc_w = f_{1w} * f_{2w}$. Consequently, the configuration priority is calculated as the sum of the prioritized feature pairs which are present in the configuration. It should be noted that a configuration is valid in FM iff it does not contradict any implicit or explicit constraints introduced by the FM.

Combinatorial interaction testing (CIT) is a constructive approach that builds test suites in order to systematically test all configurations of a system [16]. In this paper we are going to use the pairwise coverage criterion, which is the

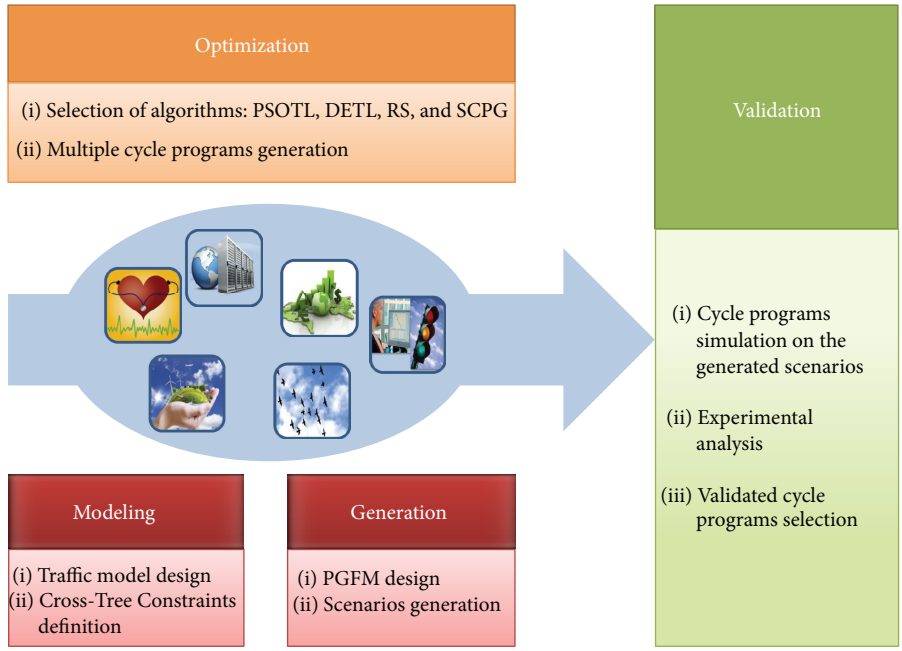

Figure 3: Modeling, optimization, generation, and validation phases of our proposed strategy.

most popular method in CIT and is based on the assumption that most errors originating in a parameter are caused by the interaction of two values [47]. This criterion is satisfied if all feature pairs ($[(f_1, f_2), (f_1, \overline{f_2}), (\overline{f_1}, f_2)$, and $(\overline{f_1}, \overline{f_2})]$) are present in at least one configuration of the test suite. When this technique is applied to FMs, the idea is to select a set of valid configurations, where the errors could be present with higher probability. In addition, the prioritization of the features makes it possible to first test the more frequent or more important features.

4. Validation Strategy

At this point, once we have explained some preliminary required concepts, we describe the main steps involved in our validation strategy for the sake of a better understanding of this study. Figure 3 illustrates the general scheme of this procedure, which consists of four main phases: cycle programs optimization, Feature Model Design, Scenarios Generation, and Cycle Programs Validation.

A brief explanation of how the four phases of our validation strategy are carried out is given in the following:

(1) *Optimization*. As described in Section 3, given an urban scenario related to a certain area in a real city, the cycle programs of traffic lights operating in this area are optimized by means of several specialized algorithms: PSOTL, DETL, RS, and SCPG, described in Section 3.3. As most of these algorithms are based on stochastic procedures, a number of different candidate solutions (representing traffic light programs) appear that must be thoroughly analyzed, with the aim of selecting the most accurate one for a wide range of traffic conditions. Note that we have used a neutral scenario for the optimization of the solutions (traffic light programs). In a later step in phase 4 (Validation), we validate these solutions with multiple scenarios with different characteristics generated in phase 3 (Generation).

(2) *Modeling*. This phase entails the feature model (Section 3.4) design for our traffic scenarios. At this stage, the human expert has to select the features and constraints and decide how the priorities are assigned to the features. More details about the traffic modeling are given in Section 5.

(3) *Generation*. The generation of scenarios is automatically carried out by means of our Prioritized Genetic algorithm for feature models (detailed in Section 6). In this way, we can obtain a test suite of scenarios that fulfill the pairwise coverage criterion. In addition, the generated scenarios are ordered by priority according to the combination of features that are involved in the specific urban area. Note that the scenarios generated are only used for the validation and not for creating better schedules by the optimizers.

(4) *Validation*. Finally, the optimized cycle programs are then validated with regard to the generated scenarios following our traffic FM. The experimental procedure leading up to accomplishing this phase is described in the experimental section (Section 8), and the resulting valid cycle programs are analyzed according to the stakeholder interests.

In addition, there exist some dependencies between these phases that have to be considered; for example, the traffic scenarios cannot be generated unless the traffic feature model has first been defined, whereas the cycle programs

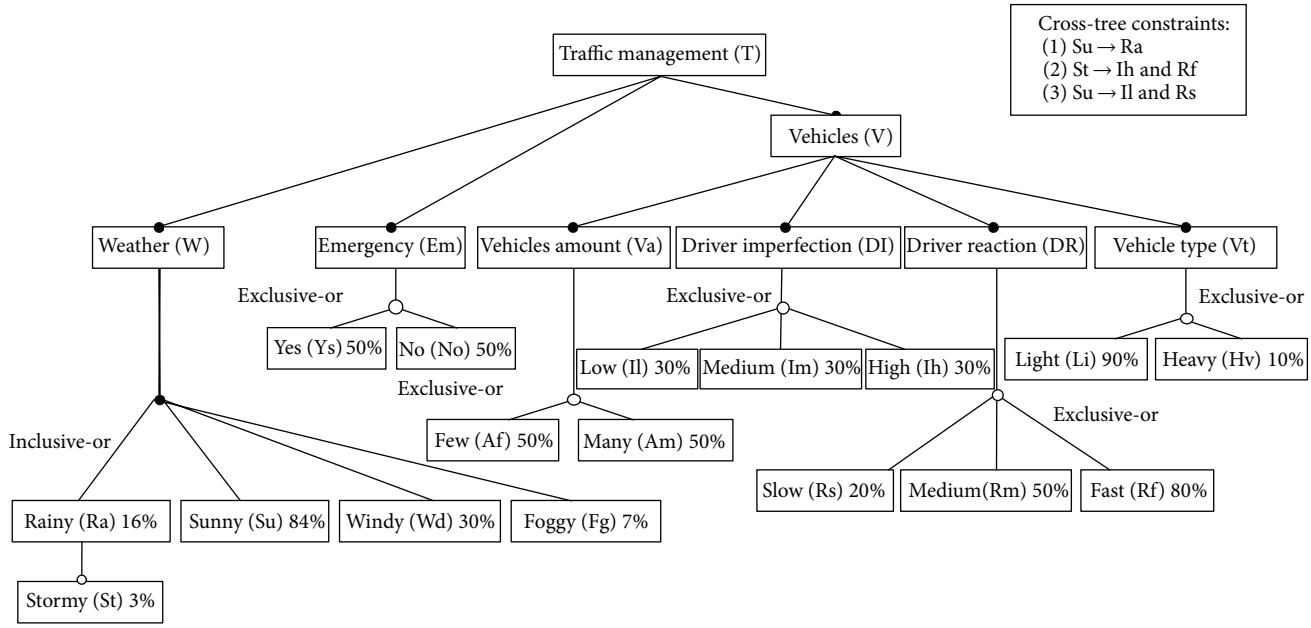

Figure 4: Traffic management feature model with priorities.

optimization phase could be done in parallel to the definition of the generation of the feature model and scenarios. The optimization phase is independent from the definition of the traffic FM because we optimise the cycle programs on a neutral scenario, which is not affected by any feature defined by the traffic FM. Finally, the cycle program validation depends on the generated traffic light programs and the traffic scenarios.

After applying these steps, we are now able to numerically evaluate a set of cycle programs of traffic lights, in order to objectively select the overall best taking into account several scenarios. With this goal, we use some traffic measures in order to evaluate a cycle program of traffic lights according to the desirable behavior of the whole traffic system. In other words, we select the best cycle program depending on the stakeholder interest such as saving fuel, reducing emissions, or avoiding traffic jams.

5. Modeling: Traffic Representation with Feature Models

Traffic management is a hard task that involves lots of varying features. Traffic is a highly variable system that could provoke a wide range of possible scenarios. Therefore, it could be represented by a feature model, which could be used for modeling the common and variable features of a system and their relationships. In addition, not every traffic scenario has the same importance or probability of occurrence. So, we have extended the FM to prioritize features in order to first test the most important scenarios.

Figure 4 shows the generated FM for representing the traffic management system. Among the main features that could describe a traffic scenario, we have taken into account several conditions affecting the traffic such as the weather, if there exists an emergency situation, and the different vehicle's characteristics. Among these vehicle features, we have the number of vehicles, type of vehicle, driver imperfection, and driver reaction time. In the FM, the features are defined as fuzzy; however, in Section 7 we set concrete values corresponding to the traffic characteristics available in the SUMO simulator. For some of these features we select the weights according to some real-world data that we describe in the following. When no data is available to justify a weight, we simply assign equal weights to all the possible values. In what follows we are going to justify the chosen features:

(i) *Weather*. The weather types we have considered are `Rainy`, `Stormy`, `Sunny`, `Windy`, and `Foggy`. They are related with a inclusive-or, so at least one feature should be selected. In order to assign the weights, we have consulted the data of the Spanish National Institute of Meteorology, concretely the data of 2012 from the Malaga airport weather station. The weight assigned is the percentage of days of the year that the associated condition held.

(ii) *Emergency*. We have considered whether an emergency situation has occurred or not because it could be important to know how traffic flow evolves in an emergency situation. Emergency values considered are `Yes` and `No`. When there is an emergency situation in the whole network, the reaction time is shorter and the traffic light programs have less time to help the driver's arrive at their destination. They are related to an exclusive-or, so only one feature can be selected. They are equally weighted.

(iii) *Vehicle*. We represent the main vehicular characteristics under this mandatory feature. For some child nodes of the `Vehicle` feature it was not possible

to obtain reliable public data, so in those cases we considered equal weight for the features. All child nodes of the `Vehicle` feature are also mandatory and are as follows:

(1) *Vehicles Amount*. The number of vehicles considered is `Few` or `Many`. They are related to an exclusive-or, so only one feature can be selected. They are equally weighted.

(2) *Driver Imperfection*. This feature represents how bad a driver is (low, medium, or high); for example, a high driver imperfection rate is a bad driver. In many traffic situations, the driver makes the difference. So, we have considered three levels of expertise. They are related to an exclusive-or, so only one feature can be selected. They are equally weighted.

(3) *Driver Reaction Time*. Time spent by the driver to carry out an action in the vehicle. We have considered three levels of reaction time (`Slow`, `Medium`, and `Fast`), in which the feature `Fast` has a larger priority than `Medium`. `Medium` has larger priority than `Slow`. They are related to an exclusive-or, so only one feature can be selected.

(4) *Vehicle Type*. We have taken into account two types of vehicles: `Light` and `Heavy`. In order to assign the weights, we have consulted public data of the traffic control center (city hall) of Malaga. Most of the vehicles are light, so we have assigned a weight of 90% to them. They are related to an exclusive-or, so only one feature can be selected.

Besides the parent-child relationships, features can also be related across different branches of the feature model with the so-called *Cross-Tree Constraints* (CTC). In this model, we have generated three CTCs that are shown in the upper right corner of Figure 4 and are as follows:

(i) $Sunny \rightarrow \overline{Rainy}$,

(ii) $Stormy \rightarrow Dri.Imperf.High \wedge Dri.ReactionFast$,

(iii) $Sunny \rightarrow Dri.Imperf.Low \wedge Dri.ReactionSlow$.

Let us explain our interpretation of the CTCs included in our traffic feature model. It is impossible for the weather to be sunny and rainy at the same time. We think this first constraint does not need any further explanation. On the one hand, when the weather is stormy, we consider that the driver's imperfection must be high and the driver's reaction must be fast because the driver is paying much more attention when the weather is bad. On the other hand, when the weather is sunny, we consider that the driver's imperfection must be low and the driver's reaction must be slow because the driver is much more relaxed.

Finally, this model represents 960 valid scenarios with different levels of importance and possibility of occurrence. The ideal situation is to generate an optimized traffic light program for each scenario, but this could be costly. Considering our previous results in program optimization for synchronizing traffic lights [9], the generation of 960 different cycle programs could take around 19 years of computation time. Moreover, here we have applied four different algorithms, so the computation time is multiplied by four, resulting in a total of 76 years of computation time. For this reason, the use of automatic intelligent algorithms to reduce the testing scenarios is mandatory for this task. Therefore, in the next section we explain how we reduce the number of testing scenarios without loosing the capacity of measuring the quality of the cycle programs.

6. Generation: PGFM Algorithm for Scenarios Generation

The Prioritized Genetic Feature Model (PGFM) algorithm is an evolutionary approach that constructs an entire test suite taking into account the feature model, the constraints between the features, and their priorities in the generation of the test suite of traffic scenarios. It is a constructive algorithm that adds one new valid scenario to the partial solution in each iteration until all pairwise combinations of features are covered. In each iteration, the algorithm tries to find the valid scenario that adds more coverage to the partial solution.

Algorithm 3 sketches the pseudocode of PGFM. As input, it receives a feature model for generating the test suite. Initially, the test suite is initialized with an empty list (Line (1)) and the set of remaining pairs (`RP`) is initialized with all the valid weighted pairwise combinations of features (Line (2)). In each iteration of the external loop (Lines (3)–(21)), the algorithm creates a random initial population of individuals (Line (5)) and enters an inner loop which applies the traditional steps of a generational evolutionary algorithm (Lines (6)–(18)). That is, some individuals (traffic scenarios in our case) are selected from the population $P(t)$, recombined, mutated, evaluated, and finally inserted in the offspring population A. An individual only contains the selected features, so the operators only affect these features. The nonselected features are those which are not contained in the individual. If a generated offspring individual is not a valid scenario (i.e., it violates a constraint derived from the feature model), it is transformed into a valid scenario by applying a `Fix` operation (Line (12)) provided by the FAMA tool [48].

The fitness value of an offspring individual (Line (13)) is the sum of the weights of the weighted pairwise combinations that would still to be covered after adding the offspring solution to the test suite. This is a minimization fitness function that promotes the generation of scenarios that cover the pairs of features with higher weights. Note that, as the search procedure advances, the cost of computing the fitness function decreases, since each time fewer weighted pairwise combinations remain uncovered.

The best individuals of $P(t)$ and A are kept for the next generation in $P(t+1)$ (Line (16)). The internal loop is executed until the maximum number of evaluations is reached. After this, the best individual found is included in the test suite (Line (19)) and `RP` is updated by removing the weighted

```
Input: Traffic Feature Model (tfm)
Output: Prioritized Test Suite
(1)  TS ← ∅ // Initialize the test suite
(2)  RP ← weighted_pairs_to_cover (tfm.features)
(3)  while not empty (RP) do
(4)   t = 0
(5)   P(t) ← Create_Population() // P = population
(6)   while evals < totalEvals do
(7)     A ← ∅ // A = auxiliary population
(8)     for i ← 1 to (populationSize/2) do
(9)       parents ← Selection(P(t))
(10)      offspring ← Recombination(parents)
(11)      offspring ← Mutation(offspring)
(12)      Fix(offspring)
(13)      Evaluate_Fitness(offspring)
(14)      Insert(offspring, A)
(15)    end for
(16)    P(t + 1) := Replace(A, P(t))
(17)    t = t + 1
(18)  end while // computing the best test case
(19)  TS ← TS ∪ bestSolution(P(t))
(20)  RemovePairs(RP, bestSolution(P(t))) // Remove the covered pairs
(21) end while // computing the best test suite
```

Algorithm 3: Pseudocode of PGFM.

pairwise combinations covered by the selected best solution (Line (20)). Then, the external loop starts again until there are no weighted pairs left in the RP set.

We set the configuration parameters of PGFM with values frequently used for genetic algorithms: one-point crossover strategy with a probability of 0.8, selection strategy binary tournament, population size of 10 individuals, mutation that iterates over all selected features of an individual and replaces a feature by another randomly chosen feature with a probability of 0.1, and termination condition of 1,000 fitness evaluations and full weight coverage in the external loop.

As a result of the execution of PGFM, Table 1 shows a list of 10 prioritized scenarios for testing. This list of scenarios (S_i with $i \in [1, 10]$) fulfills the pairwise coverage criterion, which ensures that all distinct traffic conditions are considered in our validation model. In this table, feature abbreviations in the first row are defined in Figure 4, corresponding to labels in the FM tree. A feature is included in the scenario if it is marked in the scenario's row. The last column indicates the total weight of the scenario according to the priority of the features included. In fact, we have to highlight the reduction achieved by considering only 10 scenarios from initial 960 valid scenarios, although all the possible scenarios that need to be explored are 2^{25}. This is a successful reduction, especially if we consider the time spent for an exhaustive experimentation (57 years on one single CPU) compared to the actual 8.3 days spent.

In this way, shown in Table 1, the use of priorities led us to only execute the six most important scenarios, since they guarantee 99% coverage. This indicates that several scenarios should be considered, but some of them are more important than others and so should be tested first; that is, S_1 scenario adds 63.59% coverage, then S_1 covers 63.59% of the total weight of feature pairs. The optional features that are selected in this scenario (S_1) are as follows: it is sunny, the simulation is long, there are a lot of vehicles, the driver skills are medium, the driver reaction is fast, and there is a high percentage of light vehicles. If we test under the conditions defined by S_1 we know that the results provided of our optimization algorithm for finding the traffic light cycles will cover 63.59% of city total conditions which is a lot. Doing so, we add completeness to our study and weight to its robustness in a holistic scientific analysis of the whole city.

7. Generation of Traffic Light Programs: Malaga City Case Study

As we are interested in developing an optimization solver capable of dealing with close-to-reality and generic urban areas, we have generated an instance by extracting actual information from real digital maps. From this instance, considering the scenarios computed with PGFM algorithm extracted from our FM (different traffic density, weather, etc.), the optimized cycle programs were generated. The selected urban area covers approximately 750,000 m^2 and is physically located in the city of Málaga in Spain. The information used is all real and concerns traffic rules, traffic element locations, buildings, road directions, streets, intersections, and so forth. Moreover, we have set the number of vehicles circulating, as well as their speeds, by following current specifications available from the Mobility Delegation of the Málaga's City Council. This information was collected from sensorized points in certain streets obtaining a measure of traffic density at different time intervals.

Table 1: Computed scenarios by PGFM with pairwise coverage criterion. Feature abbreviations are defined in Figure 4.

S_i	T	W	Ra	St	Su	Wd	Fg	Em	Ys	No	V	Va	Af	Am	Di	Il	Im	Ih	Dr	Rs	Rm	Rf	Vt	Li	Hv	W%
S_1	√	√			√			√		√	√	√		√	√		√		√			√	√	√		63.59
S_2	√	√			√	√		√	√		√	√	√		√	√			√		√		√	√		22.37
S_3	√	√	√				√	√	√		√	√	√		√			√	√			√	√		√	7.09
S_4	√	√	√	√		√		√		√	√	√		√	√			√	√	√			√	√		3.32
S_5	√	√	√	√			√	√	√		√	√		√	√		√		√		√		√		√	1.41
S_6	√	√			√		√	√		√	√	√	√		√	√			√	√			√		√	1.21
S_7	√	√				√		√		√	√	√		√	√			√	√		√		√		√	0.43
S_8	√	√	√			√	√	√		√	√	√		√	√	√			√			√	√	√		0.34
S_9	√	√	√	√		√		√	√		√	√	√		√		√		√	√			√		√	0.23
S_{10}	√	√	√	√		√	√	√		√	√	√		√	√			√	√			√	√		√	0.01

Table 2: Relationship between features and simulator parameters.

Features	Ra	St	Su	Wd	Fg	Ys	No	Af	Am
Parameters	$\sigma+ = 0.1$ $\tau+ = 1$	$\sigma+ = 0.1$ $\tau+ = 1$	$\sigma- = 0.1$ $\tau- = 1$	$\sigma+ = 0.1$ $\tau+ = 1$	$\sigma+ = 0.1$ $\tau+ = 1$	500 s.	1000 s.	250 u.	500 u.
Features	Il	Im	Ih	Rs	Rm	Rf	Li	Hv	—
Parameters	$\sigma- = 0.1$	–	$\sigma+ = 0.1$	$\tau+ = 1$	–	$\tau- = 1$	95% vl	85% vl	—

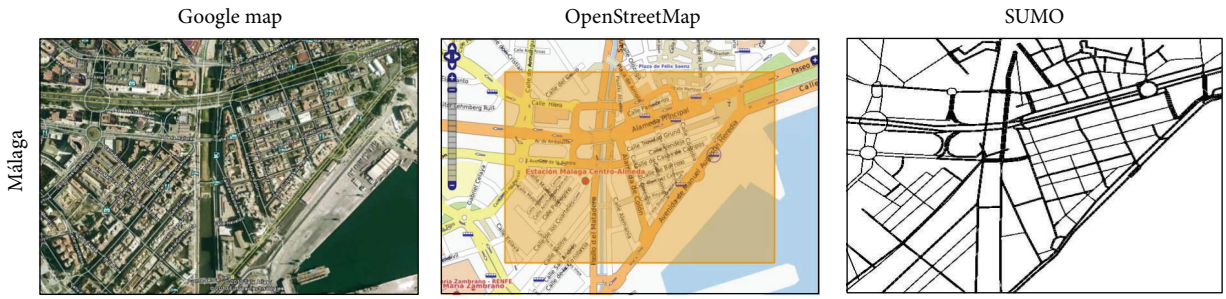

Figure 5: Malaga scenario instance. Selection from OpenStreetMaps and exportation to SUMO format.

In Figure 5, the selected area of Malaga city is shown with its corresponding captured views of OpenStreetMap and SUMO (as explained in Section 3.1). In the zone between the city center and the harbor, this instance comprises streets with different widths and lengths and several roundabouts. The main streets found in this area are Alameda Principal, Andalucía, Manuel Agustín Heredia, Colón, and Aurora. The area contains 70 intersections with 4 to 16 traffic lights at each one, adding up to a total number of 312 traffic lights. There are between 250 and 500 vehicles circulating during the analysis period; each one of the vehicles completes its own route from origin to destination circulating with a maximum speed of 50 km/h (typical in urban areas). The traffic flow is generated by means of the DUARouter tool [42] that computes vehicle routes used by SUMO using shortest path computation and dynamic user assignment (DUA) algorithms.

With regard to the driver's features, there are two main aspects to characterize the driving style: imperfection and reaction, represented in SUMO by means of parameters σ and τ, respectively. The first one is a real number in the range $[0, 1]$ for weighting the driver's skills. A high value of σ means that the driver commits many driving errors. In contrast, a low σ means good driving. The second parameter (τ) measures the driver's reaction time in seconds. In addition, we consider other parameters such as the simulation time, the number of vehicles, and the percentage of light and heavy vehicles. A heavy vehicle has lower acceleration, deceleration, and maximum velocity than a light vehicle. The optional features selected from our traffic model are the source of variability leading to different traffic scenarios.

Table 2 summarizes the parameter settings in terms of driving and simulation values with respect to each selected feature. For example, when selecting the feature Ra (rainy), parameters σ and τ are increased by 0.1 and 1, respectively. Regarding the emergency feature, if Yes is selected, the time to reach the destination is 500 seconds, but when No is selected this time is 1000 seconds. Note that the simulation time is set according to the emergency feature. So, we have scenarios with different analysis times. Finally, when feature Li is selected, there are 95% of light vehicles and 5% of heavy vehicles. In contrast, when Hv is selected, there are 85% of light vehicles and 15% of heavy vehicles.

The trace information obtained after the simulation of each traffic light program, for a given scenario, is used to compute a set of metrics: vehicles arriving at their destination (VLL), vehicles not arriving at their destination (VNLL), CO_2 emissions (CO_2), NO_x emissions (NO_x), fuel consumption (FUEL), and global journey time (GJT). In this way, it is possible to numerically quantify how accurate a given cycle program is and compare it with all other existing solutions, from human expert's programs or from automatic optimizers.

8. Experiments

This section describes a series of experiments and analyses of the results obtained so as to empirically test our approach. This allows us to put here into practice all initial specifications of our strategy, explained in Section 4, in order to validate the resulting traffic light programs in the ten scenarios generated by means of the execution of the PGFM algorithm.

8.1. Experimental Setting. As stated in the introduction, we include a varied analysis including four specialized algorithms for computing cycle programs of traffic lights: PSOTL, DETL, RS, and SCPG. The first three optimization algorithms are nondeterministic, so we have carried out 30 runs and consequently have obtained 30 different (best) cycle programs for each algorithm. The fourth algorithm (SCPG) is a deterministic technique, so it only computes one optimal cycle program. As our aim is to validate these cycle programs in different scenarios, in the case of nondeterministic algorithms we have carried out 30 independent runs of the simulator per generated scenario (10), algorithm (3), and proposed best

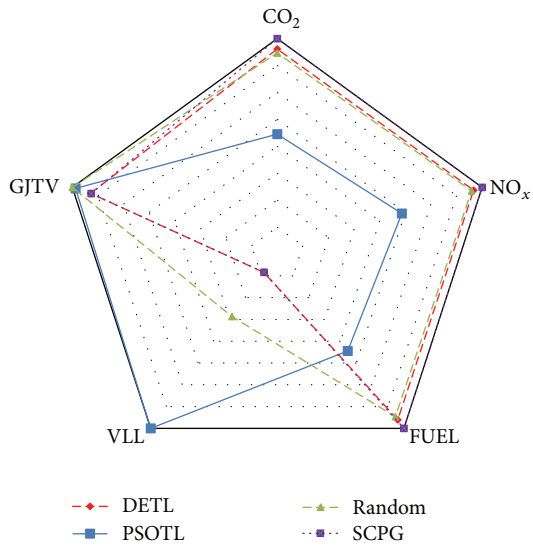

Figure 6: Normalized star diagram containing traffic accuracy metrics for all algorithms compared.

cycle program (30), adding up to a total number of 27,000 executions. This accounts for the considerable effort we have made to study the city in really different conditions of the traffic flow to attain realistic results. In the case of SCPG, we have performed 300 executions (10 scenarios × 30 independent runs).

In order to check whether the differences between the algorithms are statistically significant or just a matter of stochastic noise, we have applied the nonparametric Wilcoxon rank-sum test [49]. The confidence level was set to 95% (p value below 0.05). In addition, so as 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 nonparametric effect size measure $\widehat{A}_{12}$ statistic proposed by Vargha and Delaney [50]. Effect size provides information about the magnitude of an effect, which can be useful in determining whether it is of practical significance or not.

All the executions were run in a cluster of 16 machines with Intel Core2 Quad processors Q9400 (4 cores per processor) at 2.66 GHz and 4 GB memory running Ubuntu 12.04.1 LTS, managed by the HT Condor 7.8.4 cluster manager. In order to offer an approximate idea of the required computational effort to solve this realistic task, we have to note that, in terms of a single core machine, our complete experimentation with 27,300 independent runs would require about 8 months of computation. In contrast, in the used parallel platform, the computation of the valid scenarios required 8.3 days (26.26 seconds per run).

8.2. Experimental Results. In order to illustrate the many results obtained at a glance, Figure 6 shows the performance of algorithms for the selected metrics (VLL, CO_2, NO_x, Fuel, and GJT) for each vehicle and all scenarios. These metrics are normalized for a proper visualization. In this figure, a first observation is that parameters CO_2, NO_x, and Fuel are correlated in almost all algorithms, so, we could consider to deal with them as one single factor. Nevertheless, as one of our main focus is environmental resources consumption/emission saving, we decided to use these parameters as nonaggregate values. The fact of getting correlated results is due to the simulation strategy that indeed follows a realistic model (HBEFA).

Table 3: Best average VLL for each algorithm and scenario. The specific cycle program (out of 30) that generates the best result for this algorithm and scenario is indicated in parenthesis.

	PSOTL		DETL		RS		SCPG
S_1	**91.00%**	(13)	53.93%	(4)	66.63%	(1)	49.73%
S_2	**79.00%**	(3)	50.12%	(26)	56.88%	(4)	45.36%
S_3	**68.98%**	(3)	44.40%	(26)	50.00%	(17)	39.47%
S_4	**52.06%**	(20)	34.41%	(26)	37.57%	(24)	29.12%
S_5	**26.68%**	(14)	19.36%	(26)	20.00%	(20)	15.78%
S_6	**96.76%**	(2)	74.50%	(22)	85.83%	(0)	68.08%
S_7	**73.78%**	(20)	41.18%	(4)	49.28%	(24)	38.83%
S_8	**68.28%**	(2)	40.19%	(4)	47.06%	(24)	36.84%
S_9	**59.02%**	(3)	38.40%	(11)	43.15%	(17)	36.50%
S_{10}	**54.46%**	(20)	34.57%	(26)	38.27%	(20)	29.28%

A second observation in Figure 6 is that PSOTL obtains the maximum number of vehicles arriving at its destination with the lowest fuel consumption, and also with the lowest emissions of CO_2 and NO_x. However, for this algorithm, the global journey time (GJT) is the highest one. The explanation of this counter-intuitive result is that a vehicle, which does not arrive at its destination in the analysis period, is not used for the computation of the metrics. Thus, several vehicles do not arrive at their remote destinations, so, the global journey time will not be increased by the stats of the vehicles with remote destinations. In Figure 6, we also note that SCPG is the worst algorithm in terms of fuel consumption and CO_2 and NO_x emissions, even though it had a low number of vehicles that arrive at their destination. This means that the problem is highly difficult and lacks clear patterns for experts when we go to a large scale optimisation scenario.

In fact, as increasing the number of vehicles that arrive at their destination during the study timeframe (VLL) is an important metric for evaluating how the system prevents traffic jams, we have organized, in Table 3, the average VLL for each scenario and algorithm. We have also highlighted the particular cycle program that obtains the best average of VLL in the 30 independent executions. Therefore, we can easily see that the cycle programs generated by PSOTL are always the best at preventing traffic jams, since they guarantee that almost all vehicles reach their destination within the analysis time. However, the cycle program that obtains the best result is not always the same in all scenarios. For example, for PSOTL, programs 3 and 20 (see the numbers in parenthesis in Table 3) obtain the best results in 3 out of 10 scenarios. This means that there is no single cycle program that shows the best results in all possible traffic scenarios, as expected. These results lead us to consider, in Section 9, the priority assigned to each scenario, in order to choose the best cycle program

Table 4: Number of scenarios where significant differences exist for VLL between the algorithms ($\#\rho$) and Vargha and Delaney's statistical test results ($\widehat{A}_{12}$).

	PSOTL		DETL		RS		SCPG	
	$\#\rho$	$\widehat{A}_{12}$	$\#\rho$	$\widehat{A}_{12}$	$\#\rho$	$\widehat{A}_{12}$	$\#\rho$	$\widehat{A}_{12}$
PSOTL	—		8	0.7901	10	0.7129	4	0.8303
DETL	8	0.2099	—		6	0.3831	0	0.5503
RS	10	0.2871	6	0.6169	—		3	0.6657
SCPG	4	0.1697	0	0.4497	3	0.3343	—	

from all of them. In this way, an unexpected change in traffic conditions will not end in an enormous traffic jam, because the overall best cycle program is in some way more adaptive to a greater number of traffic conditions than an overfitted cycle program.

In order to check whether the differences between the algorithms are statistically significant or not, we have applied the nonparametric Wilcoxon rank-sum test. In Table 4 we show the number of scenarios where significant differences exist between the algorithms. In this regard, we can see that PSOTL is significantly different to RS for all the scenarios (thus, an intelligent algorithm is needed), whereas solutions provided by DETL are not statistically different to the ones provided by the human experts, represented in SCPG (so DETL is just equivalent to the expert's solutions, not better). RS is significantly better than SCPG and DETL by 3 and 6 times, respectively.

Table 4 also reports the effect size measure $\widehat{A}_{12}$ for the VLL metric for all scenarios. We interpret the $\widehat{A}_{12}$ statistic as follows: given a performance measure M, $\widehat{A}_{12}$ measures the probability that running algorithm A yields higher M values than running another algorithm B. In this regard, A represents algorithms in the columns and B represents algorithms in the rows. If these two algorithms are equivalent, then $\widehat{A}_{12} = 0.5$. A value of $\widehat{A}_{12} = 0.3$ entails that one would obtain higher values for M with algorithm A, 30% of the time. As shown in this table, PSOTL's solutions are the best for all scenarios and their differences are of actual importance in practical cases of the city. Numerically speaking, these cycle programs (the ones generated by PSOTL) are better than the ones provided by DETL, RS, and SCPG by 79.01%, 71.29%, and 83.03%, respectively. In fact, these results are labeled as *large effect size* according to Cohen's d scale [51]. These results provide us with some insights into the successful application of PSOTL's cycle programs to real traffic light schedules.

8.3. Focusing on Scenarios. From the point of view of the different generated scenarios, in Figure 7 we can observe the box plots of resulting traces in terms of metrics for all algorithms. Box plots display variation in samples of a statistical population without making any assumptions of the underlying statistical distribution. Box plots show the mean, the interquartile range (in color), and the maximum and minimum value on the extremes. In the box plots of Figure 7, there are no outliers, which are observation points that are distant from other observations. Four metrics are used, two of them measured per vehicle (CO_2 and GJT). Specifically, the second subfigure shows the percentage of time that the journey takes compared to the total analysis time; for example, 40% value means that the average vehicle takes 40% of the analysis time to reach to its destination. The other two measures used are VLL and VNLL. Note that we do not show the resulting box plots for VNLL because they are the complementary results obtained for VLL. These diagrams focus on the scenarios in order to show the variability of traffic measures, while at the same time we have tried to show the results without the specific bias introduced by a particular solving technique.

An interesting observation in Figure 7 lies in the fact that Scenario 6 (S_6) is the most favorable for the GJT and VLL indicators, whereas for the CO_2 indicator the third best result is obtained. The S_6 scenario induces a successful performance of cycle programs. In fact, the main features of S_6 are ideal for enhancing the traffic flow: sunny weather (Su), no emergency (No), few vehicles (Af), and more drivers with a high level of expertise (Il). In contrast, the most unfavorable scenario is S_5, since only a few vehicles arrive at their destination, and the CO_2 thrown up into the atmosphere is the highest for each vehicle in our study. This last scenario has unfavorable weather conditions: rainy (Ra), stormy (St), and foggy (Fg). In addition, there is an emergency (Ys), the number of vehicles is higher (Am), and there are more heavy vehicles (Hv). All these features induce adverse conditions in this scenario (S_5), which negatively influence the traffic flow.

In light of these results, we claim that providing experts with better general programs is possible by using our approach. We consider different traffic conditions in a set of modeled scenarios which provides the experts with unbiased traffic light programs. In contrast, the aforementioned literature (see Section 2) just offers ideal traffic conditions. In addition, here we go one step beyond by weighting those features with more probability of occurrence, but without missing those traffic situations that are more extreme.

9. Prioritized Analysis

In this section, as far as we know, we are the first to apply prioritization techniques that consider all possible traffic scenarios at a time. We analyze how each generated cycle program works for all scenarios as a whole. Since not all scenarios are equally important or frequent, the computed metrics are weighted according to the scenario's priority.

9.1. Analysis of Best Performing Cycle Programs. Designing robust traffic light programs is a mandatory task when

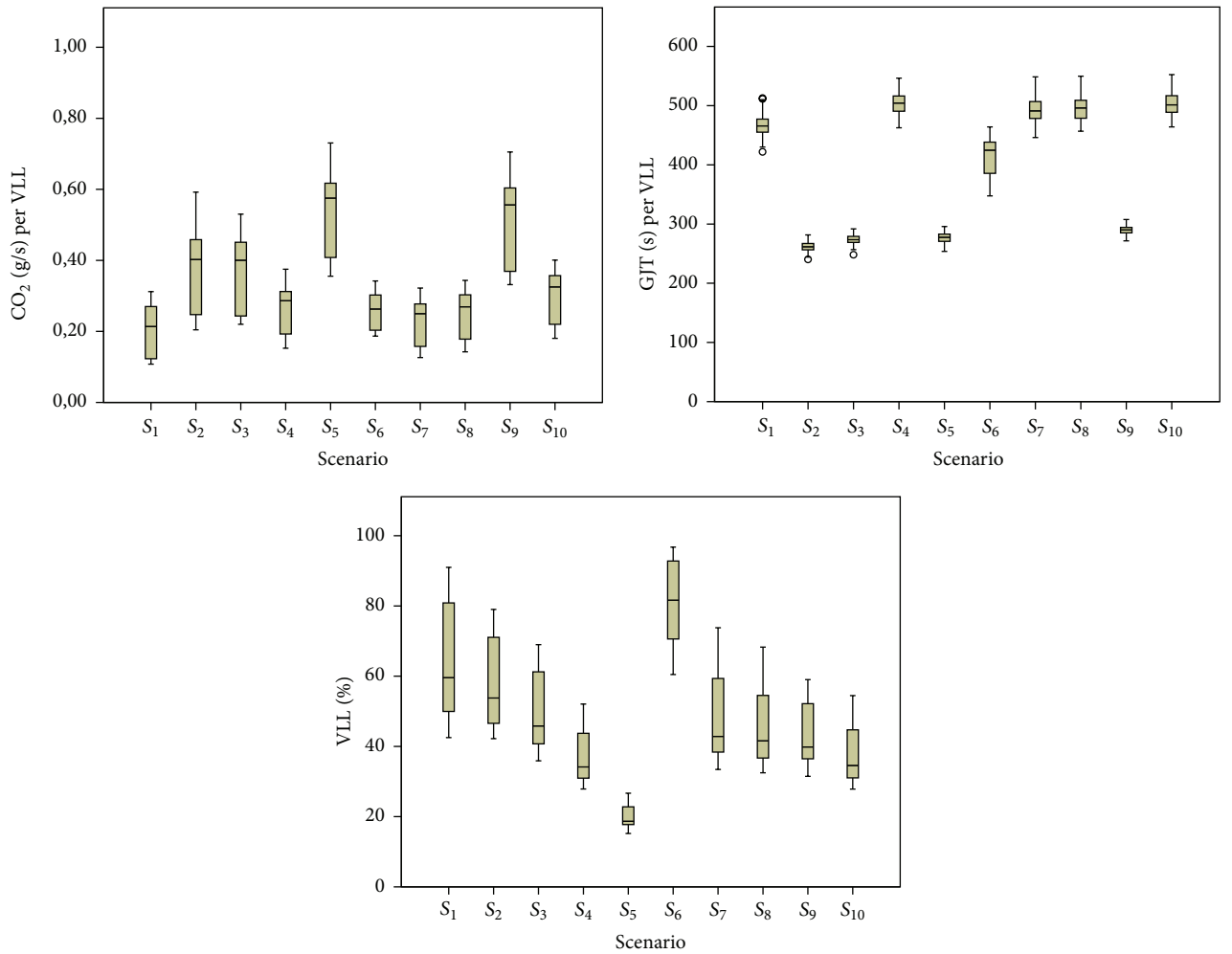

Figure 7: Boxplots of the resulting distribution traces of our studied metrics: CO_2, GJT, and VLL.

tackling vehicular urban environments. In this regard, considering all modeled scenarios as a whole helps us to give a robustness to our solutions, so we have therefore weighted these scenarios according to their priority.

In Table 5, we report our top ten cycle programs for the following metrics: VLL, FUEL, CO_2, and GJT. We have to highlight that the PSOTL13 cycle program is the best solution in terms of VLL, for the most prioritized scenario (S_1 in Table 3). However, when we consider the scenarios altogether PSOTL13 is not the best for VLL, so it appears in second place for this metric in this ranking. This fact was unexpected because of the high weight of S_1; nevertheless the best one for all scenarios as a whole (PSOTL20) is the best in three different scenarios (S_4, S_7, and S_{10} in Table 3).

In Table 5, we can observe the cycle program that is the best for each metric in all scenarios together. So, we only have to decide which metric is more important for us and then set a particular configuration. It is possible that different stakeholders could be interested in setting up different cycle programs. For example, the municipal authorities of the city may be interested in avoiding traffic jams, so the GJT metric should be optimized. However, the national government could impose a CO_2 emissions maximum rate, so the CO_2 metric also ought to be minimized; thus the best option would be to configure the traffic lights with the Program PSOTL20.

9.2. Practical Benefits. Our validation strategy provides experts with many practical benefits. In general, we have alleviated the work of the experts, with the cost reduction this implies, just by setting the priorities in the feature model. From the feature model, they obtain several validation scenarios that accomplish the pairwise coverage criterion and provide us with some confidence for choosing the best traffic lights program. In addition, the priorities of scenarios allow us to select a good solution for most traffic conditions taking into account the weight of the scenarios, previously computed by means of the PGFM algorithm.

Another practical benefit is the flexibility of the proposed approach. Since it is impossible to objectively decide which cycle program of traffic lights is the best, we provide several solutions according to the desired behavior of the system. As we have previously said, it is possible that different stakeholders could be interested in setting up a different cycle

Table 5: Ordered best performing cycle programs after considering all weighted scenarios and the expert's solution (SCPG). PSOTLs are overwhelmingly the best in all cases; DETLs just have a minor role regarding GJT.

VLL-weighted		Fuel-weighted		CO_2-weighted		GJT-weighted	
Program	Value	Program	Value	Program	Value	Program	Value
PSOTL20	359.20	PSOTL24	18.25	PSOTL20	0.1477	DETL29	375.86
PSOTL13	358.17	PSOTL23	18.26	PSOTL13	0.1520	DETL28	376.45
PSOTL2	354.45	PSOTL22	18.28	PSOTL28	0.1525	DETL26	378.47
PSOTL5	347.79	PSOTL25	18.31	PSOTL14	0.1542	DETL16	381.05
PSOTL14	344.72	PSOTL21	18.33	PSOTL16	0.1552	DETL25	381.33
PSOTL1	342.99	PSOTL29	18.34	PSOTL2	0.1553	PSOTL26	382.44
PSOTL3	342.69	PSOTL20	18.35	PSOTL27	0.1554	DETL14	386.05
PSOTL27	342.58	PSOTL28	18.37	PSOTL17	0.1561	DETL11	386.10
PSOTL16	341.67	PSOTL27	18.40	PSOTL3	0.1571	PSOTL28	386.17
PSOTL17	341.58	PSOTL26	18.41	PSOTL23	0.1576	DETL13	387.06
SCPG	200.17	SCPG	24.44	SCPG	0.3353	SCPG	399.27

program. For example, solution PSOTL20 is the best for the metrics of vehicles that arrive at their destination (VLL); however, it is the seventh in fuel consumption. Moreover, this solution is not in the top ten ranking of GJT per vehicle, so we have obtained robust cycle programs for all scenarios, but a different one if you are interested in a different metric.

The benefits of comparing the cycle programs of traffic lights in the proposed scenarios can be now numerically measured. We have compared the expert's solution and the best automatically generated solution for each metric (see values of PSOTL20 with regard to those of SCPG in Table 5). The improvement achieved in solution quality is remarkable, especially for CO_2 emissions, in which we have obtained an improvement of 126.99%. Regarding the vehicles that arrive at their destination, we have obtained an improvement of 79.45% with respect to the experts' solution. The fuel consumption per vehicle is reduced by 33.87%, which is also a highly relevant practical result. Nevertheless, the reduction is slightly lower in the case of vehicle's journey time (GJT), but still better than the expert's solution.

Since this may need revising when implemented in a real city (as do most scientific studies), we could expect a change in the percentages we here include. However, it is so important that we have strong evidence that a final practical benefit will come from using our approach.

10. Threats to Validity

Threats to validity should be considered in any empirical study. So, we have taken care of those that are applicable to our work.

Threats to *internal* validity come from the fact that we have used a single standard assignment for the parameter values of the optimization algorithms based on the author's experience. A change in the values of these parameters could have an impact on the results of the algorithms. Nevertheless, the default values found in the literature may already be sufficient, as analyzed in [52]. We have taken into account the cost of the tuning phase which could become quite high in the case of this large study involving four metaheuristics and more than 27,000 independent runs. This considerable computational cost also partially justifies not going further in our already large analysis.

Regarding *external* threats, we have identified three of them: the assignment of weights to the traffic feature model and the selection of the algorithms. To address the first of these (weights), we have consulted the data of the Mobility Delegation of Málaga Hall, as well as the Spanish National Institute of Meteorology, more concretely, the data from 2012 (Spanish National Institute of Meteorology: http://www.tutiempo.net/clima/Malaga_Aeropuerto/2012/84820.htm; Mobility Delegation of Málaga Hall: http://movilidad.malaga.eu/). The weights were assigned according to the percentage of days of the year that each condition occurred. The second *external* threat is that maybe we have not chosen the best algorithms, but we have included four distinct algorithms that represent a diverse variety of optimization techniques and concepts; even so the experiments took several weeks using a computer cluster. Although, certainly, there might be other algorithms that could potentially yield better results, the ones included are, however, very popular in current research, and in fact many articles just focus on only one or two of them.

The third *external* threat concerns the generation of dynamic traffic light programs. Our aim here is not to generate cycle programs dynamically during an isolated simulation as done in agent-based algorithms [4] but to obtain optimized cycle programs for a given scenario and timetable. In fact, what real traffic light schedulers actually demand are constant cycle programs for specific areas and for preestablished time periods (rush hours, nocturne periods, etc.), which led us to take this focus. Our approach is automatic and can be used in real city traffic centers (in progress). It is an offline step used to build or improve actual city traffic.

11. Conclusions

In this paper, we have proposed a validation strategy for the traffic light programs to be used by the human experts of Smart Mobility. We have carried out a thorough

experimentation aimed at testing our strategy on a large number of different cycle programs, generated by automatic optimizers, as well as by human expert's procedures. The main conclusions that we can draw are as follows:

(i) We propose the use of a traffic feature model with priorities, which allows us not only to reduce the number of traffic scenarios to test the available cycle programs but also to generate the most important scenarios. This can be carried out by means of the PGFM algorithm, proposed in Section 6. This algorithm is able to automatically generate a minimal prioritized test suite from a given feature model. Experts can now work with a reduced set of scenarios with similar fault detection capacity, which means a great cost reduction.

(ii) Our validation strategy allows the experts to numerically quantify the quality of a traffic light cycle program on different scenarios. This quantification is performed on different scoring metrics, like vehicles that arrive at their destination, CO_2 emissions, NO_x emissions, global journey time, and so forth. We could choose a specific cycle program depending on the traffic conditions and then the metric we want to optimize. In this way, we offer a wide range of solutions according to the stakeholder interests.

(iii) We have experimentally shown that there is no single specific cycle program which is the best for all scenarios with numerical models and results (not just with intuitive beliefs). Nevertheless, for the sake of an easy decision-making process, we also provide a method to rank the cycle programs. This is possible because we have taken into account the scenario's priority automatically computed from our feature model.

(iv) With regard to our case study, we have analyzed how the cycle programs generated by four algorithms (PSOTL, DETL, RS, and SCPG) adapt to different traffic conditions (ten different scenarios) in Malaga city. We have compared the generated cycle programs with the solution provided by the experts (SCPG). Considering the prioritization of scenarios, the improvement achieved in solution quality is remarkable, especially for CO_2 emissions, in which we have obtained a reduction of 126.99% compared with the experts' solutions.

As a matter of future work, we plan to consider several scenarios in a single fitness evaluation of solutions in optimization algorithms. Then, we expect to enhance the search procedure of the algorithms in cycle programs for the most influential traffic conditions. Moreover, we plan to deal with a multiobjective model of the problem according to stakeholders interests. As a result, we will provide a Pareto front, the objectives of which are the minimization of the CO_2 emissions, the minimization of the global journey time, or the minimization of traffic jams.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research has been partially funded by the Spanish Ministry of Economy and Competitiveness (MINECO) and the European Regional Development Fund (FEDER), under contract TIN2014-57341-R moveON Project (http://moveon.lcc.uma.es). It has been also partially funded by Grant TIN2014-58304 (Spanish Ministry of Sciences and Innovation), Regional Project P11-TIC-7529/P12-TIC-1519, and by the University of Malaga, under contract UMA/FEDER FC14-TIC36. Finally, Javier Ferrer acknowledges the Grant with code BES-2012-055967 from MINECO.

References

[1] A. Caragliu, C. Del Bo, and P. Nijkamp, "Smart cities in Europe," *Journal of Urban Technology*, vol. 18, no. 2, pp. 65–82, 2011.

[2] C. Harrison, B. Eckman, R. Hamilton et al., "Foundations for smarter cities," *IBM Journal of Research and Development*, vol. 54, no. 4, pp. 1–16, 2010.

[3] R. G. Hollands, "Will the real smart city please stand up?" *City*, vol. 12, no. 3, pp. 303–320, 2008.

[4] S. Lämmer and D. Helbing, "Self-control of traffic lights and vehicle flows in urban road networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2008, no. 4, Article ID P04019, 2008.

[5] G. Lim, J. J. Kang, and Y. Hong, "The optimization of traffic signal light using artificial intelligence," in *Proceedings of the 10th IEEE International Conference on Fuzzy Systems*, pp. 1279–1282, Melbourne, Australia, December 2001.

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

[7] R.-S. Chen, D.-K. Chen, and S.-Y. Lin, "ACTAM: cooperative multi-agent system architecture for urban traffic signal control," *IEICE Transactions on Information and Systems*, vol. 88-D, no. 1, pp. 119–126, 2005.

[8] J. C. Spall and D. C. Chin, "Traffic-responsive signal timing for system-wide traffic control," *Transportation Research Part C: Emerging Technologies*, vol. 5, no. 3-4, pp. 153–163, 1997.

[9] J. Garcia-Nieto, A. C. Olivera, and E. Alba, "Optimal cycle program of traffic lights with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 6, pp. 823–839, 2013.

[10] J. Sánchez, M. Galán, and E. Rubio, "Applying a traffic lights evolutionary optimization technique to a real case: 'Las Ramblas' area in Santa Cruz de Tenerife," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 25–40, 2008.

[11] T. Nagatani, "Effect of speed fluctuations on a green-light path in a 2D traffic network controlled by signals," *Physica A: Statistical Mechanics and Its Applications*, vol. 389, no. 19, pp. 4105–4115, 2010.

[12] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-oriented domain analysis (FODA) feasibility study," Tech. Rep. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[13] M. Mendonca and D. Cowan, "Decision-making coordination and efficient reasoning techniques for feature-based configuration," *Science of Computer Programming*, vol. 75, no. 5, pp. 311–332, 2010.

[14] M. Johansen, O. Haugen, and F. Fleurey, "Properties of realistic feature models make combinatorial testing of product lines feasible," in *Model Driven Engineering Languages and Systems*, J. Whittle, T. Clark, and T. Kühne, Eds., vol. 6981 of *Lecture Notes in Computer Science*, pp. 638–652, Springer, Berlin, Germany, 2011.

[15] M. F. Johansen, Ø. Y. Haugen, and F. Fleurey, "An algorithm for generating t-wise covering arrays from large feature models," in *Proceedings of the 16th International Software Product Line Conference (SPLC '12)*, pp. 46–55, ACM, September 2012.

[16] M. B. Cohen, M. B. Dwyer, and J. Shi, "Constructing interaction test suites for highly-configurable systems in the presence of constraints: a greedy approach," *IEEE Transactions on Software Engineering*, vol. 34, no. 5, pp. 633–650, 2008.

[17] D. Krajzewicz, M. Bonert, and P. Wagner, "The open source traffic simulation package SUMO," in *Proceedings of the Infrastructure Simulation Competition (RoboCup '06)*, Bremen, Germany, 2006.

[18] A. W. Williams and R. L. Probert, "A measure for component interaction test coverage," in *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, vol. 30, pp. 304–311, IEEE, Beirut, Lebanon, June 2001.

[19] M. Grindal, J. Offutt, and S. F. Andler, "Combination testing strategies: a survey," *Software Testing Verification and Reliability*, vol. 15, no. 3, pp. 167–199, 2005.

[20] R. Kuhn, Y. Lei, and R. Kacker, "Practical combinatorial testing: beyond pairwise," *IT Professional*, vol. 10, no. 3, pp. 19–23, 2008.

[21] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys*, vol. 43, no. 2, article 11, 2011.

[22] M. B. Cohen, M. B. Dwyer, and J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," in *Proceedings of the International Symposium on Software Testing and Analysis (ISSTA '07)*, pp. 129–139, ACM, 2007.

[23] R. C. Bryce and C. J. Colbourn, "One-test-at-a-time heuristic search for interaction test suites," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pp. 1082–1089, ACM, London, UK, July 2007.

[24] E. Salecker, R. Reicherdt, and S. Glesner, "Calculating prioritized interaction test sets with constraints using binary decision diagrams," in *Proceedings of the 4th IEEE International Conference on Software Testing, Verification, and Validation Workshops (ICSTW '11)*, pp. 278–285, IEEE, Berlin, Germany, March 2011.

[25] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 61–102, 2011.

[26] F. Ensan, E. Bagheri, and D. Gasevic, "Evolutionary search-based test generation for software product line feature models," in *Proceedings of the 24th International Conference on Advanced Information Systems Engineering (CAiSE '12)*, pp. 613–628, Gdansk, Poland, June 2012.

[27] C. Henard, M. Papadakis, G. Perrouin, J. Klein, P. Heymans, and Y. Le Traon, "Bypassing the combinatorial explosion: using similarity to generate and prioritize t-wise test configurations for software product lines," *IEEE Transactions on Software Engineering*, vol. 40, no. 7, pp. 650–670, 2014.

[28] E. Engström and P. Runeson, "Software product line testing—a systematic mapping study," *Information & Software Technology*, vol. 53, no. 1, pp. 2–13, 2011.

[29] P. A. da Mota Silveira Neto, I. do Carmo Machado, J. D. McGregor, E. S. de Almeida, and S. R. de Lemos Meira, "A systematic mapping study of software product lines testing," *Information and Software Technology*, vol. 53, no. 5, pp. 407–423, 2011.

[30] S. Oster, F. Markert, and P. Ritter, "Automated incremental pairwise testing of software product lines," in *Software Product Lines: Going Beyond: 14th International Conference, SPLC 2010, Jeju Island, South Korea, September 13–17, 2010. Proceedings*, J. Bosch and J. Lee, Eds., vol. 6287 of *Lecture Notes in Computer Science*, pp. 196–210, Springer, Berlin, Germany, 2010.

[31] A. Hervieu, B. Baudry, and A. Gotlieb, "PACOGEN: automatic generation of pairwise test configurations from feature models," in *Proceedings of the 22nd IEEE International Symposium on Software Reliability Engineering (ISSRE '11)*, pp. 120–129, IEEE, Hiroshima, Japan, December 2011.

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

[33] N. M. Rouphail, B. B. Park, and J. Sacks, "Direct signal timing optimization: strategy development and results," in *Proceedings of the 11th Pan American Conference in Traffic and Transportation Engineering*, Gramado, Brazil, January 2000.

[34] P. Holm, D. Tomich, J. Sloboden, and C. Lowrance, "Traffic analysis toolbox volume IV: guidelines for applying corsim microsimulation modeling software," Tech. Rep., National Technical Information Service, Springfield, Va, USA, 2007.

[35] E. Brockfeld, R. Barlovic, A. Schadschneider, and M. Schreckenberg, "Optimizing traffic lights in a cellular automaton model for city traffic," *Physical Review E*, vol. 64, no. 5, Article ID 056132, 12 pages, 2001.

[36] A. M. Turky, M. S. Ahmad, M. Z. Yusoff, and B. T. Hammad, "Using genetic algorithm for traffic light control system with a pedestrian crossing," in *Rough Sets and Knowledge Technology: 4th International Conference, RSKT 2009, Gold Coast, Australia, July 14–16, 2009. Proceedings*, vol. 5589 of *Lecture Notes in Computer Science*, pp. 512–519, Springer, Berlin, Germany, 2009.

[37] L. Peng, M.-H. Wang, J.-P. Du, and G. Luo, "Isolation niches particle swarm optimization applied to traffic lights controlling," in *Proceedings of the 48th IEEE Conference on Decision and Control, Held Jointly with the 28th Chinese Control Conference (CDC/CCC '09)*, pp. 3318–3322, IEEE, Shanghai, China, December 2009.

[38] S. Kachroudi and N. Bhouri, "A multimodal traffic responsive strategy using particle swarm optimization," in *Proceedings of the 12th IFAC Symposium on Transpotaton Systems (CT '09)*, pp. 531–537, Redondo Beach, Calif, USA, September 2009.

[39] K. B. Kesur, *Metaheuristics in Water, Geotechnical and Transport Engineering*, Elsevier, Philadelphia, Pa, USA, 2013.

[40] J. García-Nieto, E. Alba, and A. C. Olivera, "Swarm intelligence for traffic light scheduling: application to real urban areas," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 2, pp. 274–283, 2012.

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

[42] M. Behrisch, L. Bieker, J. Erdmann, and D. Krajzewicz, "SUMO—simulation of urban mobility: an overview," in *Proceedings of the 3rd International Conference on Advances in*

System Simulation (SIMUL '11), pp. 63–68, Barcelona, Spain, October 2011.

[43] M. Clerc, "Standard PSO 2011," Tech. Rep., Particle Swarm Central, 2011, http://www.particleswarm.info/.

[44] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer, Berlin, Germany, 2005.

[45] SPLOT, "Software Product Line Online Tools," 2013, http://www.splot-research.org.

[46] D. Benavides, S. Segura, and A. Ruiz-Cortés, "Automated analysis of feature models 20 years later: a literature review," *Information Systems*, vol. 35, no. 6, pp. 615–636, 2010.

[47] B. Stevens and E. Mendelsohn, "Efficient software testing protocols," in *Proceedings of the Conference of the Centre for Advanced Studies on Collaborative Research (CASCON '98)*, p. 22, IBM Press, 1998.

[48] P. Trinidad, D. Benavides, A. Ruiz-Cortes, S. Segura, and A. Jimenez, "FAMA framework," in *Proceedings of the 12th International Software Product Line Conference (SPLC '08)*, p. 359, Limerick, Irland, September 2008.

[49] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*, Chapman & Hall/CRC, 4th edition, 2007.

[50] A. Vargha and H. D. Delaney, "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.

[51] R. J. Grissom, "Probability of the superior outcome of one treatment over another," *Journal of Applied Psychology*, vol. 79, no. 2, pp. 314–316, 1994.

[52] A. Arcuri and G. Fraser, "Parameter tuning or default values? An empirical investigation in search-based software engineering," *Empirical Software Engineering*, vol. 18, no. 3, pp. 594–623, 2013.