TOWARDS A COMBINATION OF DISCRETE-EVENT SIMULATION WITH MACHINE LEARNING FOR SMART-PARKING

Antoine Dominici, Laurent Capocchi, Emmanuelle De Gentili, Jean-Francois Santucci
SPE UMR CNRS 6134, University of Corsica
Campus Grimaldi, 20250 Corte (France)
{dominici_a,capocchi,gentili,santucci}@univ-corse.fr

ABSTRACT

In several cities, finding a parking spot is often a source of frustration for many drivers. If we ignore the obvious solution to add more parking slots, a less expensive and easier solution could then be to optimize the current parking slots by visibility, management and policies. This is how Smart-parking projects has become popular, it has allowed to upgrade parking productivity and effectiveness without adding parking slots. This paper propose to combine the discrete-event simulation provided by the discrete-event system specification formalism with a supervised learning algorithm in order to classify parking slots depending on their occupation time in order to predict departures. In comparison, the city Bastia (France) currently accommodates more than 450 sensors. These have been used to test the proposed approach from the case above.

Keywords: Discrete-event, Modeling, Simulation, Smart-Parking, Neural network.

1 INTRODUCTION

Nowadays, the management of the parking slots occupancy in a city is a major issue in regulating traffic cars. Mathematical models based on supervised artificial intelligence algorithms have been developed in order to predict phenomena of traffic accumulation at peak hours according to the occupancy rates of parking slots accumulated over a long period (Lin, Rivano, and Le Mouël 2017, Mendoza-Silva, Gould, Montoliu, Torres-Sospedra, and Huerta 2019, Lin 2015). It supervised algorithm therefore require large training and testing dataset and it is based on hyper-parameters that are difficult to configure. In most cases, its dataset are not accessible and it is necessary to simulate them in order to exploit the supervised learning algorithms. In fact, there is some Parking datasets as in CNRPark project However all these projects have different goals and so there datasets are specifically made for that. For example CNRPark made a datasets mainly for visual recognition of car assisted by machine learning but there timing data are refresh every 5 minutes. So, to counter this leak of data, we decide to simulate our data reflecting humans behavior.

On the other hand, discrete-event modeling and simulation (M&S) is an area that deserves special attention in the field of artificial intelligence (Wallis and Paich 2017, Feng, Chen, and Lu 2018). Indeed, the combination of discrete events M&S with prediction or classification algorithms would allow for example: to provide a M&S framework dedicated to the generation of learning data and to test artificial intelligence algorithms devoid of real data or (ii) to consider more precisely early events (responsible for the parking slots congestion) and to give an alternative action policy (to the driver who wants to park).

This paper proposes to combine the discrete-event simulation provided by the discrete-event system specification (DEVS) (Zeigler and Sarjoughian 2013) formalism with a supervised learning algorithm in order to predict the occupation time of a parking slot. This combination uses the simulation to randomly generate the input learning dataset following the Poisson law and helps the modeler to develop classification model without having to wait real data acquisition.

The methodology consists in developing data generation DEVS models based on the discrete Poisson distribution which can help to describes the behavior of the number of parking occupancy slot events occurring in a fixed time interval is used as a simple model $(Pham, Tsai, Nguyen, Dow, and Deng 2015)$. The generated events are used for learning and testing a neural network DEVS model, which aims to classify parking slots according to their availability period. Finally, the prediction of the occupancy periods of parking places from the neural network will be used in a mobile application allowing users to view the occupancy of parking places as well as their predicted period of occupation.

The advantage of DEVS lies in the possibility of modeling and exploiting discrete-event simulation to generate learning data (from a Poisson discrete law) which is used by the neural network. In addition, DEVS allows the simulation to be combined with the inline learning loop of the neural network. Indeed, thanks to the DEVS simulation, the evolution of the parking slots occupancy is integrated into the learning of the neural networks which can be done inline. In this kind of system, inline learning is very useful. In fact the behavior of users can be modified by external events such as: change in parking policy, weather, public tourism event, etc. In this context, an inline neural network can be very useful.

The paper is organized as follows: Section 2 gives some background about the DEVS formalism and the DEVSimPy environment which is used to model and simulate the model of Smart-parking. A subsection presents a related work about the machine learning algorithms applied in the context of Smart-parking. Section 3 presents the proposed approach based on the use simulated input data to a neural network as machine learning technique to predict the occupation times of parking slots. Finally, a conclusion is given and some perspectives are envisioned.

2 BACKGROUND

2.1 Discrete-Event system Specification Formalism and DEVSimPy Environment

The Discrete-Event System Specification (DEVS) formalism was introduced by Zeigler in the seventies (Zeigler 1976, Zeigler and Sarjoughian 2013) for modeling discrete-event systems in a hierarchical and modular way. DEVS formalizes what a model is, what it must contain, and what it doesn't contain (experimentation and simulation control parameters are not contained in the model). Moreover, DEVS is universal and unique for discrete-event system models. Any system that accepts events as inputs over time and generates events as outputs over time is equivalent to a DEVS. With DEVS, a model of a large system can be decomposed into smaller component models with couplings between them. DEVS formalism defines two kinds of models: (i) atomic models that represent the basic models providing specifications for the dynamics of a sub-system using function transitions; (ii) coupled models that describe how to couple several component models (which can be atomic or coupled models) together to form a new model. An atomic DEVS model can be considered as an automaton (like Mealy FSM) with a set of states and transition functions allowing the state change when an event occur or not. When no externals events occur, the state of the atomic model can be changed by an internal transition function. When an external event occurs, the atomic model can intercept it and change its state by applying an external transition function. The life time of a state is determined by a time advance function called ta. Each state change can produce output message via an output function called lambda. A simulator is associated with the DEVS formalism in order to exercise instructions

of coupled model to actually generate its behavior. The architecture of a DEVS simulation system is derived from the abstract simulator concepts associated with the hierarchical and modular DEVS formalism.

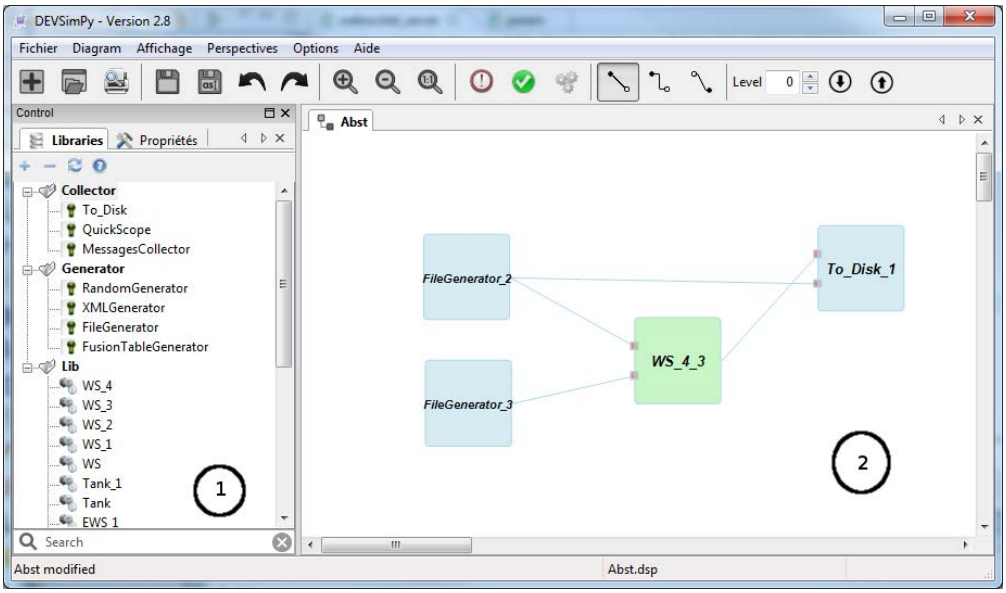

Figure 1: DEVSimPy general user interface.

DEVSimPy (Python Simulator for DEVS models) (Capocchi, Santucci, Poggi, and Nicolai 2011, group) is a user-friendly interface (Figure 1) for collaborative M&S of DEVS systems implemented in the Python language. DEVSimPy is an open source project under GPL V3 license and its development is supported by the University of Corsica "Pasquale Paoli" Computer Science research group. The DEVSimPy project uses the Python programming language for providing a GUI (based on wxPython (Rappin and Dunn 2006) graphic library which is a wrapper of the most popular WxWidgets C library) for the PyDEVS (Bolduc and Vangheluwe 2001) and PyPDEVS (Tendeloo 2014, Van Tendeloo and Vangheluwe 2014, Van Mierlo, Mustafiz, Barocca, Van Tendeloo, and Vangheluwe 2015) which is the the Parallel DEVS implementation of PyDEVS APIs. DEVSimPy has been set up to facilitate both the coupling (part 2 in Figure 1) and the re-usability of the PyDEVS classic DEVS models and the PyPDEVS Python Parallel DEVS models which are stored inside repositories (part 1 in Figure 1).

In this paper, DEVS has been used in order to modeling and simulate sensors network that generate parking availability information in order to help drivers find more efficiently desired parking slots using a neural network based classification.

2.2 Machine Learning for Smart-Parking Solution

In the context of Smart-parking, some Markov chain based algorithms are used to predict the departure of a car from a parking slot by using Poisson distribution process (Lin, Rivano, and Le Mouël 2017). They are used to give drivers a probability of availability for one or multiple parking slots at the arrived time as shown in (Pullola, Atrey, and El Saddik 2007). Users can find the parking slot with the highest chance of being able to park their car. With this solution, there is no need of a big dataset and that's why it is the most common used method in Smart-parking solutions. However, a bigger dataset allows to use more complex algorithms as a neural network, support vector regression and regression tree (Zheng, Rajasegarar, and Leckie 2015). However giving occupancy probabilities to parking slots is not the only way to provide smart-services to a city. In (Demisch 2016), the authors show that they can give services not only to users

but also to municipalities. As shown, they made a dynamic pricing system on parking slot which is related to demand, this may therefore be a solution for many cities to split up vehicle streams and increase park slot availability rates in high demand zones. Moreover they also made an occupancy prediction on park slots. So in this kind of system, both users and operators find their interest. There are 23 Smart-parking projects (*Lin*, *Rivano*, *and Le Moul*2017)created since 1996 and they all have different specification as the data acquisition model, but also services that they provide.

In this paper, we have a known data model, and we want to make predictions, which is well suited for supervised learning. Indeed we want to predict the departure date of a user of a parking slot based on the data model listed above. The proposed approach will be focused on a supervised learning method. Nevertheless, there are a lot of different methods for supervised models (*Bishop*2006). Due to the use of simulation, we have a large dataset that are used to make parking slot departure predictions based on neural network.

3 THE PROPOSED APPROACH

Figure 2 depicts the proposed approach involving three processes: (i) DEVS model has been used to simulate sensor data information with a pre-processing phase (ii) a NN model has been used to predict departure time for each sensor (iii) a mobile application has been implemented to display remotely the available slots and the simulated state of them.

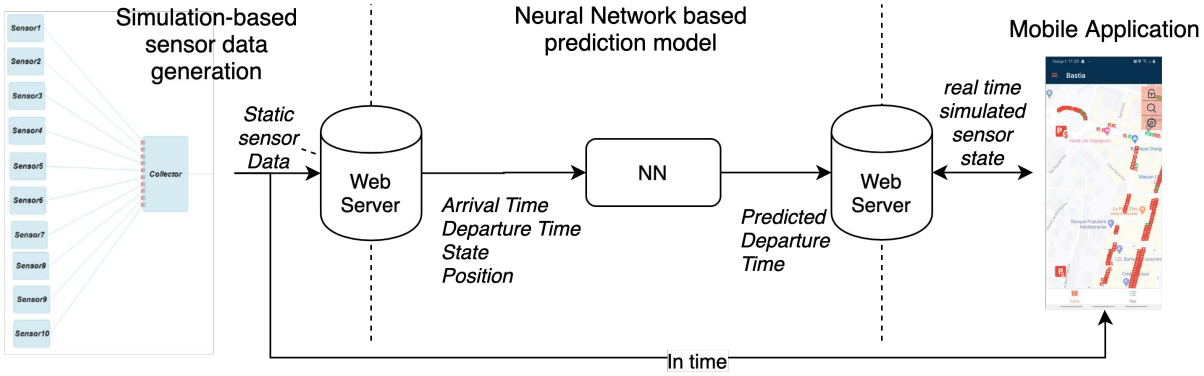

Figure 2: Proposed approach based on a pre-processing phase (Simulation based sensor data acquisition), a major process based on supervised algorithm (NN based classification) dedicated to predict the departure time of parking slot displayed in the end process through a mobile app.

3.1 DEVSimPy Modeling

The discrete-event simulation has been used in order to make a machine learning model based on simulated input pattern. Due to its modular hierarchical aspect, DEVS is a very interesting tool for creating data in this project. It allows both to apply genericity but also to simplify this process. In addition, composition in atomic model with the possibility in DEVSimPy of being able to change the input parameters of each model makes the test phase much more accessible. It also allows us to be able, in real-time mode, to simulate the evolution of car park states in order to be able to develop and test the mobile application as we can see on the figure 2. A Poisson based random generator DEVS model has been implemented to return a random occupation state for each of the sensors according to a Poisson distribution.

Poisson's rare events law is used to calculate the probability of a parking slot sensor occupying time. The problem can be formulated as follows: During a period of parking activity, a parking slot is occupied during

a period m (expressed in minutes for example) on average. Poisson's law makes it possible to calculate the probability that the place is occupied during 1,2,3, ..., k period in a given time interval.

The probability is given by the following formula (from (Ahrens and Dieter 1982)):

$$P(k,m) = e^{-m}.\frac{m^k}{k!}, \tag{1}$$

where m is at the same time the mean and the variance of the law. We use the inversion method to efficiently generate Poisson distributed events (Willmot 1987).

The formula 1 has been used by the generator DEVS model according to an algorithm that can be resumed as follows: (i) we randomly draw a decimal number between 0 and 1 and consider it as a cumulative probability (the probability that there are 0,1,2 or k events in a Poisson distribution of parameter m) (ii) thanks to the calculation of the cumulative probability of Poisson's law, we seek to which random variable k this corresponds.

The DEVSimPy environment based on the DEVS formalism has been used to implement the proposed approach that includes a random generation value process based on the Poisson law. First, an atomic DEVS model (Sensor1 in Figure 3 for example) has been implemented to generate events with a life time corresponding to the occupancy time of a parking space according to the Poisson distribution. The atomic DEVS model Collector has been implemented to collect all events and to classify them in order to prepare the input data set for the NN model (pre-processing phase). The model WebServer exposes the occupancy prediction information (web service) to the mobile application.

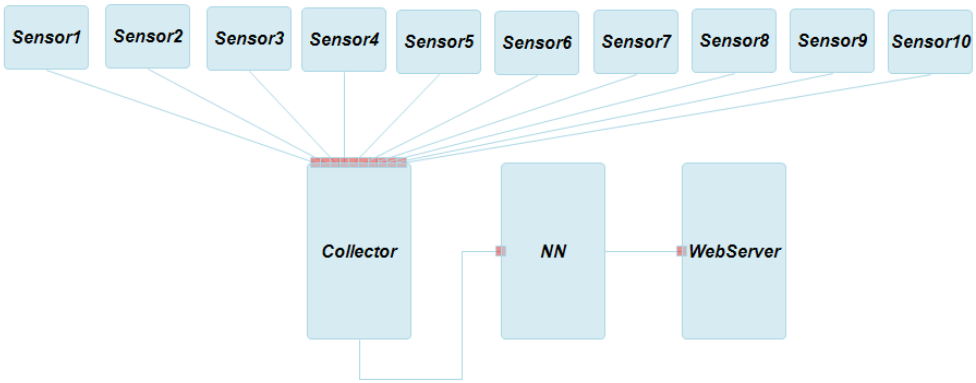

Figure 3: DEVSimPy simulation model used to simulate the occupancy of 10 sensors in a Smart-parking. The model Collector aggregates all of its input events to classify them in pre-processing of data for the neural network.

For being able to predict a vehicle departure from a parking slot we first had to choose the corresponding methods according to the datasets and the inline learning possibilities. As we said in section 2.2, NN has been used as a supervised learning algorithm. The NN model has been developed using the Keras (Chollet et al. 2015) library which was designed to provide a simplistic interface for quickly creating prototypes by building neural networks that can work with TensorFlow (Abadi, Barham, Chen, Chen, Davis, Dean, Devin, Ghemawat, Irving, Isard, and et al. 2016). In addition, it is developed on Python language, and therefore is more easily compatible with DEVSimPy. The DEVSimPy atomic model NN in Figure 3 embed this Keras-based implementation.

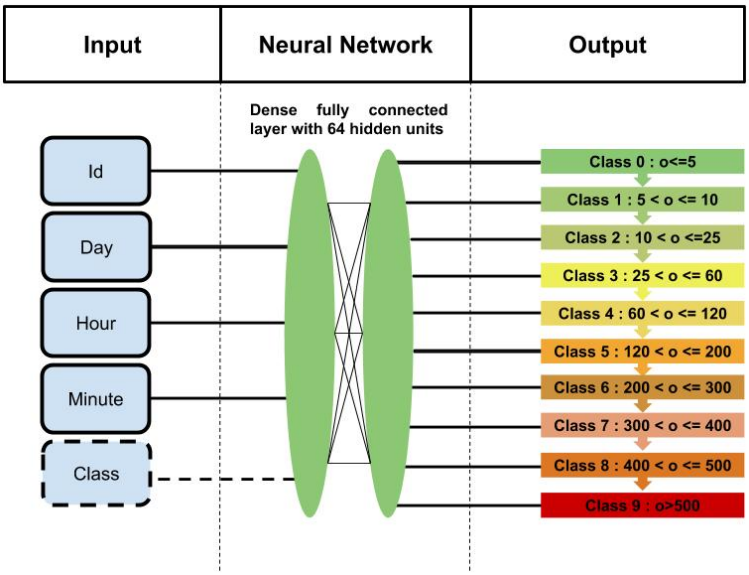

Figure 4: The proposed Neural Network (NN model in Figure 3) architecture with time information as inputs and membership classes as outputs.

The classification of the occupancy (outputs of the NN model in Figure 4) is defined in Table 1. These classes were fixed with a widening gap but that they will in the future be fixed to the different activity exercised by the residents in order to avoid class crossing and therefore to lose precision.

class	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
occ.	<= 5	]5,10]	]10,25]	]25,60]	]60,120]	]120,200]	]200,300]	]300,400]	]400,500]	>500

Table 1: Duration classes used by the neural network. For example, if an occupancy (occ.) is between 10 and 25 minutes, the sensor belongs to the class c_2.

Finally, the DEVSimPy atomic model WebServer allows to expose the simulation results as web services. In this way, membership of occupancy classes can be exploited to be displayed on a map for example.

3.2 DEVSimPy Simulation Results and Analysis

The simulation has been performed with the following software/hardware characteristics: Windows 10; Python 3.7.6; 6 CPU (2.2 GHz Intel Core i7) and 32GB of RAM.

The parameters of the NN model (figure 4) are $momentum = 0.9$, $weightdecay = 1e-6$, $learningrate = 0.1$. The parameters of the Poisson law used in the 10 Sensor models are different for each simulated sensor. Respectively from ID 0 to 9 they are: $m = \{5, 20, 45, 68, 70, 150, 270, 330, 450, 600\}$. These numbers are randomly generated expect for two who need to be close enough to have the same result. After the simulation both sensors will generally be in the same duration class (Table 1).

This first modeling approach implies that we do not have an equal number of examples from each class and the dataset is an Imbalanced Classification Dataset (Shukla and Bhowmick 2017). Nevertheless, accuracy metric must be equally important in both classes. Additionally, to improve the approach and to make dataset more balanced when fitting the supervised learning algorithm, we setup a NN model taking in input the following set (see figure 4): {id: (sensor ID), day: [1-7], hour: [0-24], minute: [0-60], duration class: [c_0-c_9]} for training and giving the duration class (Table 1) of the parking slot. The input data are stored in

CSV and imported into the NN for the training and testing phase. We separate the input (resp. output) as a variable X (resp. Y). The latter will be binary encoded in a certain number of classes (here 10). For example, class 3 will therefore give in binaries [0,0,1,0,0,0,0,0,0,0]. Then we divide the training data from the test data. We choose to keep 75 percent of the data for the training dataset and 25 percent for the testing dataset. This ratio has been choose after trying different combination and it seems the most optimize ratio for the machine learning $(Bishop1995)$. This separation is done randomly with a seed. The data is finally ready to be used in the neural network. Before the training we already knew that the accuracy on the NN will be really low because of the data. The data had no correlation between them because of the simple Poisson distribution based simulation. So the NN will have some difficulties to predict the departure time if it can't base his prediction on some connectedness. Because of all these negative point the accuracy with these data was only 36%.

In fact the data needs to be link between them in a way to get more correlation. So the system need to be integrate in constraint system which will reflect users behaviour with parking. For making the data more consistent with more correlation, we set up some rules in the generation of park slot occupancy duration according to the non-homogeneous Poisson Processes (Ross 2010). Without any numeric data taken from the real parking slots we had to make a constraint simulation based on the experience. To achieve that, we created users categories:

- The worker:
  - which park is car the morning, eat at work and come back at home the night
  - which park is car the morning, go back at home for lunch and return to work before leaving back home.
- The customer:
  - which randomly come each hour of the day to make some grocery, or buy something.
  - which will go at restaurant mostly at lunch and dinner
  - which will take a drink mostly at night
- The delivery or intervention man which come at each hour of the day to make a delivery.

To make that system, we made a new atomic model named "ConstraintPoissonSensor" and in each internal transition, following the Poisson law, we simulate different case as the worker, the customer and the delivery man. They all got different impact on the time the simulate user will stay on the park slot depending on what type it is. So we follow the Poisson law but we put some condition in a way to make data more realistic. That's why we called it, the constraint Poisson sensor law.

The simulated sensor, for each cycle that it will make, will execute one of these cases with a certain percent chance of happening following the Poisson law. This chance is based on the visual ascertainment and we tried to be as close as we can see from the reality cases.

We didn't set up any cycle limit on the simulation and we activate the no time limit option in DEVSimPy. We choose to use only 10 atomic models, but this number can be modified without changing the process. Moreover, here, we just want to obtain some data who fits to the general behavior of parking slots. Once the data simulated, we made graph representing the turn over of these 10 slots depending one the hour.

We had more realistic data as shown on the figure 5. It shows the number of arrival (the count of events) happening for each hour. We did not make a scale of days because the simulation doesn't contain the rule for each day of the week. The figure 5 depicts the fluctuation peaks for each realistic arrival time slot. So we can say that the data reflects with the average drivers behaviour.

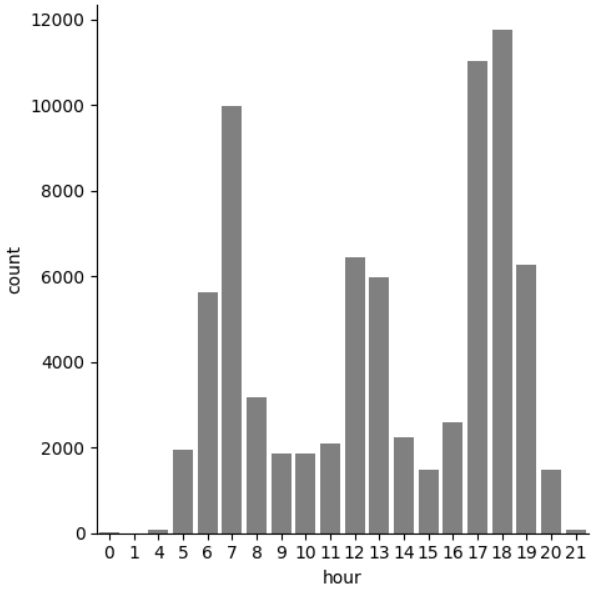

Figure 5: Park slot arrival count with constraint simulated sensor following the users categories based on non-homogeneous Poisson process.

With this more balanced dataset, we set up the NN model (figure 4) and we train it to obtain an accuracy of 49 percent which is still too low for the future usage considering the desired tolerance. Our goal is for our users to have less than 3 minute waiting time, thus the results are too low. For example, we can improve these results by considering some additional constraints such as:

- the reservation time: billing time will be an huge facilitator prediction for NN because of the impact that it had on the departure time
- the sensor location: as real impact on the users interest for a certain park slot. It will help also to know the average behaviour of user for a know place(grocery, near house park slot, etc..).
- daily behaviour consideration: the dataset will need to be also scaled on day. With some rules apply to the simulation base of current day of the week, we can follow in a more realistic way the user behaviour depending on the day. It allows the neural network to understand the link between the day and users conduct.

In this paper, only the last constraint has been considered. Basically, the parking's slot are empty during the weekend except for special events which we will not count here. As a result we have set up a duration generation rule for free park slot, during weekends we increase the slot availability rate.

For all week except for the weekend as well as Friday, users have in general the same behaviour. So there is no need to make different rules for these days. But the users have different behaviours during the weekend depending on if they live inside the city or if they live outside. We have decided to make some constraint for these two different types of users:

- downtown resident:
  - Sedentary: citizen that stay all the weekend in the city.

  - Active: citizen who make some outside city activities during the weekend. They can go back into the city at any day of the weekend depending of their activities.
- outside resident:
  - Customer: User that will come into the city to make different activities (shopping, grocery, coffee, walks, restaurant). There is different activities and they all had a custom generation time based Poisson Distribution law.

After integrating these constraints to the simulation, we simulate and train the NN with this new dataset and 60.61 percent has been obtained which is slightly better than the previous model and can be explain by the more complex correlation between data.

The add of sensor location constraint in the simulation has been ignored (because of the realistic data leak that we are missing for the moment) and the park slot billing time has been considered. We supposed that when a driver arrive on a parking slot, the driver parks and goes and buy a ticket for a certain amount of time. But in case they don't respect the billing park duration, an infraction rate of 20 percent is raised. The accuracy of the NN with this new value is 96.6 percent. This result means that we need at least as much data as we got in the simulation model to get sufficient accuracy in the realistic model.

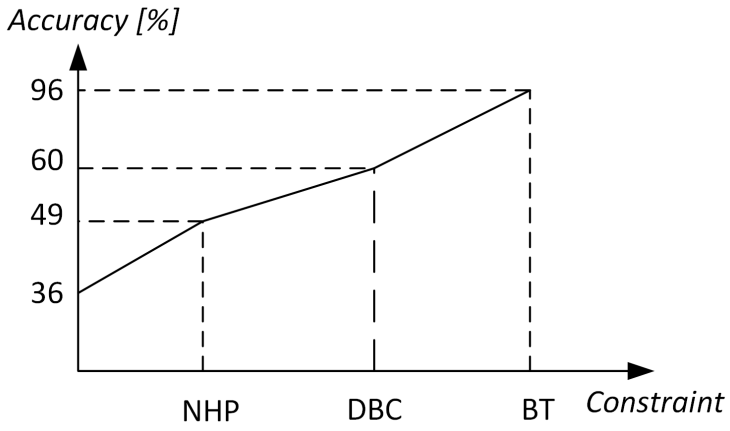

Figure 6: Evolution of the NN accuracy during the integration of the constraints: non-homogeneous Poisson (NHP), Daily Behavior Consideration (DBC), Billing Time (BT).

Figure 6 shows the evolution of the NN accuracy when more complex constraint are added. The simulation process is useful to identify constraints which should be applied in a real dataset. Moreover, the simulation also allows to anticipate the configuration/development of the NN and the mobile application. Now that we have configured the NN based on simulated data, we can ask ourselves how efficient is it when confronted with real data? In this way, a real case on the city of Bastia (France) including more than 450 sensors has been used in order to test the proposed approach.

3.3 Application on the Real Case of Bastia City

Bastia is a city located in Corsica, an island part of France. In this city, as in many others, the problem of parking slots is on everyone's mind because we are missing a lot of space. That's why the city decide to use Smart-parking as a primary solution to resolve these issues. We installed 340 sensors on street to detect parking slots availability. We installed also another 110 sensors on the street to detect limited time park slot

where the user can only park for 40 minutes. We have collected data for one year and we now have historic data for over 1 million parking instances.

At the beginning of the project no real data were available. The simulation has been used to anticipate the configuration phase of the NN that will be used on the real data.

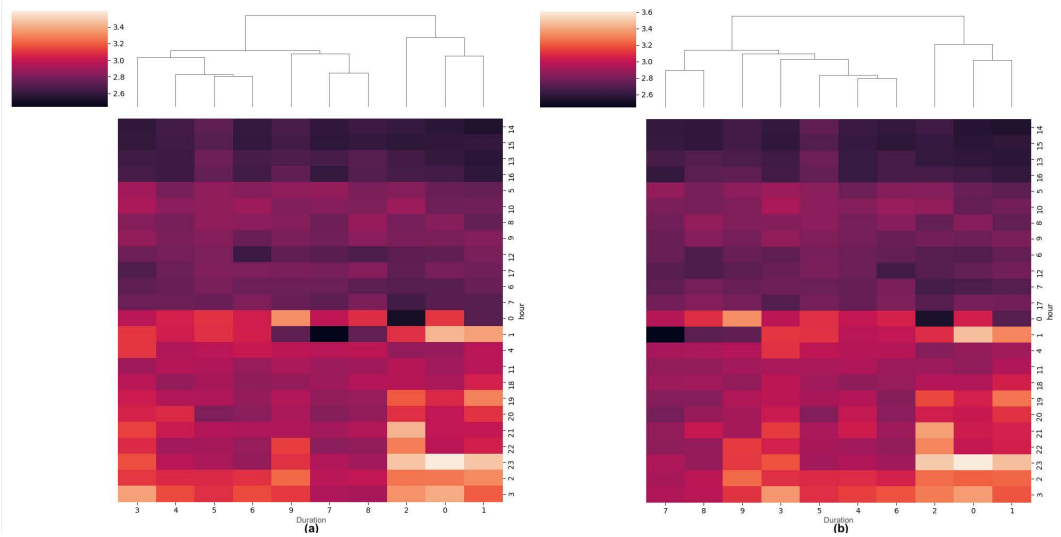

Figure 7: Dendrogram with heat maps showing the comparison between real (a) and simulated data (b).

The two different datasets (real and simulated) have been considered under the same format in order to generate dendrograms with heat maps. Figure 7 shows that the two dendrograms are similar, but we can see more complex links on the left dendrogram ((a) in Figure 7). Obviously, the real data points out some classes that are not considered in the simulated data. However, the similarity of the input data suggests the use of previous trained NN on the real data. The accuracy of the NN on this model with simulated data was 60.61 percent.

After testing with real data we obtain a lower accuracy but we envision to apply the pre-progressing phase on the real data based on the dendrograms analysis and to consider the billing time information in order to significantly increase the accuracy.

4 CONCLUSION AND PERSPECTIVES

In this paper, a combination of discrete-event simulation provided by the discrete-event system specification (DEVS) formalism with a supervised learning algorithm has been proposed in order to classify parking slots depending on their occupation period. More specifically for the Smart-parking context, the proposed approach is based on a neural network learned using input dataset according to a Poisson law and generated by a discrete-event simulation. The simulation results make it possible to obtain a prediction model for the parking slots classification that depends on their occupancy rate.

In perspective, it is envisaged to develop a Markovian prediction model based on reinforcement learning in order to determine an optimal policy of actions to guide the driver in his search for a parking slot. Indeed, a reinforcement learning model based on a reward algorithm (Q-Learning) with a model-free approach could be considered.

REFERENCES

Abadi, M., P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, and et al.. 2016. "TensorFlow: A System for Large-Scale Machine Learning". In *Proc. of the 12th USENIX Conference on Operating Systems Design and Implementation*, pp. 265–283. USA, USENIX Association.

Ahrens, J. H., and U. Dieter. 1982, June. "Computer Generation of Poisson Deviates from Modified Normal Distributions". *ACM Trans. Math. Softw.* vol. 8 (2), pp. 163–179.

Bishop, C. M. 1995, Nov. *Neural Networks for Pattern Recognition*. 1 ed. Oxford University Press, USA.

Bishop, C. M. 2006. *Pattern recognition and machine learning*. springer.

Bolduc, J.-S., and H. Vangheluwe. 2001. "The modelling and simulation package PythonDEVS for classical hierarchical DEVS". *McGill University, MSDL Technical Report MSDL-TR-2001–01*.

Capocchi, L., J. F. Santucci, B. Poggi, and C. Nicolai. 2011, June. "DEVSimPy: A Collaborative Python Software for Modeling and Simulation of DEVS Systems". In *Proc. of 20th IEEE International Workshops on Enabling Technologies*, pp. 170–175.

Chollet, François and others 2015. "Keras". https://github.com/fchollet/keras.

Demisch, A. 2016. "Demand-responsive pricing on the cheap: Estimating parking occupancy with meter payment data". *Transportation Research Record* vol. 2543 (1), pp. 125–133.

Feng, K., S. Chen, and W. Lu. 2018, Dec. "MACHINE LEARNING BASED CONSTRUCTION SIMULATION AND OPTIMIZATION". In *2018 Winter Simulation Conference (WSC)*, pp. 2025–2036.

DEVSimPy group. "DEVSimPy". https://github.com/lcapocchi/devsimpy. Online; accessed 1 March 2020.

Lin, T., H. Rivano, and F. Le Mouël. 2017, December. "A Survey of Smart Parking Solutions". *IEEE Transactions on Intelligent Transportation Systems* vol. 18 (12), pp. pp. 3229–3253.

Lin, T. S. 2015, Dec. *Smart parking : Network, infrastructure and urban service*. Theses, INSA de Lyon.

Mendoza-Silva, G. M., M. Gould, R. Montoliu, J. Torres-Sospedra, and J. Huerta. 2019. "An Occupancy Simulator for a Smart Parking System: Developmental Design and Experimental Considerations". *ISPRS International Journal of Geo-Information* vol. 8 (5).

Pham, T. N., M.-F. Tsai, D. B. Nguyen, C.-R. Dow, and D.-J. Deng. 2015. "A cloud-based smart-parking system based on Internet-of-Things technologies". *IEEE Access* vol. 3, pp. 1581–1591.

Pullola, S., P. K. Atrey, and A. El Saddik. 2007. "Towards an intelligent GPS-based vehicle navigation system for finding street parking lots". In *2007 IEEE International Conference on Signal Processing and Communications*, pp. 1251–1254. IEEE.

Rappin, N., and R. Dunn. 2006. *WxPython in action*. Manning.

Ross, S. M. 2010. In *Introduction to Probability Models (Tenth Edition)* (Tenth Edition ed.)., edited by S. M. Ross. Boston, Academic Press.

Shukla, P., and K. Bhowmick. 2017, March. "To improve classification of imbalanced datasets". In *2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS)*, pp. 1–5.

Tendeloo, Y. V. 2014. "Activity-aware DEVS simulation". Master's thesis, University of Antwerp.

Van Mierlo, S., S. Mustafiz, B. Barocca, Y. Van Tendeloo, and H. Vangheluwe. 2015. "Explicit Modelling of a Parallel DEVS Experimentation Environment". In *Proc. of the Symposium on Theory of Modeling & Simulation*, pp. 860–867. San Diego, CA, USA, Society for Computer Simulation International.

Van Tendeloo, Y., and H. Vangheluwe. 2014. "The Modular Architecture of the Python(P)DEVS Simulation Kernel (WIP)". In *Proc. of the Symposium on Theory of Modeling & Simulation - DEVS Integrative*, DEVS '14, pp. 14:1–14:6. San Diego, CA, USA, Society for Computer Simulation International.

Wallis, L., and M. Paich. 2017, Dec. "Integrating artifical intelligence with anylogic simulation". In *2017 Winter Simulation Conference (WSC)*, pp. 4449–4449.

Willmot, G. E. 1987. "The Poisson-inverse Gaussian distribution as an alternative to the negative binomial". *Scandinavian Actuarial Journal* vol. 1987 (3-4), pp. 113–127.

Zeigler, B. P. 1976. *Theory of Modeling and Simulation*. Academic Press.

Zeigler, B. P., and H. S. Sarjoughian. 2013. "System Entity Structure Basics". In *Guide to Modeling and Simulation of Systems of Systems*, Simulation Foundations, Methods and Applications, pp. 27–37. Springer London.

Zheng, Y., S. Rajasegarar, and C. Leckie. 2015. "Parking availability prediction for sensor-enabled car parks in smart cities". In *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 1–6. IEEE.

AUTHOR BIOGRAPHIES

ANTOINE DOMINICI is a PhD student in Computer Science at the UMR CNRS 6134 Research Lab. of the University of Corsica (France). His email address is dominici_a@univ-corse.fr.

LAURENT CAPOCCHI was born in Bastia, Corsica, France. He received a MSc in electrical engineering from "Ecole Superieure d'Ingénieurs de Nice Sophia Antipolis", ESINSA, Nice, France in 2001 and the PhD in Computer Science from University of Corsica "Pasquale Paoli", Corte, France in 2005. He is SCS (Society for Modeling and Simulation International) member and he is also faculty of the SPE ("Sciences pour l'environnement") CNRS 6134 Research Lab. since 2001. His research goal is to propose an original approach for the modeling and simulation of system of systems using a discrete event specification coupled with machine learning techniques. He is founder member of the DEVSimPy suite open source project (https://github.com/capocchi/DEVSimPy) and he contributes actively to its development. His email address is capocchi@univ-corse.fr

EMMANUELLE DE GENTILI is an Associate Professor at the University of Corsica (France) since 2004. He holds a Ph.D. in modeling and simulation of complex systems in the University of Corsica, SPE CNRS 6134 Research Lab, and an post doctorate in collaboration with the Scuola Superiore Sant'Anna, Pontedera (Italy), on the management of communication protocols of autonomous sensor networks. Today, her work focuses on AI and optimization and integration of heterogeneous systems. Her email address is gentili@univ-corse.fr.

JEAN-FRANCOIS SANTUCCI has been a full professor in computer sciences at the University of Corsica (France), SPE CNRS 6134 Research Lab since 1995. He was assistant professor at the EERIE School, Nimes, France from 1987 to 1995. His email address is santucci@univ-corse.fr.