Simulation-driven multi-objective evolution for traffic light optimization

Alessandro Cacco[0000−0002−3251−3551] and Giovanni Iacca[0000−0001−9723−1830]

Department of Information Engineering and Computer Science,
University of Trento, Povo 38123, Italy
alessandro.cacco@studenti.unitn.it, giovanni.iacca@unitn.it

Abstract. The constant growth of vehicles circulating in urban environments poses a number of challenges in terms of city planning and traffic regulation. A key aspect that affects the safety and efficiency of urban traffic is the configuration of traffic lights and junctions. Here, we propose a general framework, based on a realistic urban traffic simulator, SUMO, to aid city planners to optimize traffic lights, based on a customized version of NSGA-II. We show how different metrics -such as number of accidents, average speed of vehicles, and number of traffic jams- can be taken into account in a multi-objective fashion to obtain a number of Pareto-optimal light configurations. Our experiments, conducted on two city scenarios in Italy and different combinations of fitness functions, demonstrate the validity of this approach and show how evolutionary optimization is an effective tool for traffic light optimization.

Keywords: Traffic light optimization · simulation of urban mobility · multi-objective evolutionary algorithm.

1 Introduction

Almost every day, we experience long waits at traffic lights or, even worse, we get stuck in traffic jams. In fact, the ever growing number of cars has made road congestion a phenomenon that is almost impossible to control, especially in larger cities where, even in the presence of efficient public transport, cars are still the preferred choice for private mobility. A recent study from the European Union reported that congestion in the EU costs nearly *100 billion* EUR, or 1% of the EU's GDP, annually[1]. It has been estimated that in 2017 in Europe, where over 60% of the population lives in urban areas of over 10,000 inhabitants and urban mobility accounts for 40% of all CO2 emissions of road transport, an average driver spent in congestion, over a period of 220 working days, a number of hours ranging between 45.73 (UK) and 18.13 (Finland)[2].

These numbers speak for themselves: road congestion is a substantial problem in terms of economy, safety and drivers' comfort, which makes the question of

[1] Source: https://ec.europa.eu/transport/themes/urban/urban_mobility_en

[2] Data collected by TomTom: https://ec.europa.eu/transport/facts-fundings/scoreboard/compare/energy-union-innovation/road-congestion_en

how to enhance urban mobility, while at the same time reducing congestion, accidents and pollution, one of the greatest challenges of our time.

To handle this problem, the traditional approach is to use decentralized (self-regulated) traffic lights, or queue-based models based on traffic statistics. In this sense, the traffic light timing settings are usually tested *live*, and then adjusted *a posteriori* based on the traffic statistics. However, not only this way of managing the problem is expensive and time consuming, but also it can also create confusion in drivers, who would find that the road configurations change over time. Thus, this solution might cause even more congestion and accidents, exacerbating the problem or jeopardizing the safety on the streets.

Here, we tackle this problem by performing computer simulations coupled with numerical optimization. In particular, we focus on how to define the optimal traffic light configuration for controlling the crossroads (in the following, we will refer to them interchangeably also as junctions, or intersections) in a certain area of interest, taking into account different traffic densities (to model time-variant traffic conditions) and different contrasting goals. We model this problem in terms of multi-objective optimization, that we solve *in silico* by means of a realistic mobility simulator, SUMO [1,2], coupled with the well-known multi-objective evolutionary algorithm NSGA-II [3]. The latter is configured with custom mutation/crossover operators capable to handle our solution representation, which encompasses various parameters of traffic junctions and traffic lights timings.

Our approach continues a research path initially opened in [4], a seminal paper where a (single-objective) evolutionary algorithm (EA) was tested for the first time -although in a limited experimental setup- in connection with the microscopic traffic simulator FLEXSYT-II [5], to optimize traffic light timings.

Later research has further investigated the use of Evolutionary Algorithms for traffic optimization, with most of the existing studies being limited to single-junction optimization, and differing mainly for the kind of traffic simulation adopted. For instance, a microscopic simulation approach base on Cellular Automata was proposed in [6,7,8], coupled with a standard Genetic Algorithm (GA). A custom Matlab simulator was implemented instead in [9], where a GA was used to optimally control in "real-time" (i.e., during the traffic light operations) the light timings. A simplified, custom simulator was also used with a GA in [10], while a more realistic traffic simulator called Traffic Simulator Framework (TSF) was used in [11].

More recently, two works by Nguyen et al. [12] and Bravo et al. [13] used for the first time SUMO in connection with EAs to perform traffic light optimization. In fact, these two papers inspired our work. In particular, in [12] SUMO was coupled with a memetic version of NSGA-II (including a local search algorithm) to improve the anytime behavior of the evolutionary algorithm. The main difference w.r.t. our work is that in [12] only one objective (the no. of vehicles entering and leaving the simulation scenario) was related to a specific domain goal, being the other simply the simulation time: in other words, there was no actual domain multi-objective optimization. Furthermore, in [12] the experimental setup consisted of a single urban map, where the traffic lights in only one junction were tuned by the EA. Here, we instead tune the traffic lights of *all* the junctions in a certain area of interest (potentially, an entire city), which makes the problem space much larger and more difficult to explore. This kind of holistic approach is indeed more similar to the one -named HITUL- proposed by Bravo et al. [13], who for the first time called for a shift from traditional junction-local control to global control approaches based on advanced computational resources and techniques. This concept was implemented in [13] by connecting SUMO with a single and multi-objective EA, and tested it on a large-scale scenario in Málaga.

Our work here follows a similar approach: despite the increased computational complexity, as shown in [13] optimizing simultaneously all the junctions in a certain area of interest -thus leveraging synergistic effects between multiple junctions' traffic lights- makes it possible to provide high-quality solutions even in the most complex traffic scenarios that characterize modern urban mobility. However, while our work shares a several similarities with the paper by Bravo et al. [13], we further improve upon their work in at least two main aspects:

- We consider multiple traffic densities *at the same time*, i.e. we look for solutions that are robust against multiple traffic conditions (while in [13] one traffic profile at a time is considered: this feature represents a major novelty of our framework w.r.t. previous works).
- We implement a fully customizable mechanism to combine multiple objectives in a robust optimization fashion: with our framework, the user can for instance decide to optimize w.r.t. the worst or average traffic conditions, reducing the variance of the objectives across various conditions, etc. This feature -that we deem as crucial in practical contexts- is not present in [13].

Finally, we conduct an extensive experimental analysis on two areas of Trento and Milan, Italy, having very different topologies and sizes, thus showing the flexibility of our framework to different city scenarios.

The rest of the paper is organized as follows. Section 2 describes the proposed framework. Section 3 presents the experimental setup and the numerical results. Finally, section 4 concludes this work.

2 Proposed framework

The impact of traffic lights on road performance is strongly related to the specific road network configuration. For instance, different speed limits, lane directions, potential reservations etc. may heavily influence the traffic flow. Furthermore, performance indicators can also be specific, depending on the scenario road types (i.e., arterial, urban, suburban roads) and on particular needs (e.g. reduce incidents or emissions in a certain area).

The fundamental idea of our framework is to account for these aspects by using a microscopic traffic simulator, namely SUMO [1,2], and connect it to a multi-objective evolutionary algorithm, NSGA-II [3], in order to optimize the traffic junction and light configurations in a certain area of interest w.r.t. user-defined fitness functions (and combinations thereof). In the next sections, we describe the details of the proposed framework.

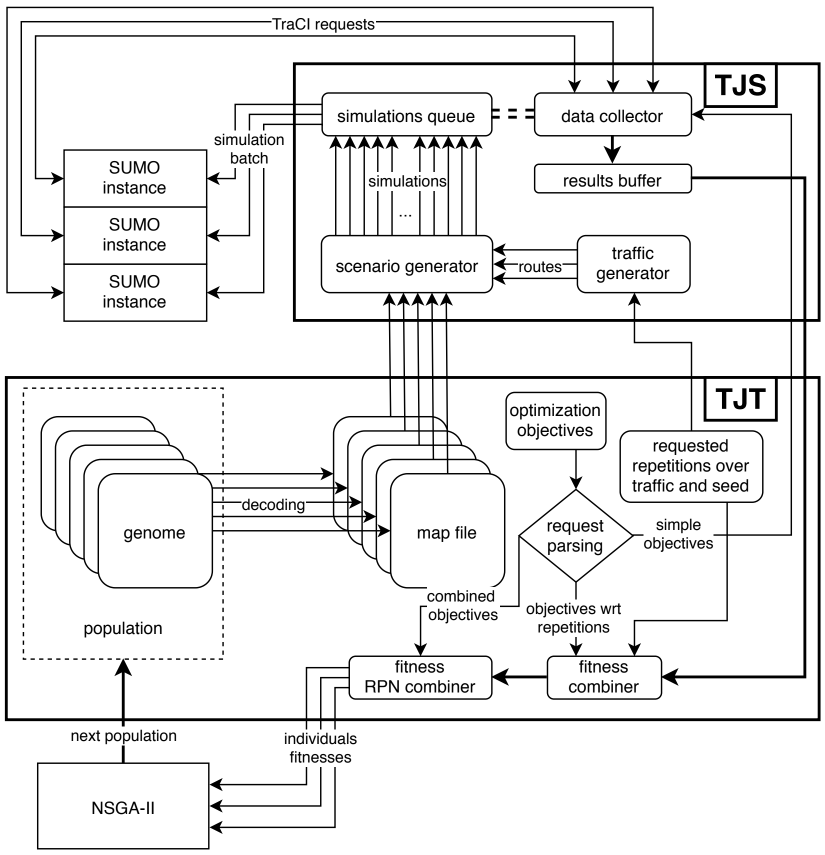

Fig. 1. The framework architecture. The TJS module manages the batch-execution of multiple SUMO simulation instances and provides the data needed for the fitness evaluations. The TJT module acts as an interface between NSGA-II and the simulator.

2.1 Framework architecture

An overview of the architecture is shown in Figure 1, where the main components are represented:

– *Traffic Junction Tuner (TJT)*: builds the initial generation, translates each candidate solution generated by the NSGA-II module to make it available in SUMO, collects the fitness values to allow user-defined combinations, and provides the combined fitness functions to the NSGA-II module.
– *Traffic Junction Tuner - SUMO tools (TJS)*: provides a batch-execution interface for the SUMO simulator, abstracting the TraCI [14] APIs to launch and advance simulations, and gathering all the performance information needed for the fitness evaluation of a scenario. Moreover, it exposes the *NET-CONVERT* and *randomTrips* SUMO tools, needed to prepare the scenario and generate synthetic traffic, respectively.

– *SUMO* [1,2]: the microscopic traffic scenario simulator.
– *NSGA-II* [3]: the algorithm module, based on the *inspyred* Python library [15].

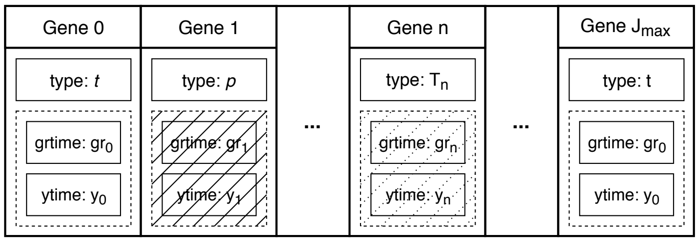

Fig. 2. Graphical representation of the genome. The type T_n of a gene can either be p or t. Genes with type p maintain the traffic light timings values inactive. The length of the genome corresponds to the total number of junctions in the scenario.

2.2 Solution representation

The individuals composing the population that NSGA-II optimizes are particular instances of the input map. Specifically, each traffic junction corresponds to a specific gene in the genome of an individual, which can either be of type p or t: in the former case, the corresponding junction is based on standard priority[3], in the latter case the gene corresponds to a traffic light. This solution representation is based on some fundamental simplifications, which considerably reduce the overall complexity of the optimization process:

– A traffic junction can only be of type t or p. Roundabouts, zip merges and other junction types are not considered, being converted to an allowed type during the initialization.
– Traffic light phases apply to an entire edge incoming to a junction, meaning that dedicated light phases for specific lanes are not possible (e.g. no right-turn preferential lanes).
– Each edge incoming to a junction has a dedicated traffic light phase. Each edge has its own phase and no combinations are allowed (e.g. the classic four-way 2-phases traffic light cannot be modeled).

Additionally, a t gene carries additional parameters, namely *grtime* and *ytime*, corresponding to the traffic light timings for green and yellow lights. A graphical representation of the genome is shown in Figure 2.

2.3 Mutation and crossover operators

At the first generation, individuals are randomly generated with randomized junction types and random timings within the allowed boundaries. Subsequent generations are manipulated via *ad-hoc* mutation and crossover operators. When an individual is randomly selected for mutation, each of its genes is subject to mutation with a certain probability mr, to generate an offspring I_i as follows:

[3] Vehicles on a low-priority edge have to wait until vehicles on a high-priority edge have passed the junction.

– The junction type may switch from t to p (or vice versa) if:

$$\sigma_{I_i} \times \rho_{I_i} > 1, \quad \rho_{I_i} \sim \mathcal{N}(0,1) \tag{1}$$

where σ_{I_i} (initialized to 1) is a hyper-parameter that adaptively decreases over the generations until it reaches a threshold $\epsilon = 0.01$ (see [16]).

– The traffic light timings can change by a random amount (even in the case of a junction type p, where they will be ignored) as follows:

$$grtime'_{I_i} = grtime_{I_i} + 10 \times \tau^g_{I_i}, \quad \tau^g_{I_i} \sim \mathcal{N}(0,1) \tag{2}$$

$$ytime'_{I_i} = ytime_{I_i} + 10 \times \tau^y_{I_i}, \quad \tau^y_{I_i} \sim \mathcal{N}(0,1) \tag{3}$$

As for crossover, it is applied with probability cr on two randomly selected individuals (I_i and I_j), to generate two offspring I_k obtained as follows:

– For each junction, the type is randomly inherited from one of the parents, with an equal probability;
– The traffic light timings are computed as a randomized weighted average of the parents' values:

$$grtime_{I_k} = w^g_{I_k} \times grtime_{I_i} + (1 - w^g_{I_k}) \times grtime_{I_j}, \quad w^g_{I_k} \sim U(0,1) \tag{4}$$

$$ytime_{I_k} = w^y_{I_k} \times ytime_{I_i} + (1 - w^y_{I_k}) \times ytime_{I_j}, \quad w^y_{I_k} \sim U(0,1) \tag{5}$$

A summary of the algorithm parameters is reported in Table 1.

Table 1. Main parameters of the evolutionary algorithm.

Parameter	**Description**
mr	Probability of a mutation happening on a selected individual (0.5).
cr	Probability of crossover happening on two selected individuals (0.5).
$[gr_{min}, gr_{max}]$	Time boundaries for the green light, enforced in initialization, mutations and crossover ($[1, 40]$ sec).
$[y_{min}, y_{max}]$	Time boundaries for the yellow light, enforced in initialization, mutations and crossover ($[1, 10]$ sec).
N_{pop}	Population size (50).
N_{gen}	Number of generations (50).

2.4 Simulations and data collection

In order to evaluate each individual in SUMO, its genome is decoded into a pair of *.nod* and *.tll* files, containing respectively the junctions types and the traffic light logics (i.e., phases definitions). These files are then loaded by the TJS module in order to generate a *.net* scenario file, along with the necessary amount of *.rou* files, containing synthetic traffic routes with a given *period*[4]. Further information about SUMO files can be found in the SUMO documentation [17]. In

[4] By default, this generates vehicles with a constant period and arrival rate of 1/period per second. By using values below 1, multiple arrivals per second can be achieved. Routes are generates such that a new vehicle with a random path and destination is inserted at a certain starting position every *period* seconds, determining a certain average traffic density.

our framework, every individual is then simulated in SUMO with various traffic densities, and for each density multiple times with different seeds (every time with randomly generated vehicle routes). For what concerns the vehicle model, only a standard automobile model has been considered, as per the SUMO default vehicle controller *carFollowing-Krauss*, a modification of the original Krauß controller [18], please refer to the SUMO documentation for further details.

The simulation flow is represented in Figure 3, while Table 2 describes the main parameters of the simulation process.

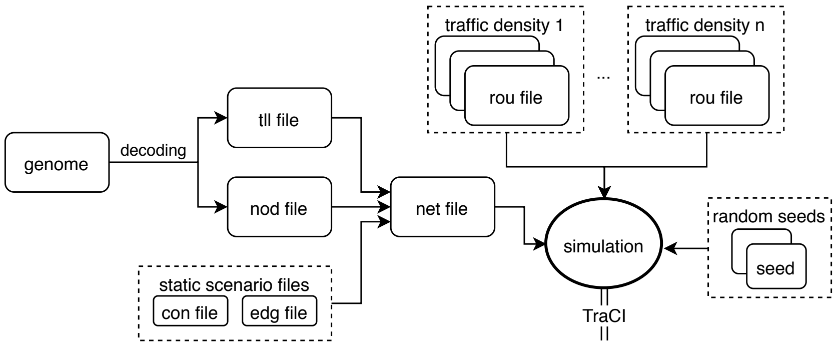

Fig. 3. Simulation flow of a single individual. The genome is decoded into a *.tll* and a *.nod* file, which are then combined with a *.con* connection file and a *.edg* file, representing how the road segments are connected to junctions. The simulations are therefore performed on an individual final scenario file *.net*.

Table 2. Main parameters of the simulation process.

Parameter	Description
traffic-rates	List of traffic *periods* for the synthetic traffic generation. These values are representative of the expected traffic on the scenario (e.g. at different times of the day) (1.5 sec, 2 sec, 2.5 sec).
random-routes	No. of random vehicle routes sets (i.e. *.rou* files) to generate and simulate on the scenario, for each value of *traffic-rates* (5).
single-route-repetition	No. of simulation repetitions to perform for each value of *traffic-rates*, with different random seeds (3).
end	Simulation duration, used also for routes generation (1 h).
step-size	Simulation time-step (1 sec).

2.5 Optimization objectives

The fitness evaluation is based on information gathered via the TraCI API available in SUMO: information about simulation events are retrieved at every simulation step by TJS, to be later combined into the final individual fitness vector considering all traffic densities, routes sets and seed repetitions. Specifically, *simple objectives* (*SO*) can be defined as:

$$\begin{aligned}
&fit_k^{id} \in \{arrived, accidents, teleported, avg\_speed, var\_speed\} \\
&T_k^{RR} \in \{H, L, M, V\} \\
&T_k^{TR} \in \{H, L, M, V\} \\
&SO_k = T_k^{TR}\, T_k^{RR}\, fit_k^{id}
\end{aligned} \tag{6}$$

with fit^{id} being the original value collected via TraCI (see Table 3), and T^{RR} and T^{TR} specifying how to combine the information from the different repetitions (Highest, Lowest, Mean, Variance), respectively for routes sets and traffic densities. Additionally, some of the TraCI-collected values can be normalized prior to be combined, according to the rules in Table 3. It should be noted that the framework can be easily extended to collect further TraCI data, for instance about fuel consumption or emissions.

Simple objectives are then combined into an *objective definition* O_j, which consists of an identifier ϕ_j and a valid reverse polish notation (RPN) expression of SO, operators in RPN_{ops}, and real numbers:

$$\begin{aligned}
RPN_{ops} &= \{+, -, \times, /, min, Max\} \\
tk_j^i &\in \{SO_0\ SO_1\ \dots\ SO_s\} \cup RPN_{ops} \cup \mathbb{R} \\
\phi_j &\in \{+, -, *\} \\
\mathrm{O}_j &= \phi_j[tk_j^0\ tk_j^1\ \dots\ tk_j^n]
\end{aligned} \tag{7}$$

An objective (as handled by NSGA-II) is then given by the evaluation of the corresponding RPN expression on the SO values of the individuals. The ϕ symbol specifies whether the objective has to be maximized, minimized, or just saved without using it as fitness, respectively with $+$,$-$ and $*$. An example of valid notation of (semicolon separated) objectives is as follows:

$$+[\mathit{MMarrived}\; 2\; *]\ ;\ -[\mathit{HMteleported}\; \mathit{MMaccidents}\ *\ 10\ /]\ ;\ -[\mathit{HHaccidents}].$$

3 Numerical results

We have extensively tested the proposed framework on two different maps, respectively of Trento and Milan city center. These map files have been obtained from OpenStreetMap [19] and automatically converted to the SUMO compatible XML files via the NETCONVERT tool. The objective configurations are shown in Table 4. The main difference between the two scenarios are the number of traffic junction to optimize, and the map complexity: Trento has 998 junctions in a smaller space, with many curves and links, while Milan has 3776 junctions that are farther apart, along mostly straight perpendicular roads. Trento is more challenging from in terms of synergy across junctions, whereas on Milan the algorithm must explore a much larger search space.

All the experiments have been executed on an Intel® i9-7940X@3.10 GHz 14 cores-28 threads CPU with 64 GB RAM, running Ubuntu 18.10. The SUMO

Table 3. Fitness request identifiers available in TJS.

Identifier (fit_{id})	**Description**	**Normalization**
arrived	Number of vehicles which were able to complete their entire route and arrive at their final destination.	Arrived over departed vehicles ratio.
accidents	Number of vehicles which were involved in an accident. Note that crashed vehicles are removed from the simulation according to SUMO standard logic.	Crashed over departed vehicles ratio.
teleported	As specified in the SUMO documentation [17], the default behavior for a vehicle in a traffic jam is to "teleport" to the next edge of its route (as such, it can be used as a proxy to measure congestion). The *teleported* identifier keeps track of the number of teleports that happened during the execution. Note that a vehicle may teleport multiple times in the same simulation.	Number of teleports over 10 times the departed vehicles.
avg_speed	Average speed of all the vehicles in the scenario throughout all the simulation duration.	Average speed over 2 times the maximum speed limit of the scenario.
var_speed	Speed variance of all the vehicles in the scenario throughout all the simulation duration.	N.A.

simulations has been performed in parallel, to take advantage of the processor multi-threading capabilities. All code and numerical results are available as Supplementary Information online[5].

For both the scenarios, first we ran 50 test simulations on the original junction configuration, in order to compare the results of the individuals fitness with this *reference scenario*. It should be noted that the simplifications made in Section 2.2 for the genome encoding does not apply to these original maps, so in this case the traffic is able to exploit preferential lanes and dedicated light phases, making the comparison biased against the evolving individuals. In the next two sections we will see the details results of the evolutionary algorithm on the two scenarios.

3.1 Trento optimization runs

The `TNO` boxplots (Figure 5), matrix plots (Figure 6) and parallel coordinates plot (Figure 7) clearly show how the fitness improves over the generations, eventually reaching a performance similar to the *reference scenario* (shown in gray

[5] `https://github.com/alecacco/Traffic-Junction-Tuner`

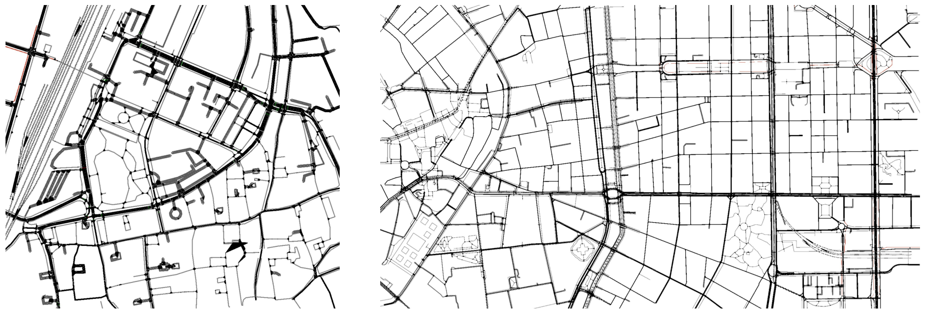

Fig. 4. Urban maps used in the experiments: Trento (left), Milan (right).

Table 4. Objective configurations used in the experiments.

id	Scenario	Objectives	Normalization
`TN0`	Trento	-[MMteleported];-[HHaccidents];+[MMarrived]	Yes
`TN1`	Trento	-[MMteleported];-[HHaccidents];+[MMarrived]; *[VMarrived];*[VMteleported]	No
`TN2`	Trento	-[MMteleported 2 *];-[HHaccidents];+[MMarrived]; -[VMarrived];-[VMteleported]	Yes
`MI0`	Milan	-[MMteleported];-[HHaccidents];+[MMarrived]; *[VMarrived];*[VMteleported]	Yes
`MI1`	Milan	-[MMteleported];-[HHaccidents 20 *];+[MMarrived]; *[VMarrived];*[VMteleported]	Yes

in the matrix and parallel coordinate plots). Similar results (not shown here for brevity) have been achieved with `TN1` and `TN2`, with even better results in the former. Interestingly, an improvement over the non-fitness data (i.e., metrics that are not explicitly optimized by the evolutionary algorithm), **[VMarrived];*[VMteleported]*, has also been observed in `TN1`: this highlights how the optimization collaterally reduced the performance difference across different traffic densities. On the other hand, while `TN2` objectives included explicitly the optimization of these parameters, the best individuals didn't perform as well as the best ones of `TN1` (this might be due to a more difficult identification of Pareto-optimal solutions with an increased number of objectives). Additionally, `TN2` progress was slower, also probably due to the higher number of objectives.

Overall, the `TN` experiments can be considered successful, especially considering the complexity of the Trento scenario, our simplifications in the model, and their direct impact on traffic junction synergies.

3.2 Milan optimization runs

Due to the characteristics of the Milan scenario, individuals are expected to evolve slower than the ones in the case of Trento. In fact, this behavior has been observed in the `MI0` and `MI1` runs, still achieving interesting results: the boxplots in Figure 8 show how the fitness values improve over the generations, while the matrix plot in Figure 9 gives more insights on the progress w.r.t. the reference scenario (shown in gray). As it can be observed in the latter the reference is almost outperformed by the best evolved individuals. The parallel coordinates plot (Figure 10) also show how the evolution progressed towards the reference fitness, reaching it and outdoing at the last generation.

The difference between `MI0` and `MI1` runs is the second objective (as stated in Table 4), which in the former case caused the decrease of accidents to be less significant, while in the latter the *20 ** factor better guided the optimization, reaching the reference *HHaccidents* fitness.

Overall, the `MI` runs achieved satisfying results, in particular considering the scale of the scenario and, again, the simplification that has been made on the genome encoding.

4 Conclusions

In this paper we presented a general framework for traffic light optimization. The framework couples a realistic simulator for urban mobility, SUMO, with a multi-objective Evolutionary Algorithm based on NSGA-II. We tested the framework on two city maps with various combinations of optimization goals, taking into account the robustness of each solution (i.e., a traffic light configuration generated by NSGA-II) w.r.t. different levels of traffic density. This latter aspect shows how the tool can be applied in practical urban scenarios with time-variant traffic density, giving the user the possibility to optimize e.g. w.r.t. the worst or the average density scenarios.

Our results provided a number of interesting insights on the possible trade-off solutions that can be obtained on the two maps, taking into account conflicting goals. Furthermore, they showed the flexibility of the proposed framework, that has been designed such that it can be easily extended to additional goals and arbitrary combinations thereof.

In future works, we will include into our framework the possibility to take into account different types of vehicles (such as buses, trams, bicycles, etc.), each with its own traffic behavior and density. In addition to that, we will focus our attention on more detailed intersection models, and test the framework on larger and more complex city maps. As for what concerns the algorithmic details, we will try to investigate alternative mutation/crossover operators, in order to characterize their behavior, and compare single vs multi-objective evolutionary algorithms as well as incremental local search methods.

Finally, we will consider the possibility to provide our framework as a Software-as-a-Service, available to the general public and the local administrators in order to make better informed city plan choices.

Acknowledgments

We thank Andrea Ferigo for his contribution to a preliminary implementation of the framework proposed in the paper.

References

1. Krajzewicz, D., Erdmann, J., Behrisch, M., Bieker, L.: Recent development and applications of SUMO - Simulation of Urban MObility. International Journal On Advances in Systems and Measurements **5**(3&4) (2012) 128–138
2. Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Fltterd, Y., Hilbrich, R., Lcken, L., Rummel, J., Wagner, P., WieBner, E.: Microscopic Traffic Simulation using SUMO. In: International Conference on Intelligent Transportation Systems. (2018) 2575–2582
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE transactions on evolutionary computation **6**(2) (2002) 182–197
4. Taale, H., Bäck, T., Preuss, M., Eiben, A., De Graaf, J., Schippers, C.: Optimizing traffic light controllers by means of evolutionary algorithms. In: European Congress on Intelligent Techniques and Soft Computing. Volume 3. (1998) 1730–1734
5. Taale, H., Middelham, F.: Flexsyt-ii-a validated microscopic simulation tool. IFAC Proceedings Volumes **30**(8) (1997) 883–888
6. Sanchez, J.J., Galan, M., Rubio, E.: Genetic algorithms and cellular automata: A new architecture for traffic light cycles optimization. In: Congress on Evolutionary Computation. Volume 2., IEEE (2004) 1668–1674
7. Turky, A.M., Ahmad, M.S., Yusoff, M.Z.M.: The use of genetic algorithm for traffic light and pedestrian crossing control. International Journal of Computer Science and Network Security **9**(2) (2009) 88–96
8. Turky, A.M., Ahmad, M., Yusoff, M.Z.M., Hammad, B.T.: Using genetic algorithm for traffic light control system with a pedestrian crossing. In: International Conference on Rough Sets and Knowledge Technology, Springer (2009) 512–519
9. Singh, L., Tripathi, S., Arora, H.: Time optimization for traffic signal control using genetic algorithm. International Journal of Recent Trends in Engineering **2**(2) (2009)
10. Teo, K.T.K., Kow, W.Y., Chin, Y.: Optimization of traffic flow within an urban traffic light intersection with genetic algorithm. In: International Conference on Computational Intelligence, Modelling and Simulation, IEEE (2010) 172–177
11. Gora, P.: A genetic algorithm approach to optimization of vehicular traffic in cities by means of configuring traffic lights. In: Emerging Intelligent Technologies in Industry. Springer (2011) 1–10
12. Nguyen, P.T.M., Passow, B.N., Yang, Y.: Improving anytime behavior for traffic signal control optimization based on nsga-ii and local search. In: International Joint Conference on Neural Networks, IEEE (2016) 4611–4618
13. Bravo, Y., Ferrer, J., Luque, G., Alba, E.: Smart mobility by optimizing the traffic lights: A new tool for traffic control centers. In: International Conference on Smart Cities, Springer (2016) 147–156
14. API, T.: `https://sumo.dlr.de/docs/TraCI.html`
15. inspyred: `https://pypi.python.org/pypi/inspyred`
16. Garrett, A.: inspyred: bio-inspired algorithms in Python (2015)
17. SUMO: `https://sumo.dlr.de/docs`
18. Krauß, S.: Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. PhD thesis, Dt. Zentrum für Luft-und Raumfahrt eV, Abt. Unternehmensorganisation und-information. (1998)
19. OpenStreetMap: `https://www.openstreetmap.org`

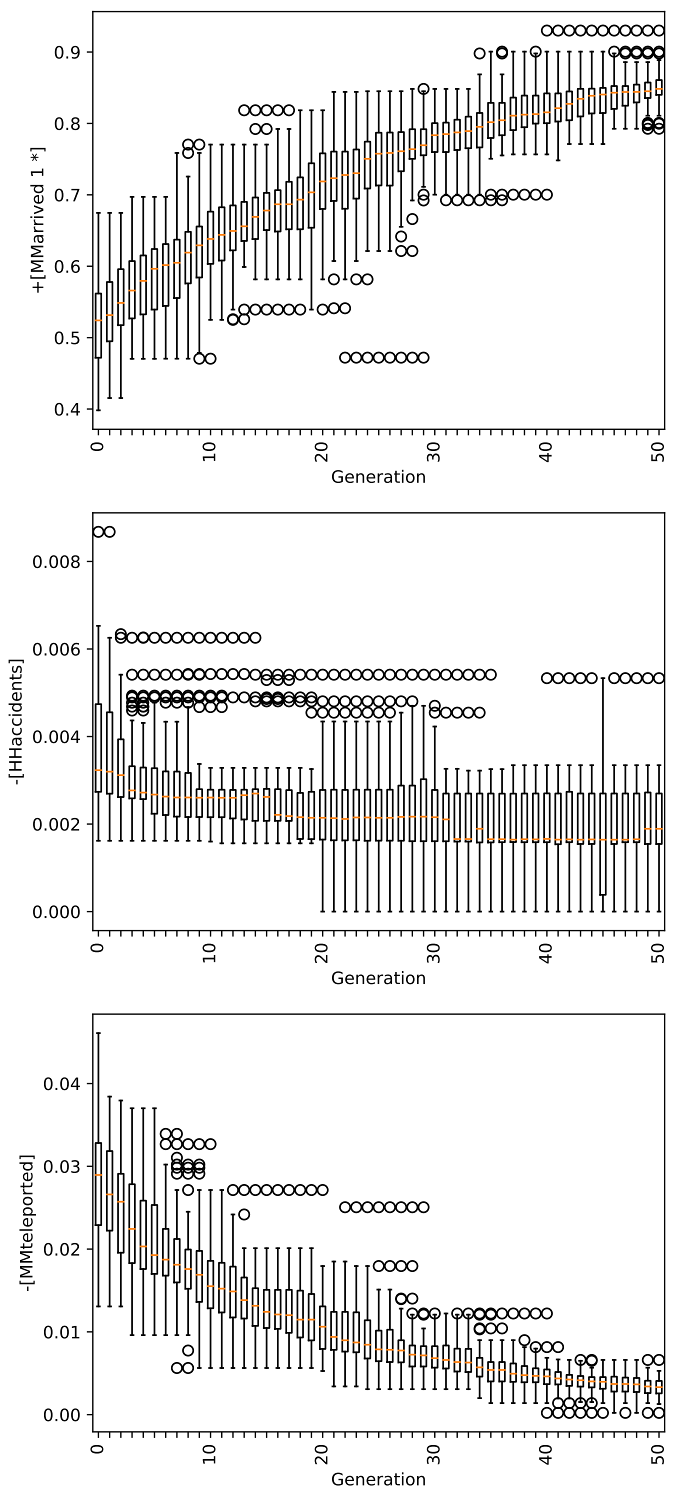

Fig. 5. TNO scenario: generational plot. Each boxplot represents the distribution of the objective function value within a generation.

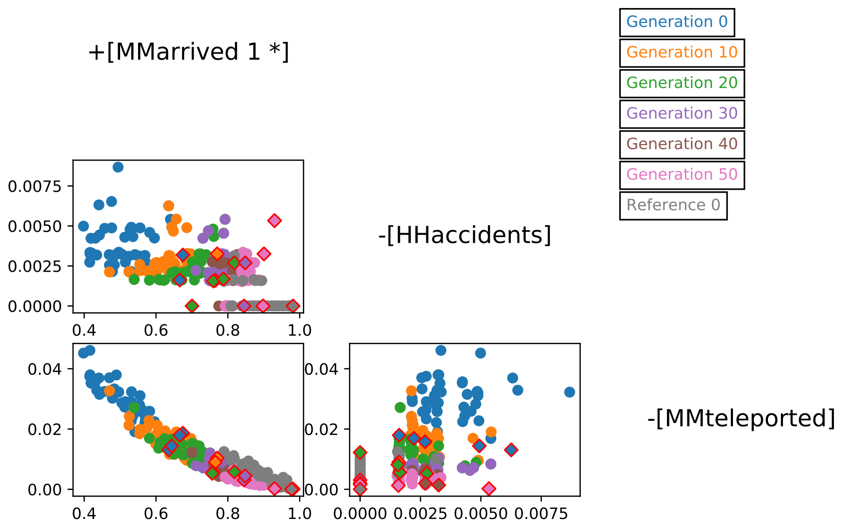

Fig. 6. `TNO` scenario: matrix plot. The labels on the diagonal indicate the x axis of the plots in the same column and the y axis of the plots in the same row. The diamond markers indicate the final Pareto-optimal solutions.

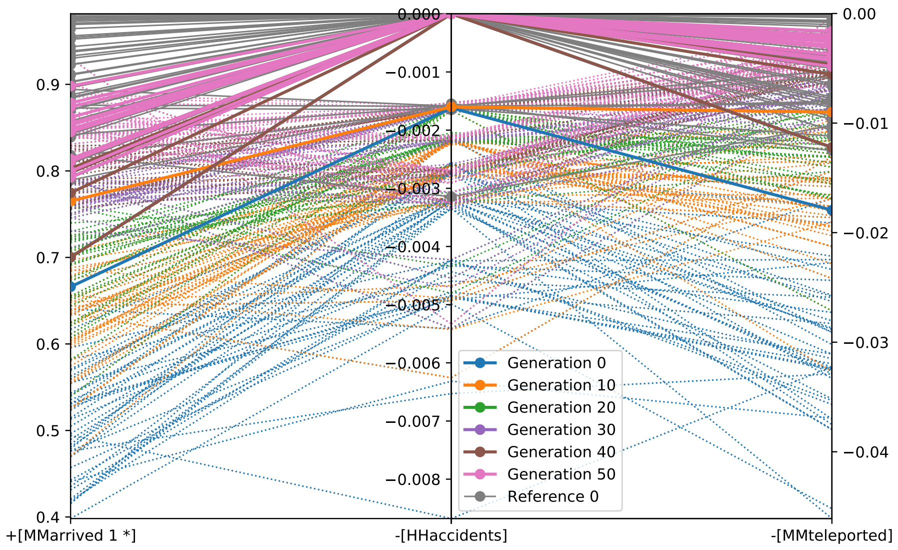

Fig. 7. `TNO` scenario: parallel coordinates plot. The vertical axes indicate the optimization objectives, while the connected lines represent the fitness of each individual. The Pareto-optimal solution (at each generation) and the reference solutions are highlighted with thicker lines.

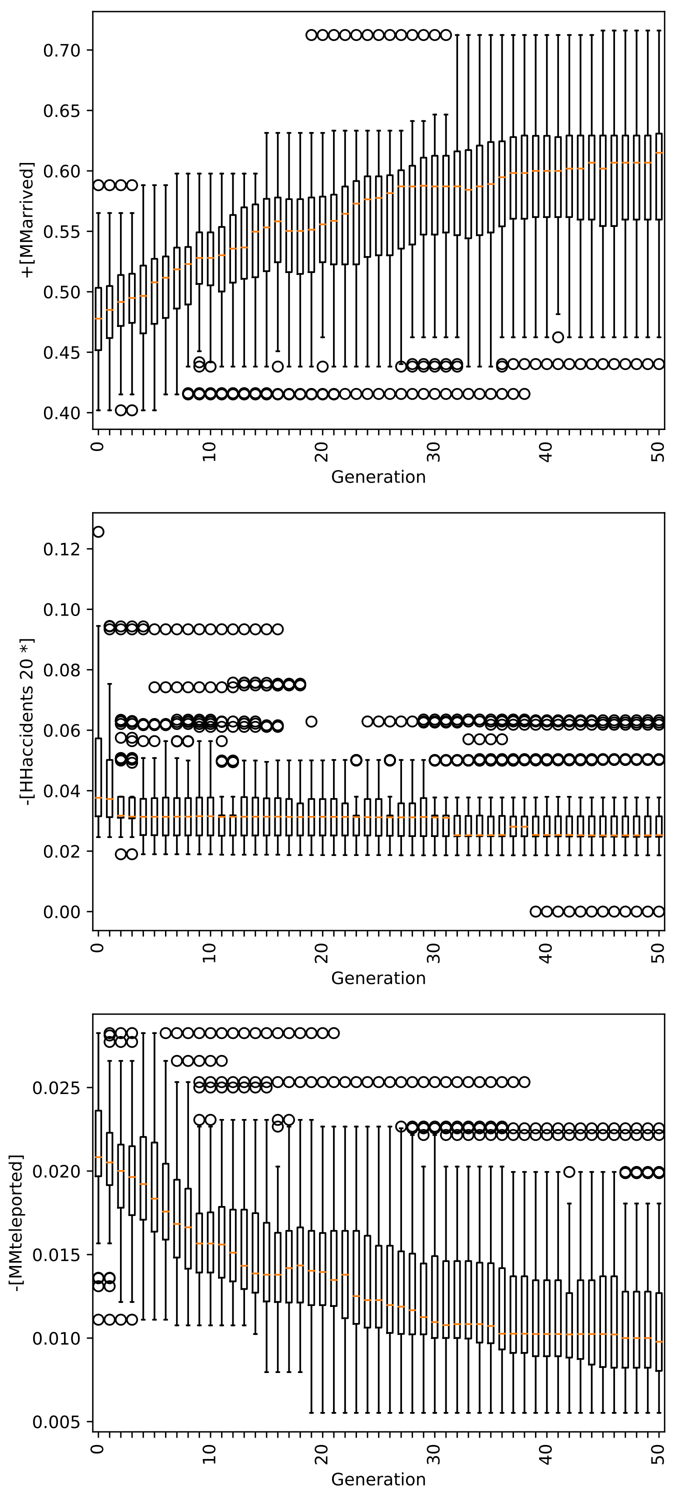

Fig. 8. `MI1` scenario: generational plot. Each boxplot represents the distribution of the objective function value within a generation.

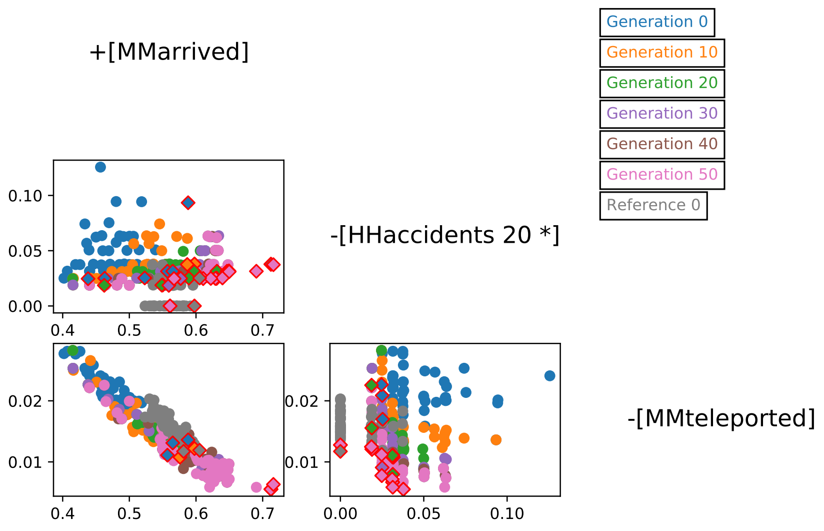

Fig. 9. `MI1` scenario: matrix plot. The labels on the diagonal indicate the x axis of the plots in the same column and the y axis of the plots in the same row. The diamond markers indicate the final Pareto-optimal solutions.

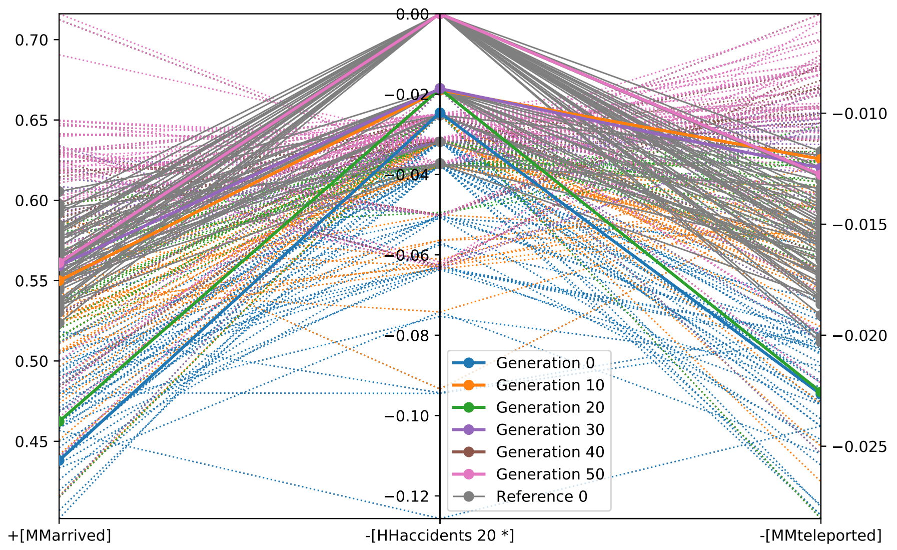

Fig. 10. `MI1` scenario: parallel coordinates plot. The vertical axes indicate the optimization objectives, while the connected lines represent the fitness of each individual.The Pareto-optimal solution (at each generation) and the reference solutions are highlighted with thicker lines.