A SUMO-Based Parking Management Framework for Large-Scale Smart Cities Simulations

Lara CODECÁ
EURECOM
Communication Systems Department
06904 Sophia-Antipolis, France
Codeca@eurecom.fr

Jakob ERDMANN
German Aerospace Center
Institute of Transportation Systems
Rutherfordstr. 2, 12489 Berlin, Germany
Jakob.Erdmann@dlr.de

Jérôme HÄRRI
EURECOM
Communication Systems Department
06904 Sophia-Antipolis, France
Haerri@eurecom.fr

***Abstract*—We collectively decided that investing in smart cities, and consequently smart mobility, is the appropriate direction to solve traffic congestion and sustainable growth issues. Among the problems linked to traffic congestion, we find the complexity of efficient multi-modal commuting and the eventual search of a parking spot. Ideally, mobility should be a transparent service for the users and the quest to find parking should not exist in the first place. In order to achieve this goal, we need to study large-scale parking management optimizations. Recently we reached the computational power to simulate and optimize large-scale cities, but problems such as the complexity of the models, the availability of a reliable source of data, and flexible simulation frameworks are still a reality. We present the general-purpose Python Parking Monitoring Library (PyPML) and the mobility simulation framework. We discuss the implementation details, focusing on multi-modal mobility capabilities. We present multiple use-cases to showcase features and highlight why we need large-scale simulations. Finally, we evaluate PyPML performances, and we discuss its evolution.**

***Index Terms*—Parking Management, Information Uncertainty, Large-Scale Mobility Optimizations.**

I. Introduction

Traffic congestion presents a widespread problem in large urban agglomerations. The increasing number of motorized vehicles requires transport infrastructure upgrades; however, these infrastructures have issues with keeping up with the growth. The study of smart cities aims to improve the services by optimizing the available resources, and to provide new directions for future urban planning. Among the daily problems we face in our cities, parking presents a variety of issues that span from time wasted looking for a parking spot, to increased traffic congestion and pollution in the area [1].

Ideally, parking should be seamlessly integrated into the trip planning. A person should just set personal preferences and decide the destination; the system should determine the most efficient trip in terms of time, preferences, and energy consumption. Parking availability, location, and integration in the mobility infrastructure are bound to play a significant role while deciding which the most efficient plan is. Commuting is a significant source of traffic congestion [2]. Multi-modal Park & Ride (P+R) commuting solutions should be provided seamlessly as a service, and the selection of the parking spot should be planned before reaching the congested area because the possible alternative will more likely be suboptimal.

Over time, parking management studies focused on technology and sensors [3], [4], local optimizations such as time sharing and bookings [5], and crowd-sourcing approaches to finding free spots [6]. Large-scale parking management studies are more complicated to achieve for various reason: (i) the lack of reliable aggregated information, (ii) the complexity of city-wide mobility simulation, and (iii) the absence of flexible optimization tools to implement and evaluate possible solutions.

Among the information required for parking management studies, we find location and capacity, current and future availability, and the possibility of bookings and subscriptions. Major parking areas are often associated with additional mobility services such as airports, train and bus stations, and car sharing. Additional meta-data on the surroundings would provide significant insight and tuning possibilities for a multi-modal approach to parking management and optimization. All the above-mentioned data should be collected and aggregated in real-time in order to take effective decisions. In reality, some degrees of uncertainty has to be considered in order to provide robustness to the system.

Uncertainty may come from different sources. Intelligent parking areas are equipped with a variety of wireless and wired sensors used to monitor the status of the spots [3]. These sensors may fail due to interference, systems flaws, and deterioration over time. Another source of uncertainty is linked to the processing and aggregation delay of the gathered data. Furthermore, malicious behavior has an impact on the consistency of the information associated with parking management [7], [8]. Last but not least, moving in the direction of smart cities and intelligent interconnected vehicles, connectivity issues play their role in the coherence of the crowd-sourced information, even without taking into account malicious behavior.

We decided to study large-scale parking management optimizations using Simulation of Urban MObiltiy (SUMO) [9] with the Monaco SUMO Traffic (MoST) Scenario [10], and we implemented the general-purpose Python Parking Monitoring Library (PyPML) and additional Traffic Control Interface (TraCI) [11] API. PyPML is freely available on GitHub `https://github.com/lcodeca/pypml` under GPLv3 license.

The use of PyPML with the additional TraCI API we implemented enables the study of large-scale parking management optimizations. The abstraction provided by PyPML allows flexibility in the implementation of various optimizations, focusing on the problem at hand, and not the data aggregation. The additional TraCI APIs are meant to provide the tools to easily actuate the optimization, with a focus on flexible multi-modal optimization strategies. For example, using this framework is possible to study the impact of various parking optimizations on time spent looking for a parking spot, pollution generated by cruising for parking, efficient usage of multiple parking areas, access and location of multi-modal hubs, and more.

In the remainder of this paper, we discuss the state-of-the-art, the simulation framework, and we present and evaluate multiple optimizations. Section II presents the current state of parking management options and simulation tools. The simulation framework and the implementation details are discussed in Section III. In Section IV we showcase and evaluate multiple parking management optimizations, and in Section V we evaluate PyPML performances. Finally, we discuss conclusion and future work in Section VI.

II. Related Work

Since the '90s, a lot of work has been done on Parking Guidance and Information (PGI) systems, Parking Reservation Systems (PRS), and Intelligent Parking Assistant (IPA) applications. Extensive surveys can be found in [12] and [13], with a focus on wireless communication in [14].

The default architecture of an intelligent parking management system is mainly composed of the monitoring mechanism (sensor and video-based), the information gathering and dissemination, the telecommunication infrastructure, and finally the (centralized or distributed) controller. An old example of this infrastructure based on wireless sensors network can be found in [15]. The authors present and evaluate a prototype based on low-cost wireless sensors disseminated in a parking lot, where the information detected are reported to a central database. Another example of an intelligent parking management infrastructure is presented in [16]. Along with wireless sensors, the authors discuss the use of high-range smart devices and wireless payment solutions to enable convenient management of public parking in the urban area. With this focus, they present a conceptual architecture of IPA and the prototype-scale simulations of the system.

Over time, multiple approaches have been used to optimize the parking management systems.

A critical factor in the optimization is the parking availability prediction. In [17] the authors present an analytical model for both on and off-street parking availability predictions based on multivariate spatiotemporal models. They use the model to predict parking availability and the prediction errors are used to create the recommendations. The results are validated using real-time parking data in the areas of San Francisco and Los Angeles.

Another approach is based on resource reservation and pricing. For example, iParker [18] is a system based on dynamic resource allocation and pricing using mixed-integer linear programming. iParker offers guaranteed parking reservations with the lowest possible cost and searching time for drivers, and the highest revenue and resource utilization for parking managers. In the paper, the authors propose pricing policies that can be implemented in practice. In [19], the authors present a smart parking platform based on genetic optimization algorithms. The architecture is based on IoT, and the free parking space closest to the current location is computed with a genetic algorithm. An agent-based approach is presented in [20], where a dynamic parking negotiation and guidance platform is used to consider negotiable parking prices. The agent-based coordination network mediates the interaction between drivers and car park operators. The agents have capabilities including planning, monitoring, and coordination.

All the solutions mentioned above are validated using a small number of parking lots, disseminated over a restricted area. The motivation is quite simple: the complexity of analytical models and the scalability of large-large scale parking management simulation always presented an issue. Nonetheless, we have the necessity of large-scale smart city optimization in order to achieve sustainable growth in our cities. PyPML and the simulation infrastructure we present aims to address precisely this issue, enabling large-scale parking optimizations with a focus on multi-modal commuting infrastructure.

III. Simulation Framework

SUMO [9] is an open-source general-purpose microscopic traffic simulator that includes various tools to import, model, and reshape traffic mobility. The active community enables the support required to implement extensions, and the developers made possible for us to include the additional TraCI API in SUMO. The functionality required to work with parking areas are implemented in SUMO v1.0.0.

In order to work with large-scale parking optimizations, a state-of-the-art mobility scenario is required. We chose MoST Scenario[1] [21] because it is based on real data and contains parking area information. The MoST Scenario details and usage are presented in Section IV-A.

Figure 1 presents the overview of the simulation framework. SUMO, at the top in yellow, loads the configuration files from MoST Scenario, and it runs the simulation. The solid arrows are calls to the TraCI API. PyPML, to the right in blue, reads from SUMO and aggregates the data. The dashed arrows are calls between the actual optimization, at the bottom in green, and PyPML. We decided to separate the monitoring from the actuation to achieve more flexibility and enhance performances, empowering the user with full control of the additional information that can be requested to SUMO, and the actuation.

[1]MoST Scenario: https://github.com/lcodeca/mostscenario (Last access: 9/2018)

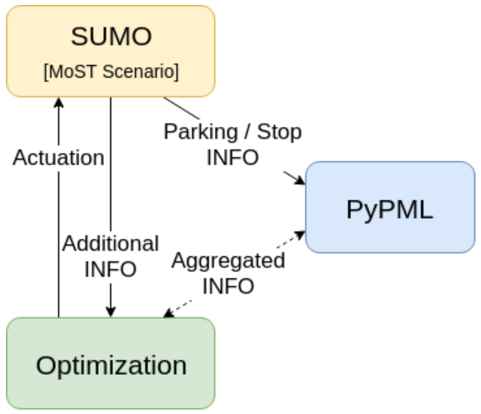


Fig. 1. Simulation Framework Overview.

In the following paragraphs, we discuss in more detail the implementation of this framework; the code is open source and available on GitHub[2].

A. SUMO and Parking Areas

SUMO has built-in parking capabilities. Parking areas and street parking spaces are defined similarly. These areas are used to remove the vehicles from the network and are defined using the `<parkingArea>` element in the XML configuration files [22]. It is possible to define the position and angle of each parking spot to obtain fish-bone or parallel parking. Each parking area has a fixed capacity defined through its attributes and child elements. Vehicles cannot enter the parking area once this capacity is reached.

SUMO includes a model for dynamically rerouting to an alternative parking area once the target area is at capacity. This model uses location-based triggers to encode the area in which a driver gains knowledge about the capacity conditions (e.g., when reaching the full parking space or earlier via parking guidance system). To make use of this model, the user must define rerouter elements that declare the trigger location and the list of alternative parking areas associated [23], with both probability and visibility. The probabilities are normalized over all the definitions in a rerouter, and if the visibility is set, the occupancy of the parking area is known before reaching it. This mechanism allows the model of line-of-sight (visibility is false) and the presence of a parking information system (visibility is true) as well. To model individual preferences concerning the choice of the alternative parking area, it is possible to define weights associated with probability, capacity, occupancy, distance, and time.

Nonetheless, there are limitations. The capacity of a parking area is fixed and it is not directly linked to the dimensions of the parking vehicles. Some workarounds can be done to overcome the problem, but the solution is cumbersome and not flexible.

There is no built-in solution to study parking subscription, variable parking capacity by vehicle type and rerouting choice linked to multi-modal solutions.

B. TraCI and Parking Management

In order to interact with the simulation at runtime, we use TraCI. Some API required to work with parking management were already available; others were implemented and integrated into version 1.0 of SUMO. The complete documentation is available on-line[3]. The *capacity* and the current *occupancy* of a parking area are accessible using the generic `simulation.getParameter` function. For each simulation step, the list of vehicle identifiers that are *stopping* and *restarting* is available through `simulation.getParkingStartingVehiclesIDList` and `simulation.getStopEndingVehiclesIDList` respectively. Before SUMO 1.0, the association between the vehicle that is stopping and the parking area was unavailable. In order to create the connection, we implemented `vehicle.getNextStops`, a generic interface that returns the list of upcoming stops (both parking areas and other kinds of stops) associated with a vehicle.

Moving from parking monitoring to optimization, other APIs are required to actuate the computed optimization. Among the APIs already available we have `simulation.findRoute` and `simulation.findIntermodalRoute`, two interfaces used to *find* the best *route* and the best *inter-modal route* respectively. Additionally, the collection of APIs to *create a person trip* are already implemented:

- `person.appendDrivingStage`,
- `person.appendWalkingStage`,
- `person.appendWaitingStage`.

These functions can be used to build a multi-modal trip: the driving stage can be used for both riding a vehicle (e.g., public transports and car sharing) and drive a vehicle.

Until now, the possibility of rerouting a vehicle to a different parking area was not implemented, and the association between the vehicle and the passengers was not available at runtime. We implemented `vehicle.rerouteParkingArea` to change the parking destination of a vehicle; by default, the behavior is similar to the rerouters: a new route to the parking area is defined, and to maintain consistency in the simulation, the walking route for the passengers is set. Additionally, we implemented `vehicle.getPersonIDList` to obtain the list of people in the vehicle, necessary information to update the plan for the passengers in a case more efficient solution can be found. For example, if the selected parking destination is full, but an alternative parking area that is further away by walk has the current occupancy very low (higher probability of finding the free spot on arrival) may be a better option. In case there is a bus line available between the new parking area

[2]SUMO repository: https://github.com/eclipse/sumo (Last access: 9/2018) PyPML repository: https://github.com/lcodeca/pypml (Last access: 9/2018)

[3]TraCI documentation: http://www.sumo.dlr.de/daily/pydoc/traci.html (Last access: 9/2018)

and the final destination of the passengers, it becomes the best choice, even while being further away.

While performing an optimization, it is essential to take into account that the presence of rerouters alters the simulation, and SUMO internal routines have priority over TraCI API.

C. PyPML: Parking Monitoring

PyPML is a parking monitoring library written in Python that uses TraCI to retrieve and aggregate information on the simulation. The monitoring is built using the `TraCI.StepListener` class provided by SUMO. This implementation implies that at each simulation step the monitoring function implemented by PyPML collects and aggregate data. PyPML uses TraCI subscriptions [24] everywhere possible to improve performances.

For each parking area, we aggregate by default the current occupancy, the occupancy over time, and the projections by vehicle type. With projections, we mean the intention of a vehicle of using a parking area. This information is updated monitoring the changes in the upcoming stops for each vehicle. Additionally, both capacity and subscription for each vehicle type can be set and modified over time, enabling the modeling of reserved areas. Finally, for each parking area, the uncertainty is modeled with a normal distribution, defined through parameters μ and σ.

The set of parking areas handled by PyPML can be directly initialized using a dictionary or can be updated at runtime using getter and setter functionalities. It is possible to limit the number of vehicles monitored in order to minimize the overhead. The fastest option monitors only the vehicles that have a stop defined from the beginning of the simulation. The slowest option monitors every vehicle in the simulation, allowing more flexibility in the final optimization, but paying its price with additional overhead.

In order to access the aggregated information, iterators are provided for passengers, vehicles, parking areas, and rerouters. In case rerouters are defined, the library outputs a warning as a reminder that internal SUMO routines have priority over TraCI API, with a possible impact on the optimization.

The main functionalities implemented in PyPML are:

- `PyPML.get_free_places`,
- `PyPML.get_closest_parking`,
- `PyPML.compute_parking_travel_time`.

The function `PyPML.get_free_places` returns the number of available spots from the given parking area. The result can be computed taking into account different options: uncertainty, projections, subscriptions, and capacity by vehicle type, if set.

The function `PyPML.get_closest_parking` performs a table lookup to return the list of the n reachable parking areas ordered by estimated travel time. The number of results returned is a parameter in the function. For performance reasons, the lookup table is updated using `PyPML.compute_parking_travel_time` and the user must call the function. In this way, different kind of estimation can be done (estimated travel time on an empty network or with a different network state). This table is computed using `simulation.findRoute`, so the weight of the edges can be manipulated using the `edge.setEffort` TraCI API.

Limitations: There are some limitations to keep into account while using PyPML. In order to obtain consistency, the library must acquire data every time-step of the simulation. Even if SUMO can handle multiple TraCI servers, we advise not to use them due to how the TraCI subscriptions are implemented. PyPML provides the functions `PyPML.get_traci_vehicle_subscription` and `PyPML.get_traci_simulation_subscription` in order to retrieve the last time step simulation data for the subscriptions used by the library. Moreover, `TraCI.simulationStep` cannot be used to fast-forward a simulation, or all the data during that period will not be processed by PyPML. These limitations are linked to the use of TraCI and cannot be overcome without completely changing the interaction with the simulator. Additionally, the concepts of capacity by vehicle type and subscription by vehicle type are abstractions provided by PyPML and SUMO is not aware of them. It implies that using `vehicle.rerouteParkingArea` it is possible to reroute more vehicle of a specific type to a parking area than the maximum capacity specified in PyPML without getting errors from SUMO. In this case, the solution is provided by using `PyPML.get_free_places` without uncertainty and enabling capacity and subscriptions flags.

In Section IV we present some examples of parking optimization and their comparison. In Section V we evaluate the performances of PyPML.

IV. Use-cases

We compare seven optimizations based on the same mobility scenario in order to showcase the capabilities of PyPML and to highlight the impact of features such as uncertainty and parking area location. We setup three base simulations using only SUMO capabilities, and then we performed two different optimizations, both with and without uncertainty.

A. MoST Scenario

The MoST Scenario [21] is a city-wide mobility scenario based on Principality of Monaco and the neighboring French cities. It provides a state-of-the-art 3D playground with various kind of vehicles, vulnerable road users (pedestrians and two-wheelers), and public transports. The latter is based on buses and trains, with more than 20 routes with over 150+ stops. The mobility is built using the activity-based mobility paradigm [25] and the 14 Traffic Assignment Zone (TAZ) provided by the scenario. The mobility represents a morning rush hour of an average weekday, with a population of 35K entities. The activity-based mobility generation uses multi-modal means of transports, and vehicles need to find a parking spot in order for the people to reach their destination. The scenario has 120+ parking area with a total capacity of 60K+ spots, this information is retrieved from OpenStreetMap (OSM) [26] and the Monaco parking website [27]. Figure 2 shows the location

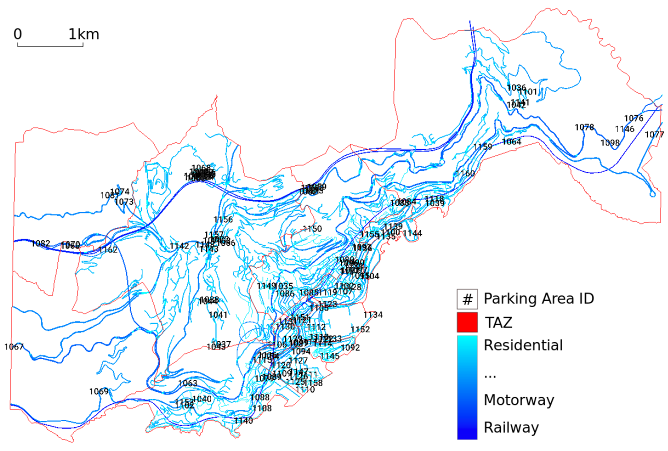


Fig. 2. MoST Scenario overview and parking locations.

of the parking areas, the TAZ in red, and the streets are colored by priority (light blue, lower priority; dark blue, higher priority).

B. SUMO: full, partial, and no visibility.

The first three optimizations are the one built-in in SUMO and achieved using static configuration files. The parking rerouters are positioned on every edge containing a parking area. One option is to use the rerouters to provide complete visibility on the availability of every other parking area, by using the `visible` tag set to true. In this case, each vehicle has the opportunity to reroute to a different parking area in case its destination is full. The vehicles use the default choice model configuration and drive to the closest free parking area. This solution can be associated with the presence of information signs scattered in the city. Another possibility is to remove the visibility in the rerouter completely. In this case, each vehicle has to reach the parking area to discover if it is full, and the choice of the next-best parking has to be taken without knowing the availability in the surrounding parking areas. Again, vehicles drive to the closest alternative. Finally, we created a middle ground more realistic and already used in the cities, where only parking areas with more than 1000 spots are marked as visible, allowing preemptive rerouting.

In these experiments, there is no uncertainty in the values used for the optimization.

C. PyPML: parking areas load balancing and driver-oriented optimizations

In order to implement these optimizations all the rerouters must be removed from the simulation; otherwise SUMO has the final word on the optimization. We implemented two different optimizations that would represent different points of view, the driver that requires parking very close to the destination, and the parking areas administrator that wants to optimize the use of all the resources. Both these optimizations can be implemented using the same set of APIs provided by the framework, focusing on the logic behind them, and leaving the data aggregation to PyPML.

1) Parking areas load balancing: In one experiment the optimization is based on the centralized load balancing of the parking areas. In this case, every minute all the parking areas are checked to evaluate if load balancing is required. The availability is computed using `PyPML.get_free_places` and taking into account the predictions (number of vehicles that expressed the intent of using the parking area). If the load balancing is required, a selection of vehicles is rerouted to an alternative parking area, close to the previously selected one, with more availability. The alternative options are gathered using the function `PyPML.get_closest_parkings`; after their evaluation, the vehicle's plan is updated using `TraCI.rerouteParkingArea`.

In one version of the experiment, there is no uncertainty, so the values obtained are always correct. In the other experiment, the function `PyPML.get_free_places` is called with the uncertainty parameter set, where the error is a normal distribution with $\mu = 0$ and $\sigma = capacity * 0.2$.

2) Proactive vehicle optimization: The last variation is based on a selfish decision taken by each vehicle. Every minute the vehicle asks for the current availability of the chosen parking, taking into account the predictions. In case the parking has less than ten spots available, the vehicle looks for an alternative, it evaluates the alternatives, and finally, it updates the plan. The functions used in this variant are the same used by the parking areas load balancing, but in this case, the decision is decentralized and driver-oriented.

In one experiment the values are always correct, and in the other, the normal distribution for the uncertainty used by the previous optimization is applied.

D. Evaluation

We compared the different experiment using the following metrics: parking occupancy over time, number of destination changes, and total travel time for both vehicle and person. In all the figures: *Full Visibility*, *Partial Visibility*, and *No Visibility* are the experiment described in Section IV-B; *Proactive Veh.* and *Proactive Veh. with Unc.* are the experiment described in Section IV-C2, proactive vehicles with and without uncertainty; *Proactive P.M.* and *Proactive P.M. with Unc.* are the experiment described in Section IV-C1, the centralized load balancing of the parking management with and without uncertainty.

1) Occupancy over time.: Although the optimization uses the complete MoST Scenario with 127 parking areas, we selected only a few of them to showcase the different behavior that can be seen. Figure 3 presents the occupancy over time of four parking areas in Monaco, all with the same capacity (250 spots), and positioned in strategic locations. We selected them to show how the same optimization algorithm performs differently in different locations. The parking areas are located as follow:

- "Centre" is located in the middle of Monaco city.
- "Centre-Riviera" is close to "Centre", but is on the shoreline, close to the sea.

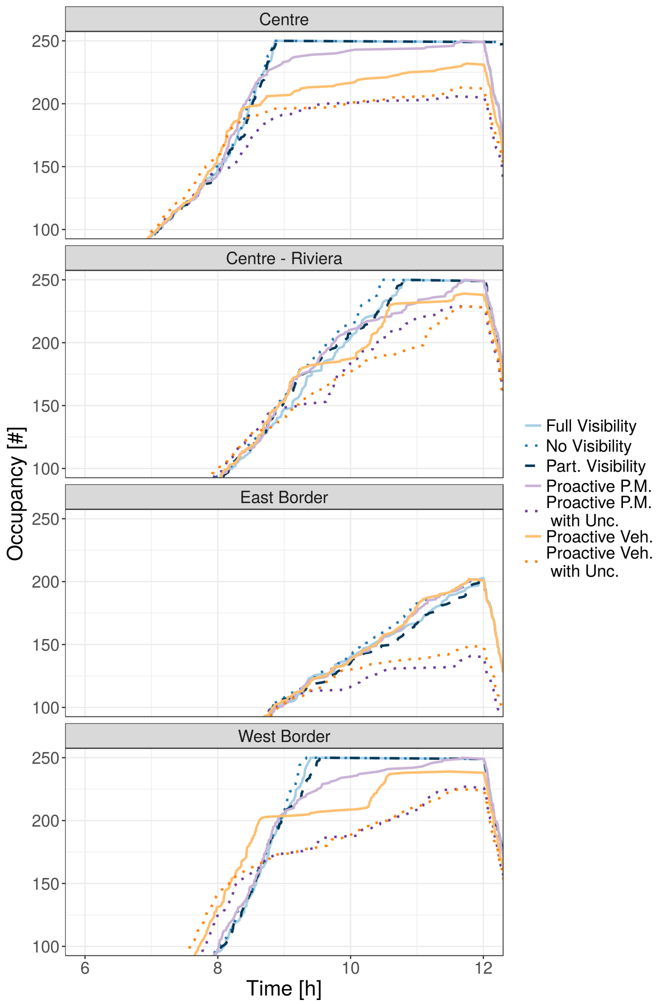
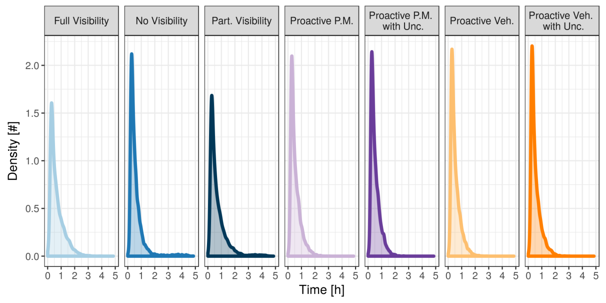
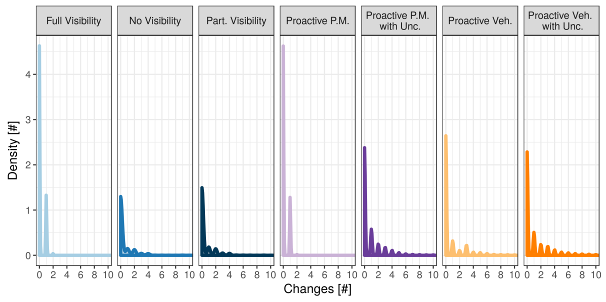


Fig. 3. Parking occupancy over time.



(a) Travel Time.



(b) Number of Parking Changes.

Fig. 4. Distributions.

- "East Border" is located in the city but close to the east border with France; in this location, we find one of the main gateways (major roads) to access the city.
- "West Border" is still located in the city but on the western border with France; close to this location, we have three main gateways to Monaco.

It is important to keep into consideration that, given the limited size of the city, all four parking areas are located in an area of three square kilometers.

Analyzing the four graphs we can see that the only parking area that never reaches full capacity is the "East Border". It is mainly due to reachability issues, it is further away from the center compared to others parking areas, and once the vehicles are already inside the city, reaching it in congested traffic is an issue.

Focusing on a single graph and then comparing the behavior with the others, we see that the three SUMO optimizations tend to behave similarly, reaching full capacity almost at the same time. On the contrary, we can notice two different behaviors in the load balancing with and without uncertainty, showing that the presence of errors in the data makes harder to achieve a consistent load balancing.

Additionally, the presence of uncertainty in the information makes two different behaviors, the selfish vehicle optimization, and the centralized load balancing, produce similar results.

2) Total travel time and destination changes.: Figure 4 shows the distribution of the total travel time (4a) and the distribution of the number of the parking destination changes (4b). Regarding travel time, the SUMO experiment without visibility and the four experiments with the optimization assisted by PyPML look very similar, but the same experiments evaluated in terms of rerouting showcase a very different behavior. Aside from the centralized load balancing without uncertainties, all the other experiments require many changes in the parking destination, showing instability in the optimization. This unstable behavior is better outlined in Figure 5, where the Empirical Cumulative Distribution Function (ECDF) of the number of parking destination changes is plotted for all the experiments. The graph shows the similarity between full visibility and the centralized load balancing without uncertainty, and at the same time, how impactful is the presence of uncertainty in the centralized optimization.

Finally, Figure 6 show the mean number of parking destination changes (6a) and mean total travel time (6b). In these simplified graphs, we can see how uncertainty has different impacts on different optimizations. In Figure 6a we see that uncertainty increases the number of rerouting in case of centralized optimization, but decreases them for selfish vehicle optimization. Additionally, the impact of uncertainty on the median travel time for both vehicles and people is reversed compared to the number of destination changes.

On another note, we can see that for both proactive vehicles

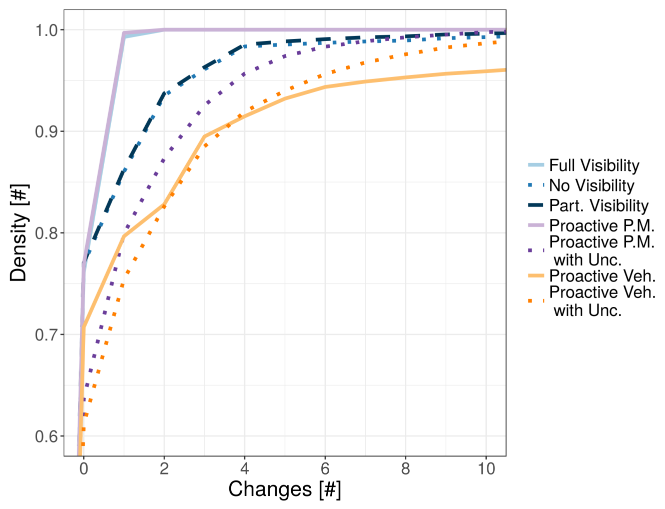


Fig. 5. Cumulative distribution function of the number of parking changes.

optimizations the number of outliers (vehicles with more than ten destination changes) is higher than all the other optimizations (Figure 5). Nonetheless, the selfish optimization manages to achieve low mean travel times compared to others (Figure 6b).

Remarks

The use-cases we presented show how PyPML allows easy implementation and comparison of different large-scale parking managers, and how location and uncertainty plays an essential role in the optimizations. Due to the large-scale nature of these optimizations, simulation performances present a significant concern.

V. Performance Evaluation

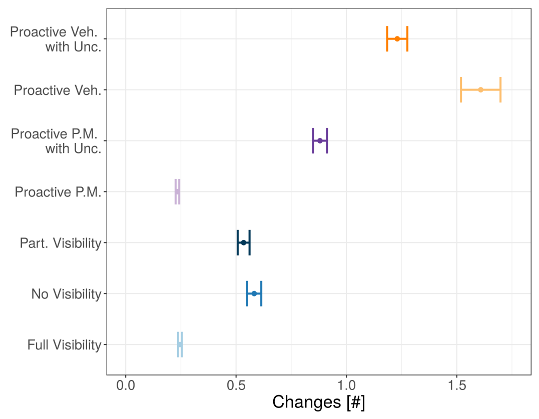
It is well known that the use of TraCI presents a bottleneck in the simulation. We want to be sure that the use of PyPML to monitor the parking areas and the simple use of TraCI to retrieve the parameters we used in the evaluation is linear.

We run all the simulation using a computer with an SSD hard drive, 16G Synchronous DDR3 RAM with speed 1.600 MT/s, and an Intel® Core^{TM} i7-6500U CPU at 2.50GHz and 4.096 KB cache.

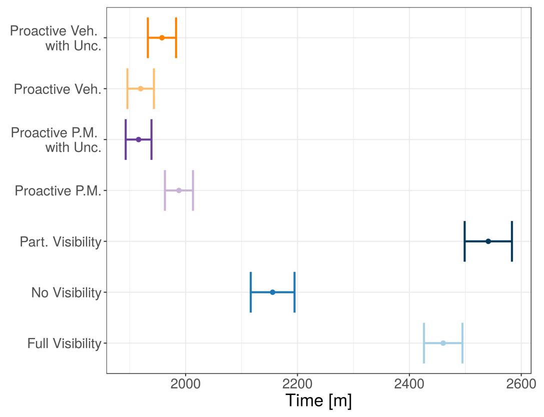
The metrics are the one provided by SUMO under the performances section in the statistics [28]:

- Duration: the amount of elapsed time (ms) while computing the simulation.
- Updates Per Second (UPS): the number of simulated vehicles on average per second of computation time.

Figure 7 shows duration and UPS comparison between the base experiment without data collection and without TraCI, and the seven use-cases presented. The base simulation without TraCI is the fastest and more efficient. The three SUMO simulation that uses TraCI only to retrieve the data used for the evaluation, see the duration doubled and the UPS halved. The four optimizations that use PyPML present a comparable level of deterioration with the SUMO experiments, showing that the



(a) Mean number of parking changes. The bars represents the 95% percentile.



(b) Mean travel time. The bars represents the 95% percentile.

Fig. 6. Parking Destination Changes and Travel Time.

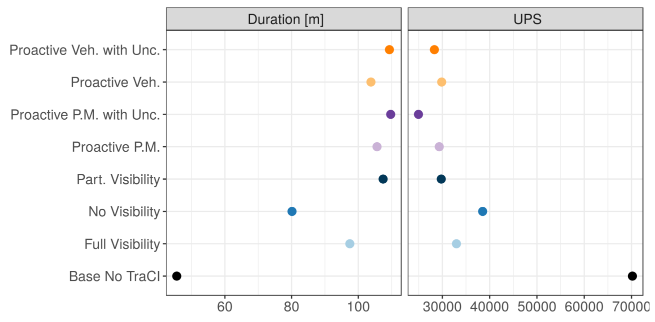


Fig. 7. Performances

deterioration scales with the complexity of the optimization, and not with the parking monitoring in itself.

VI. Conclusion and Future Work

In this paper, we presented the general-purpose Python Parking Monitoring Library (PyPML), a tool integrated with the Simulation of Urban MObiltiy (SUMO) framework that uses Traffic Control Interface (TraCI) to gather and aggregate the parking monitoring information. PyPML presents an abstraction layer that can be interrogated to optimize and take decisions on large-scale parking management. In order to actuate the optimization, we implemented additional TraCI APIs that are already integrated in SUMO v1.0.0.

We implemented, evaluated and compared seven optimizations in order to highlight the necessity of large-scale parking management optimizations and to showcase PyPML capabilities. Three optimizations were made using only the built-in functionality of SUMO. Two different kinds of optimization (centralized load balancing and selfish vehicle decision) were performed both with and without uncertainty using PyPML. The evaluation shows the necessity of modeling uncertainty and the impact of the parking locations on the optimizations. Finally, we evaluated PyPML performances compared to the base simulation without TraCI and the SUMO simulations without PyPML, showing that the degradation in performances for the monitoring is only linked to the use of TraCI, and not the abstraction.

In the future, we plan to use this framework to work on parking areas management used as multi-modal hubs, evaluating different optimizations focused on the trade-off between best parking usage, and user-centered trip plans.

Additionally, as we showed how uncertainty plays a significant role in the optimizations, we are going to add the possibility of having uncertainty on a per-vehicle basis, and not only linked to the parking areas.

PyPML is freely available on GitHub under GPLv3 license: `https://github.com/lcodeca/pypml`.

Acknowledgment

This work was partially funded by the French Government (National Research Agency, ANR) through the "Investments for the Future", ref. #ANR-11-LABX-0031-01. EURECOM acknowledges the support of its industrial members, namely BMW Group, IABG, Monaco Telecom, Orange, SAP, ST Microelectronics and Symantec.

References

[1] D. King, "Estimating environmental and congestion effects from cruising for parking," in *Transportation Research Board 89th Annual Meeting*, Washington DC, United States, 2010.

[2] Beaudoin, Justin and Farzin, Y Hossein and Lawell, C-Y Cynthia Lin, "Public transit investment and sustainable transportation: A review of studies of transit's impact on traffic congestion and air quality," *Research in Transportation Economics*, vol. 52, pp. 15–22, 2015.

[3] Idris, MYI and Leng, YY and Tamil, EM and Noor, NM and Razak, Z, " park system: a review of smart parking system and its technology," *Information Technology Journal*, vol. 8, no. 2, pp. 101–113, 2009.

[4] Karbab, ElMouatezbillah and Djenouri, Djamel and Boulkaboul, Sahar and Bagula, Antoine, "Car park management with networked wireless sensors and active RFID," in *Electro/Information Technology (EIT), 2015 IEEE International Conference on*. IEEE, 2015, pp. 373–378.

[5] Lei, Chao and Ouyang, Yanfeng, "Dynamic pricing and reservation for intelligent urban parking management," *Transportation Research Part C: Emerging Technologies*, vol. 77, pp. 226–244, 2017.

[6] Gupte, Sanket and Younis, Mohamed, "Participatory-sensing-enabled efficient Parking Management in modern cities," in *Local Computer Networks (LCN), 2015 IEEE 40th Conference on*. IEEE, 2015, pp. 241–244.

[7] B. Hoh and T. Yan and D. Ganesan and K. Tracton and T. Iwuchukwu and J. Lee, "TruCentive: A game-theoretic incentive platform for trustworthy mobile crowd-sourcing parking services," in *15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2012, pp. 160–166.

[8] Timpner, Julian and Schürmann, Dominik and Wolf, Lars, "Trustworthy parking communities: helping your neighbor to find a space," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 1, pp. 120–132, 2016.

[9] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent Development and Applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, no. 3&4, pp. 128–138, December 2012.

[10] L. Codeca and J. Härri, "Towards multimodal mobility simulation of C-ITS: The Monaco SUMO traffic scenario," in *VNC 2017, IEEE Vehicular Networking Conference, November 27-29, 2017, Torino, Italy*, Torino, ITALY, 11 2017.

[11] A. Wegener, M. Piórkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux, "TraCI: an interface for coupling road traffic and network simulators," *CNS 08 Proc. 11th Commun. Netw. Simul. Symp.*, pp. 155–163, 2008.

[12] Hassoune, Khaoula and Dachry, Wafaa and Moutaouakkil, Fouad and Medromi, Hicham, "Smart parking systems: A survey," in *Intelligent Systems: Theories and Applications (SITA), 2016 11th International Conference on*. IEEE, 2016, pp. 1–6.

[13] A. O. Kotb and Y. Shen and Y. Huang, "Smart Parking Guidance, Monitoring and Reservations: A Review," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 2, pp. 6–16, Summer 2017.

[14] Djahel, Soufiene and Doolan, Ronan and Muntean, Gabriel-Miro and Murphy, John, "A communications-oriented perspective on traffic management systems for smart cities: Challenges and innovative approaches," *IEEE Communications Surveys & Tutorials*, vol. 17, 2015.

[15] V. W. s. Tang and Y. Zheng and J. Cao, "An Intelligent Car Park Management System based on Wireless Sensor Networks," in *2006 First International Symposium on Pervasive Computing and Applications*, Aug 2006, pp. 65–70.

[16] R. E. Barone and T. Giuffre and S. M. Siniscalchi and M. A. Morgano and G. Tesoriere, "Architecture for parking management in smart cities," *IET Intelligent Transport Systems*, vol. 8, no. 5, pp. 445–452, August 2014.

[17] T. Rajabioun and P. A. Ioannou, "On-Street and Off-Street Parking Availability Prediction Using Multivariate Spatiotemporal Models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2913–2924, Oct 2015.

[18] A. O. Kotb and Y. Shen and X. Zhu and Y. Huang, "iParker – A New Smart Car-Parking System Based on Dynamic Resource Allocation and Pricing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 9, pp. 2637–2647, Sept 2016.

[19] I. Aydin and M. Karakose and E. Karakose, "A navigation and reservation based smart parking platform using genetic optimization for smart cities," in *2017 5th International Istanbul Smart Grid and Cities Congress and Fair (ICSG)*, April 2017, pp. 120–124.

[20] Shuo-Yan Chou and Shih-Wei Lin and Chien-Chang Li, "Dynamic parking negotiation and guidance using an agent-based platform," *Expert Systems with Applications*, vol. 35, no. 3, pp. 805 – 817, 2008.

[21] L. Codeca and J. Härri, "Monaco SUMO Traffic (MoST) Scenario: A 3D Mobility Scenario for Cooperative ITS," in *SUMO 2018, SUMO User Conference, Simulating Autonomous and Intermodal Transport Systems*, Berlin, GERMANY, 05 2018.

[22] "SUMO Parking Areas," http://sumo.dlr.de/wiki/Simulation/ParkingArea, (Last access: September 2018).

[23] "SUMO Rerouting to alternative Parking Areas," http://sumo.dlr.de/wiki/Simulation/Rerouter#Rerouting_to_an_alternative_Parking_Area, (Last access: September 2018).

[24] "SUMO TraCI Subscriptions," http://sumo.dlr.de/wiki/TraCI/Object_Variable_Subscription, (Last access: September 2018).

[25] M. Balmer, K. Axhausen, and K. Nagel, "Agent-based demand-modeling framework for large-scale microsimulations," *Transportation Research Record: Journal of the Transp. Res. Board*, no. 1985, pp. 125–134, 2006.

[26] M. Haklay, "How good is volunteered geographical information? A comparative study of OpenStreetMap and Ordnance Survey datasets," *Environment and planning. B, Planning & design*, vol. 37, no. 4, p. 682, 2010.

[27] "Monaco Parking Website," https://monaco-parking.mc, (Last access: September 2018).

[28] "SUMO Simulation Output," http://sumo.dlr.de/wiki/Simulation/Output, (Last access: September 2018).