CISUC TECHNICAL REPORT TR-2017-005-December, 2017

Active Learning Metamodels for Transportation Simulators

Francisco Antunes, Bernardete Ribeiro, Francisco Pereira & Rui Gomes

December 1, 2017

Abstract

Simulation modeling is a well-known and recurrent approach to study the performance of urban systems. Taking into account the recent and continuous transformations within increasingly complex and multidimensional cities, the use of simulation tools is, in many cases, the only feasible and reliable approach to study such dynamic systems. However, simulation models can become very time-consuming when detailed input-space exploration is needed. To tackle this problem, simulation metamodels are often used to approximate the simulator results.

In this paper, we propose an active learning algorithm based on the Gaussian Processes (GP) framework that gathers the most informative data points in batches, according to both their variances and to the relative distance between them. This allows us to explore the simulator input space with fewer data points in a more efficient way, while avoiding computationally expensive simulation runs. We take advantage of the closeness notion encoded into the GP to select batches of points in such a way that they do not belong to the same high variance neighborhoods. In addition, we also suggest two simple and practical user-defined stopping criteria so that the iterative learning procedure can be fully automated.

We illustrate our methodology using three experimental settings. The results show that the proposed methodology is able to improve the exploration efficiency of the simulation input space in comparison with non-restricted batch-mode active learning procedures.

1 Introduction

Urban environments are highly complex systems involving a multitude of both internal and external variables, and their respective interrelationships, which are not often easy to identify. Additionally, these systems also exhibit an inherent stochastic nature and other unknown random phenomena that cannot be realistically described by a closed and tractable mathematical formula.

To successfully overcome the problem of intractability, simulation approaches are often employed to virtually explore the behavior of these urban systems and to assess their performances. Urban dynamics require theoretical approaches and planning methodologies that are capable of modeling the subjacent spatio-temporal transformations process from a multidimentional prespective. In fact, urban planning models usually encompass the possibility of predicting future scenarios of urban intervention focused on infrastructure improvement and service promotion. These models are simplified representations of the urban space, embedded into a computer-generated reality, considered as an experimental ground to understand the long-term performance of urban policies decisions and corresponding interventions [1].

Simulation modeling is a well-known and common approach to study real-world urban systems, specially those that prove to be highly complex to be analyzed through conventional analytic methods [2]. Nevertheless, when detailed with enough realism, urban simulation models can become computationally too time-expensive to run due to their overwhelming complexity. Moreover, if the simulator output space proves to have a complex functional structure, we might need to systematically explore the input domain with further detail, which often requires multiple and exhausting simulation experiments, turning the exploration process virtually intractable.

To address the problem of expensive simulation runs, i.e., simulations that require great computational workload and exhibit prohibitive runtimes, simulation metamodels are often used to approximate the simulation results and thus the simulation model itself. Futhermore, in experimental scenarios in which simulation data is computationally expensive to obtain, active learning also emerges as a powerful approach, as it aims to provide high prediction performance with few data points. Among many significant machine learning tools, Gaussian Processes (GP) [3], a fully Bayesian modeling approach, allow for an intuitive way to develop active learning algorithms, by providing the posterior mean and variance which in turn can be eventually used to search for the most informative data points.

In this paper, we propose an active learning approach, based on GP, that gathers the most informative simulation data points in batches, not only according to their variance but also taking into account the relative distance between them. This allows us to explore the simulator input space with fewer training points in a faster and more efficient manner, while avoiding computationally expensive simulation runs at the same time. Taking advantage of the closeness and similarity notion encoded into the GPs, mainly via the kernel function, to select batches of points that do not simultaneously belong to the same high variance neighborhoods. In addition, we also suggest two simple and practical user-defined stopping criteria so that the iterative learning procedure can be fully automated.

We illustrate our methodology using three independent experimental settings. The first consists of a synthetic data generated by a known function, which plays the role of an arbitrary simulation model. Then, we proceed to a one-dimension study of a Demand-Responsive Transportation (DRT) simulator. Finally, we explore the behavior of a road intersection implemented in a micro traffic simulation software, expanding our study to a two-dimensional input case. The obtained results show that the proposed batch-mode approach is able to improve the exploration efficiency of the simulation input space in comparison with non-restricted batch-mode active learning procedures.

The remainder of the paper is organized as follows. In the next section we provide a brief review on the main background topics and related literature. The proposed approach is detailed in Section 3, followed by the presentation of studied simulation data, experiments and discussion (see Section 4). Finally, we end this paper with the conclusion and several lines of future work.

2 Background

Active learning is a special case of supervised machine learning consisting of an iterative sampling scheme that allows the algorithm to choose the data points from which it learns, and an oracle, i.e., an instance label provider. It is particularly useful under scenarios where labeled data is expensive to obtain. Thus, the general idea of this learning paradigm is to actively select the most informative data points, as few as possible, in order to simultaneously boost the model training efficiency and its prediction performance [4]. When the active learning paradigm was first proposed, the oracle was traditionally represented by a human annotator. Nowadays, however, due to the development of technology, the oracle's role can be taken by an algorithm, a sensor, a simulator, etc. Most importantly is that the oracle is able to provide labeled instances from the ground truth underlying function describing the process of interest.

Depending on how the unlabeled data is presented to the oracle, the active learning algorithms can be divided into two classes, namely, stream-based and pool-based. Whereas in the latter the entire unlabelled data set is available for querying, in the former each data point is presented individually or in sucessive blocks [5]. In addition, because of its intrinsic iterative nature, active learning procedures must be stopped at a certain time. Although there is a vast research under the active learning paradigm, not many approaches suggest a stopping criteria [6, 7]. Some suggestions can be found in the literature such as [8, 9, 10, 6, 7, 11]. However, all the available criteria are fairly identical [4] and, according to [5], there is no best stopping rule that is suitable across all applications.

In most of the proposed active learning algorithms, the queries are presented sequentially, i.e., one at a time. This strategy can become quite inefficient for some heavy learning tasks. One way to address this issue is to select data points in batches in order to speed-up the active learning process. However, to avoid redundancy within the batch, it should simultaneouly account for diversity and informativeness among the selected data points [5]. Several batch-mode schemes have addressed this challenge, most of them within classification problems [12, 13, 14].

Although active learning has been applied in many different fields, we are particularly interested in those of simulation metamodels [15] applications. The development of simulation metamodels has been around since the early 70's [16]. Their main purpose is to serve as surrogates, or emulators, for simulation models so that expensive simulations can be avoided. These models are essentially input/output functions that approximate the true and usually much more complex unknown function inherently defined by the simulation model itself [17]. They are fitted to the input-output data generated by computer experiments and then can be used for prediction purposes, among others [18, 19]. Hence, these simulation metamodels provide a practical framework to explore the behavior of complex and time-consuming simulation experiments in a rather less expensive way.

In our case, we assume that our simulation model is perfectly validated and calibrated with respect to the problem of interest. The metamodel, which should be a valid approximation of the simulation model, is then used to effortlessly explore its inner structure and, consequently, to provide yet another analysis tool for the problem under study. The Gaussian Processes (GP) framework has been widely used for simulation metamodelling [20, 21, 22, 23, 18], and several works combining GP metamodeling with active learning strategies have also been conducted, for example, in [24] and [25]. Its Bayesian formalism provides an easy way to develop algorithms that actively learn with uncertainty. In the context of active learning and simulation metamodeling, the GPs prove to be a powerful and flexible machine learning tool for their ability to easily adapt to the new data sequentially arriving from simulator. This, however, may give rise to a problem known as concept drift, which, according to Gama et al. [26], describes those learning scenarios where the relationship between the input (feature) space and the output variable changes over time. Active learning has also been proposed as a solution to concept drift, as seen, for example, in [27], [28], [29], [30] and [31].

Within the transportation literature, the application of simulation metamodels is relatively recent. Only a handful of works are currently available and, with respect to their application domain, they can be essentially divided into two groups, namely, traffic prediction and network optimization. In 2013, Ciuffo et al. [32] developed a methodology that applies a GP-based metamodel to conduct a sensitivity analysis using the mesoscopic traffic simulator AIMSUN as a case study. Using 512 different input combinations, the authors concluded that the metamodel estimated outputs and the real simulation outputs were significantly similar, thereby showing the strength and parsimony of their methodology.

In [33] the authors developed a Bayesian stochastic Kriging metamodel that simultaneously optimizes travel behavior and dynamic traffic management using, as case study, the real-world corridors of I-270 and MD355 in the state of Maryland, USA. Using a similar case study, Chen et al. [34] used a GP metamodel to approximate the response surface of a transportation simulation with expensive-to-evaluate objective functions and random noise. The goal was to minimize the network-wide average travel time by implementing the optimal toll rates predicted by the metamodel. Similarly, Chen et al. [35] developed a metamodel-based optimization framework to solve the bilevel Mixed Network Design Problem (MNDP).

A mesoscopic Dynamic Traffic Assignment (DTA) simulator, DTALite, was used to evaluate the system response to several network design strategies. The

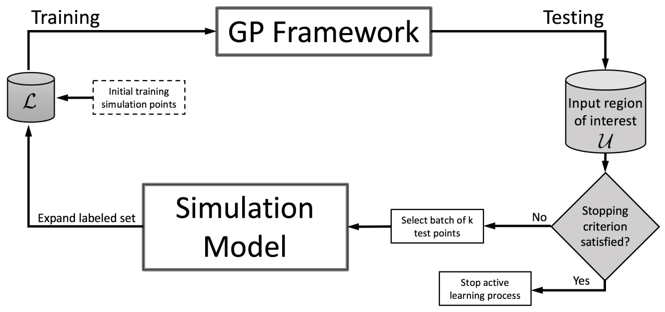

Training
GP Framework
Testing
$\mathcal{L}$
Initial training simulation points
Input region of interest $\mathcal{U}$
Stopping criterion satisfied?
No
Select batch of k test points
Simulation Model
Expand labeled set
Yes
Stop active learning process

Figure 1: Pool-based batch-mode active-learning scheme with a simulation model as oracle.

authors showed that the optimal investment could reduce the network average travel time in approximately 18% during the morning period. Finally, and very recently, Song et al. [36] presented a GP-based metamodeling framework that approximates Dynamic Network Loading (DNL) models. The authors show that the tested DNL metamodels attain high accuracy, providing prediction errors below 8%, and superior computational efficiency, up to 455 times faster than the traditional DNL approaches. Although these works represent important applications of simulation metamodeling in transport problems, the research in this area is still scarce.

3 Approach

In this work, we adopt a pool-based batch-mode active learning approach in which a simulator plays the role of oracle, as depicted in Figure 1. The machine learning model, used as metamodel, is a GP and the unlabeled pool, $\mathcal{U}$, is the input space where we want to explore the simulation behavior. The pool of labeled instances, $\mathcal{L}$, is formed by the results of the simulation runs already performed.

From a regression perspective, we want to establish a functional relationship between the simulation inputs, $\mathbf{x} \in \mathbb{R}^D$, where D is the number of inputs, and its output, $y \in \mathbb{R}$, by assuming a GP prior over this relation, $y = f(\mathbf{x})$. Then, taking advantage from the Bayesian formalism from which the GP frameworks derives, we aim to infer the conditional distribution of the output given a set of unlabeled inputs. By doing so we are not only bypassing the simulation computational workload but also providing a way to effortlessly approximate and therefore analyze the simulator behavioral structure. Notice that the simulation model is treated as a black box from which we aim to get better insights in terms of its functional behavior, while avoiding as many simulation runs as possible.

In the following we present the basis of the Gaussian Processes and then we detail the proposed active learning procedure which is build upon this modeling framework.

3.1 Gaussian Processes

A GP [3] is a stochastic process completely characterized its mean and covariance (or kernel) functions, respectively denoted as $m_f(\mathbf{x})$ and $k_f(\mathbf{x}, \mathbf{x}')$, with $\mathbf{x}$ and $\mathbf{x}'$ being two input data points and simply denoted by $\mathcal{GP}(m_f(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}'))$ where $m_f(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and $k_f(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m_f(\mathbf{x}))(f(\mathbf{x}') - m_f(\mathbf{x}'))]$.

More formally, the GP framework assumes a prior over functions, i.e., $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $f(\mathbf{x}) \sim \mathcal{GP}(m_f(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}'))$. For simplicity, it is common practice to fix $m_f(\mathbf{x}) = 0$. Thus, the prior over the latent function is given by $p(\mathbf{f}|\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n) = \mathcal{N}(0, K_f)$ where $\mathbf{f} = [f_1, f_2, \ldots, f_n]^\top$, $f_i \triangleq f(\mathbf{x}_i)$ and K_f is the covariance matrix, with its elements given by $[K_f]_{ij} = k_f(\mathbf{x}_i, \mathbf{x}_j)$. Most of the covariance functions have several free parameters, which can be optimized to fit the training data by maximizing the marginal likelihood.

After parameters are obtained, the conditional distribution at a new test point $\mathbf{x}_*$ is given by $f_*|X, \mathbf{y}, \mathbf{x}_* \sim \mathcal{N}(k_{f*}^\top [K_y]^{-1}\mathbf{y}, k_{f**} - k_{f*}^\top [K_y]^{-1} k_{f*})$, where $k_{f*} = k_f(X, \mathbf{x}_*)$, $k_{f**} = k_f(\mathbf{x}_*, \mathbf{x}_*)$ and $\mathbf{y}$ is the vector of the target values. Thus, we can naturally use the predicted Gaussian distribution at any given test point to guide the active learning process and therefore learn with its associated uncertainty.

3.2 Active Learning Procedure

The strategy presented in Figure 2 is the general batch-mode active learning procedure used in this work and it serves as the base for other algorithms forwardly presented and tested. Here, $\mathcal{L}$ represents the set of labeled training points, whereas $\mathcal{U}$ the set of unlabeled ones. The latter is defined according to input space region we aim to explore. This algorithm selects batches of k test points with highest variance (provided by the GP) in a way that each point does not originate from the same high variance neighborhood. Taking into account the spatial notion of closeness and similarity encoded into the GP via its kernel function, it is expected that spatially closer input points are more likely to have similar output values. Therefore, the hypothesis is that sampling multiple batch points from the same region may not be efficient. To avoid this situation we introduce β to therefore control the minimum distance between the selected active learning points. This parameter is a ratio with respect to the maximum possible distance between two any points (diameter) in the input space. Notice that this approach is only valid for continuous input metric spaces. So, if $\beta = 0.4$, then the minimum distance between the points is 40% of the maximum distance. The parameter k defines the size of the batch.

We also propose two simple variance-based stopping criteria controlled by α_1 and α_2, respectively. The first, which we call Criterion A, states that the algorithm stops when the total current variance (TCV) of the test points, at iteration i, is less than $(1 - \alpha_1)$% of the initial total variance (ITV) at iteration 0. Thus, if, for example, $\alpha_1 = 0.3$, the process stops when TCV is reduced 70% with respect to ITV, i.e., when $ITV(1 - \alpha_1) \geq TCV$. Note that, in stead of the total variance, which is simply the sum of the variances of all test points, we could have considered the average variance per iteration. However, since this criterion is defined as a ratio, the total number of test point would cancel out.

On the other hand, Criterion B is defined as a ratio between the average variance of the training ($AVTr$) points and the average variance of the test ($AVTs$) points at each iteration i. Here, contrary to Criterion A, since the total number of training and testing points is not the same, it makes sense to consider the average. When $AVTr/AVTs \geq \alpha_2$, the algorithm stops. The average variance at training points is less than the average variance at testing points, so this ratio lies in $[0, 1]$. Moreover, as the process advances, $AVTs$ is likely to decrease, while $AVTr$ is expected to approximately maintain its values. However, this is a more demanding criterion to be satisfied if α is close to 1. If the model, in our case the GP, is very certain at the training points and the contrary at the test points, $AVTr/AVTs \approx 0$, which will prevent the algorithm from converging at an acceptable speed. Its performance will also depend on the noise structure of the underlying function being estimated.

In each iteration a new GP model is fitted to $\mathcal{L}$. The hyper-parameters are obtained by maximizing the likelihood function conditional on this training set in a Leave-One-Out Cross-Validation (LOOCV) scheme. Afterwards, the trained GP is used to predict the simulation output values (labels) associated to the unlabeled points in $\mathcal{U}$, therefore avoiding many simulations runs. Then, several testing points are selected according to the approach described in Figure 2, their respective true labels are obtained via oracle, i.e., the simulator, and finally $\mathcal{L}$ is expanded. This iterative process is repeated until the chosen stopping criterion is satisfied.

Given the parametric nature of the proposed algorithm, many variants can be derived depending on the concrete values assigned to α_1, α_2, β and k, as well as on the used stopping criteria. We test the four following combinations which are built upon the base structure of this general approach:

- **Algorithm 1**: base approach + Criterion A with no space restriction, $\beta = 0$.
- **Algorithm 2**: base approach + Criterion A with space restriction, $\beta \geq 0$.
- **Algorithm 3**: base approach + Criterion B with no space restriction, $\beta = 0$.
- **Algorithm 4**: base approach + Criterion B with space restriction, $\beta \geq 0$.

The parameters used in these algorithms, α_1, α_2 and β, vary according to the different experiments conducted with the different simulators in study and $k = 3$. Moreover, note that in Algorithms 1 and 3 we assume $\beta = 0$, which means that there is no spatial restriction during the batch selection. Thus, these cases correspond to the standard batch-mode scheme, serving us as baseline approaches. In the following, we present and discuss the results obtained within three experimental settings. The values of the parameters used in these algorithms vary according to the experiments conducted with the different simulators in study.

```
    Input: $\alpha_1, \alpha_2, \beta \in [0, 1], k \in \mathbb{N}, \mathcal{L}, \mathcal{U}$.
    While $\frac{ITV-CTV}{ITV} < \alpha_1$ or $\frac{AVT_r}{AVT_s} < \alpha_2$ do
1:      Train a GP with $\mathcal{L}$ and obtain the predicted labels for each point in $\mathcal{U}$,
        $k_{f*}^{\top}[K_y]^{-1}\mathbf{y}$, and their corresponding variances, $k_{f**} - k_{f*}^{\top}[K_y]^{-1}k_{f*}$.
2:      Determine the top $k$ highest variance test points, $top_k$, so that they are not
        mutually closer than $\beta \times 100\%$ of the diameter of $\mathcal{U}$.
3:      Obtain the true labels for $top_k$, via simulator, and define $\mathcal{L}_+$ as the new
        labeled set.
4:      Set $\mathcal{L} = \mathcal{L} \cup \mathcal{L}_+$.
5:      Update stopping criterion.
    end
```

Figure 2: General bath-mode active learning approach with spatial restriction.

4 Experiments

In this section, we illustrate the proposed methodology using three independent experimental settings. The first consists of a synthetic data generated by a known function playing the role of an arbitrary simulation model. Then, we proceed to a one-dimension study of a Demand-Responsive Transportation (DRT) simulator. Finally, we move to a two-dimensional study of a simple road intersection implemented in a free and open-source microtraffic simulation software. We use the implementation provided by Rasmussen and Williams [3] and select the Squared Exponential function as the GP kernel.

4.1 Toy data set

For the first set of experiments, we used a toy data set generated by $f(x) = cos(5x) + \frac{1}{2}x + 2\,U(0,1)$, which plays the role of oracle, where $U(a, b)$ denotes the Uniform distribution in the interval $[a, b]$. Figure 3 shows the initial GP learning state. The initial labeled pool, $\mathcal{L}$, is comprised of 10 randomly generated training points, $\mathcal{U}$ is formed by 10000 unlabeled test points uniformly spaced in $[-6, 6]$. Figure 4 and 5 present the results. Each new active learning point added to the training set is labeled with the number of the iteration in which it was selected. Consequently, data points with the same number belong to the same batch.

Comparing Algorithms 1 and 2, the superiority of the latter is clear from an efficiency point-of-view (see Figure 4). For the same criterion threshold ($\alpha_1 = 0.6$), Algorithm 2 is over than four times faster than Algorithm 1, due to the space restriction induced by $\beta = 0.2$. This means that the testing points constituting the batches were not selected from the same high variance neighborhoods, scattering the input exploration process and thus turning it more efficient. Moreover, it is important to note the number of iterations alone does not directly measure the real involved computational workload, which is intrinsically related with the simulation model. As we adopted a batch-mode active learning scheme, we should additionally take into account the size of the batch,

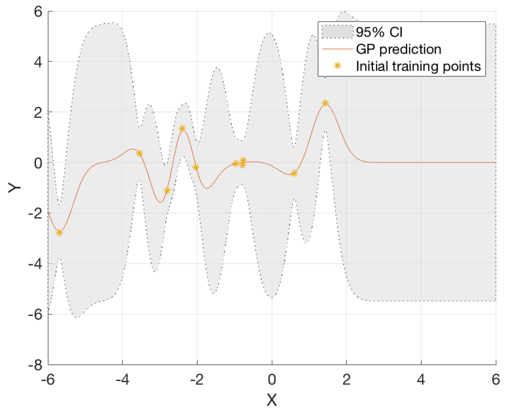


Figure 3: Initial learning state for the toy data set, where the first 10 training points were randomly scattered in the input domain $[-6, 6]$. This corresponds to iteration 0, which is shared by the four algorithms in study.

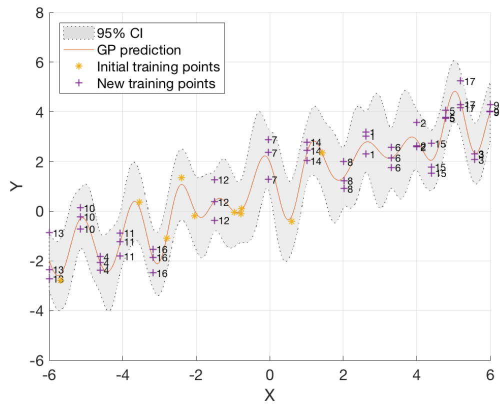
k. Therefore, whereas Algorithm 1 requested the oracle $17 \times 3 = 51$ times, in Algorithm 2 this number decreased to $4 \times 3 = 15$.

For Algorithms 3 and 4, a similar scenario has occurred. Using the same stopping rule, based on Criterion B with $\alpha_2 = 0.6$, Algorithm 4 required three less iterations than Algorithm 3. This represents a total difference of $3 \times 3 = 9$ simulation requests. However, note that, as previously mentioned in Section 3.2, Criterion B, may be harder to satisfy, which explains the lesser performance in comparison in both Algorithms 1 and 2.

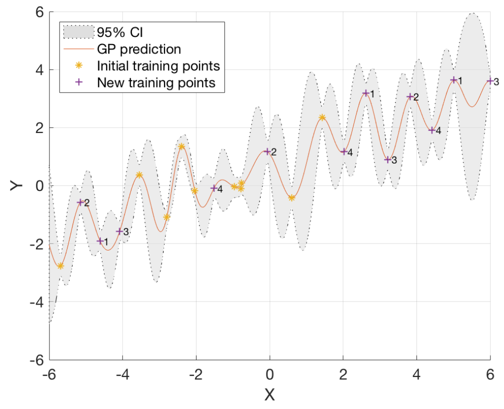
4.2 DRT simulator

Demand Responsive Transportation (DRT) systems are a kind of hybrid transportation approach between the taxi and bus solutions that address the problems that emerge from the use of fixed routes and schedules, typically found in regular public road transportation. From the transport operators' point-of-view, this traditional approach can prove to be quite expensive and inefficient in lower population density zones, such as rural areas, and in certain periods of the day. Both cases are characterized by a low, variable and unpredictable demand. Thus, DRT systems aim to provide transportation solutions that are able to adapt, in real-time, its routes and frequencies to match the actual observed demand.

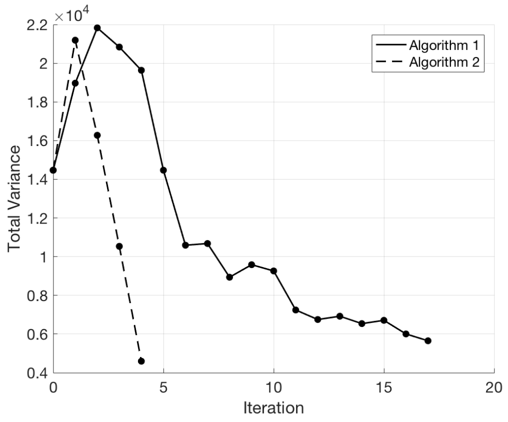
Service design is critical for the success of DRT systems, so decision-makers need to understand well how the different ways of operating the service affect its performance. The flexibility of DRTs may cause organizational problems such as, a) the number and type of requests may involve an exceedingly high number of vehicles, b) very sparse requests that are hard to combine efficiently or c) the quality of the service in terms of pickup/delivery time and travel duration might



(a)

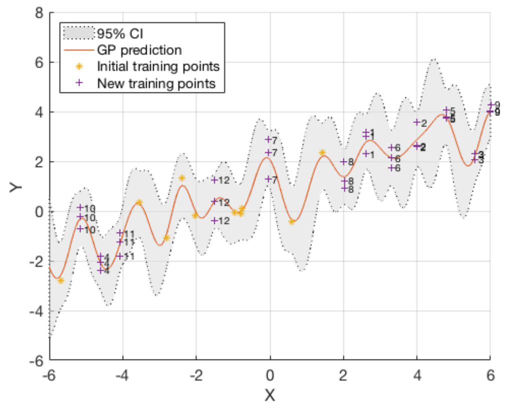


(b)

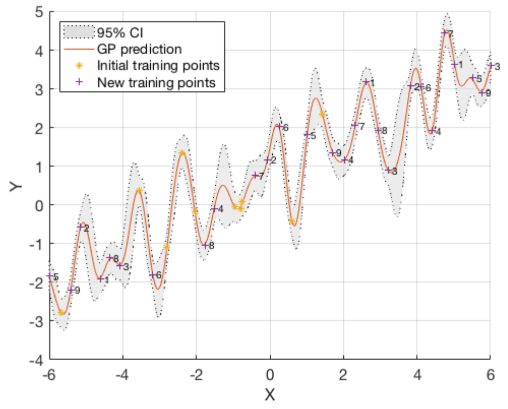


(c)

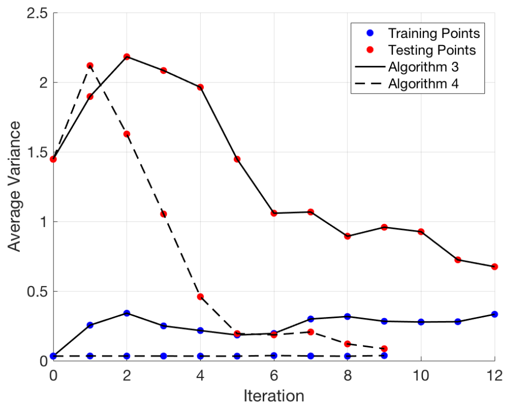
Figure 4: Final results for the toy data set using (a) Algorithm 1 with user-defined parameters $\alpha_1 = 0.6$, $\beta = 0$ and $k = 3$, and (b) Algorithm 2 with $\alpha_1 = 0.6$, $\beta = 0.2$ and $k = 3$. Panel (c) shows the number of iterations for both algorithms.



(a)



(b)



(c)

Figure 5: Final results for the toy data set using (a) Algorithm 3 with user-defined parameters $\alpha_2 = 0.4$, $\beta = 0$ and $k = 3$, and (b) Algorithm 4 with $\alpha_2 = 0.4$, $\beta = 0.2$ and $k = 3$. Panel (c) shows the number of iterations for both algorithms.

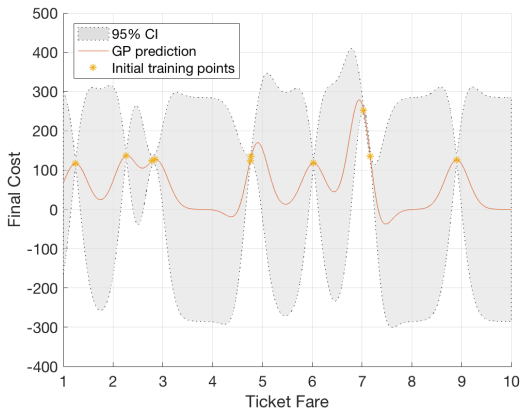
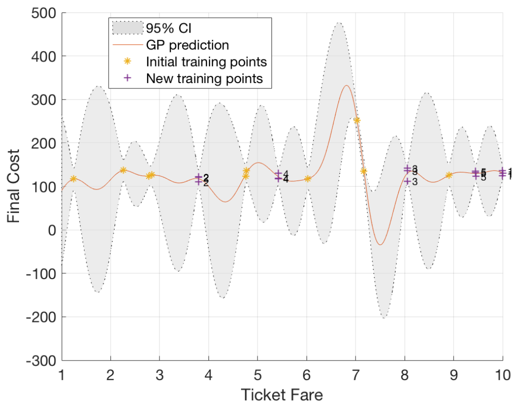


Figure 6: Initial learning state for the DRT simulation data set, where the first 10 training points were randomly scattered in the input domain $[1, 10]$. This corresponds to iteration 0, which is shared by the four algorithms in study.

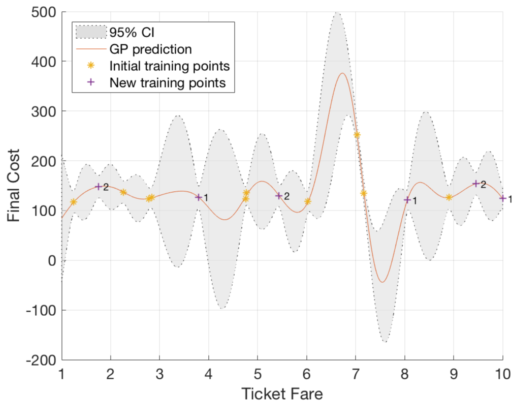
not be guaranteed with the available resources or when unpredictable events occur. These effects are often studied through simulation, whose purpose is to obtain a better understanding of the behavior of a system under a given set of input conditions, even with uncertain events. The performance of these systems can be determined by observing what happens on the network, during simulation, for different input conditions. However, when dealing with real-world events (especially with high degree of dynamism) and extremely complex road networks and demand, simulation models can become very time-consuming.

We now consider the problem of exploring the outcome of the DRT simulation model developed by Gomes et al. [37]. This simulation system integrates four submodels, covering the service area, trip requests (demand), vehicles, and real time events. It has 22 inputs/parameters and, among a few outputs that measure the system overall performance, we focus on the DRT system total operating cost. For a given demand structure, different input combinations will lead to different costs and service quality levels. We loaded the simulator with a real road-network structure, within the metropolitan region of Oporto (Portugal), with symbolic parameters values. For the sake of illustration, we then considered the Ticket Price as the input domain for which we aim to explore the outcome, Total Cost, of the simulator, maintaining the remaining inputs unchanged. From previous experiments, we concluded that, for our particular application, the simulation running times do not vary significantly from each other. Therefore, it does make sense to focus only on the number of iterations to assess the performances of the studied algorithms.

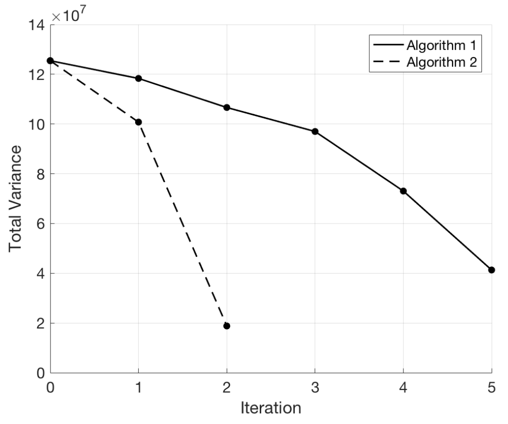
Similarly to the previous experimental setting, we decided to explore the input domain contained in the interval $[1, 10]$ using 10000 uniformly spaced test points. This test set corresponds to $\mathcal{U}$, whereas $\mathcal{L}$ is constituted by 10 random simulation data points within the same interval, as presented in Figure 6. The



(a)

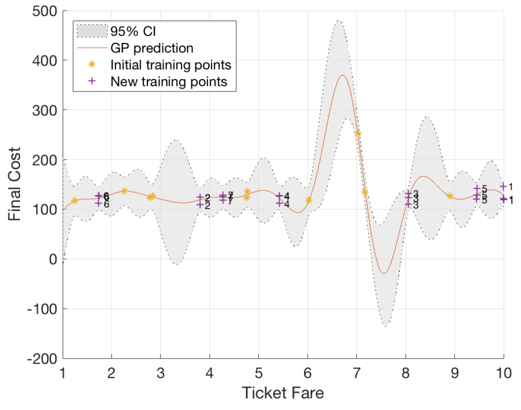


(b)

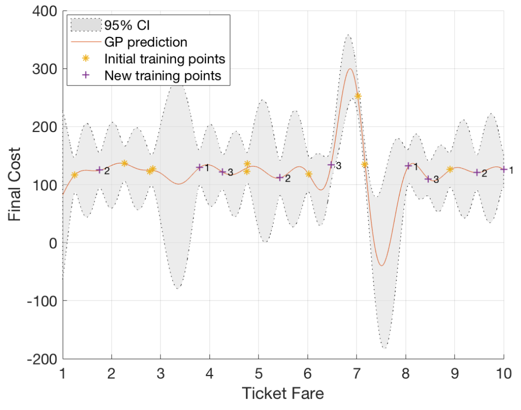


(c)

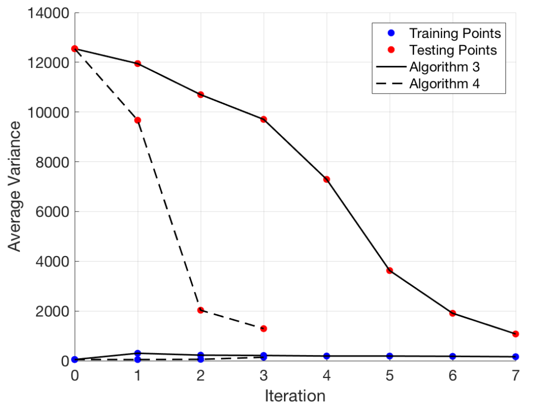
Figure 7: Final results for the DRT simulation data using (a) Algorithm 1 with user-defined parameters $\alpha_1 = 0.6$, $\beta = 0$ and $k = 3$, and (b) Algorithm 2 with $\alpha_1 = 0.6$, $\beta = 0.2$ and $k = 3$. Panel (c) shows the number of iterations for both algorithms.



(a)



(b)



(c)

Figure 8: Final results for the DRT simulation data using (a) Algorithm 3 with user-defined parameters $\alpha_2 = 0.1$, $\beta = 0$ and $k = 3$, and (b) Algorithm 4 with $\alpha_2 = 0.1$, $\beta = 0.2$ and $k = 3$. Panel (c) shows the number of iterations for both algorithms.

results for this simulation data are generally aligned with the results obtained for the toy set experiment. Again, and as depicted in Figure 7, Algorithm 2 is more than two times faster than Algorithm 1. For the same level of variance reduction, induced by $\alpha_1 = 0.60$ (about 40% of the initial total variance), the former requested the simulator $5 \times 3 = 15$ times, against 2×3 requests in the latter. It is obvious that the less runs the simulator executes the better. Therefore, for the same stopping criteria and threshold, Algorithm 2 proved to be more efficient as it was able to scatter the learning points forming the batch along different high variance neighborhoods.

For Algorithms 3 and 4 we obtained an interesting result which highlights our concerns presented in Section 3.2 regarding Criterion B. In Figure 6 we can obverse that for the first approximation (Iteration 0), the GP model assigned very little variance to the training points, meaning that, in the context of Criterion B, $AVTr \approx 0$. As the process iteratively evolves, i.e., as more simulation data points are added to the expanding data set, $AVTr$ seems to remain close to zero. On the other hand, $AVTs$ clearly shows a decreasing behavior towards zero. In several preliminary experiments, which we do not present due to space restrictions, we noticed that setting, for example, $\alpha_2 = 0.6$, as we did for α_1, was too ambitious for both Algorithms 3 and 4 to yield competitive performance. This means that these algorithms were taking too many iterations to converge to be fairly comparable to Algorithms 1 and 2. Therefore, after fixing $\alpha_2 = 0.1$, which is equivalent to say that the stopping criterion is satisfied when $AVTr$ is approximately 40% of $AVTs$, we concluded that both Algorithms 3 and 4 converged in reasonable and comparable running times. Despite these initial configuration problems, these algorithms attained similar results to those of Algorithm 1 and 2. It is worthwhile to mention that, once again, the restriction applied during the formation of the batches was a decisive factor in the reduction of the number of iterations.

4.3 Traffic simulator

In this section we move to a microscopic traffic simulation example, by exploring a road intersection implemented with the Simulation of Urban Mobility (SUMO) [38]. The studied example consists of a simple signalized intersection, depicted in Figure 9. Traffic flows in three directions only, North-South (NS), West-East (WE) and East-West (EW). The vertical axis is dedicated to important heavy vehicles, whereas in the horizontal axis we only have light passenger car traffic. Moreover, the simulation model is designed to give priority to the NS traffic over the remaining flows. Therefore, it is expected that if this flow increases, the horizontal traffic flows will potentially form more and longer queues, consequently increasing the overall total waiting time.

During each simulation run, the demand, or traffic flow, generated from each operational axis is randomly generated according to a Poisson distribution, approximated by a Binomial distribution with parameter $p \in [0, 1]$. This parameter sets how many vehicles are generated, on average, within a certain period of time. For example, if $p = 1/s$, then it means that one vehicle is ex-

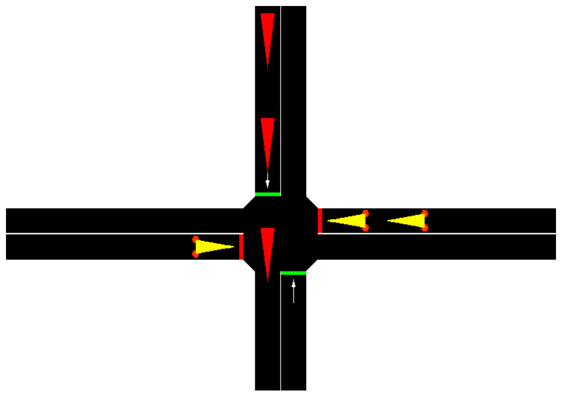
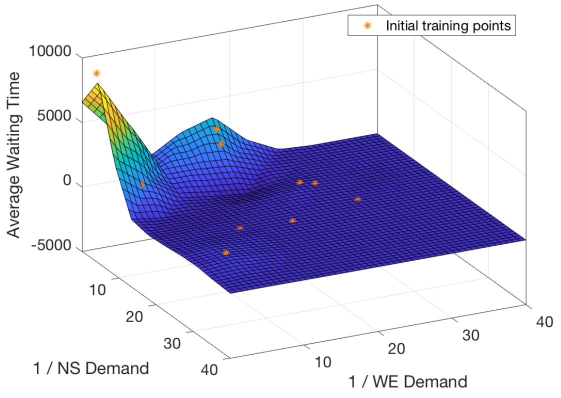


Figure 9: Visualization of the intersection with four approach lanes implemented in SUMO.

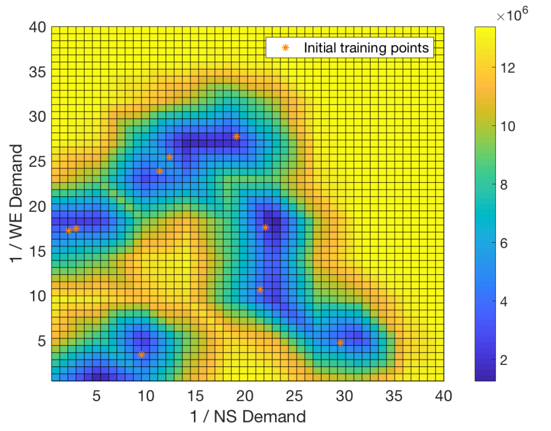
pected every interval of s seconds. Notice that the traffic flow actually increases when $s \to 0$.

The simulated example encompasses three input parameters that have a direct influence in the intersection performance, namely, the NS, WE and EW demands, each of which associated with different Binomial parameters. Moreover, it is assumed that the three different traffic flows are mutually independent. To assess the performance of the simulated road networks, SUMO has a large number of different output measures. Raw vehicle positions, trip and route information and simulation state statistics are just a few examples of possible outputs. For the sake of illustration of our methodology, we decided to focus on the total waiting time spent by all the vehicles crossing the intersection as our aggregated traffic performance measure. Our objective is to use the proposed active learning scheme to explore the simulation input space and to evaluate how it affects the total waiting time. Therefore, following a similar experimental design of the one-dimensional analyses presented in Sections 4.1 and 4.2, we now extend our study to a two-dimensional case where the traffic demands from NS and WE operational axes are considered as inputs, and the expected vehicular waiting time is our output performance of interest.

The new input region of interest ($\mathcal{U}$) is defined by the square $[0, 40] \times [0, 40]$, from which 10 random training points were selected, corresponding to the initial set of simulation runs ($\mathcal{L}$), as depicted in Figure 10(a). On the other hand, Figure 10(b) shows the variance across the entire test region. As expected, the variance near the training points is lower than the variance associated to the test points. Starting from this initial learning stage, our approach is designed to actively search for the top k highest variance neighborhoods (yellow tones regions), that are not mutually within a radius of $\beta \times 100\%$ of the diameter of the input region. In any case, in this first approximation we can already observe that the values of the average waiting time (z-axis) tend to increase when both NS and WE demands increase. This observation matches our initial guess regarding the simulator output behavior.

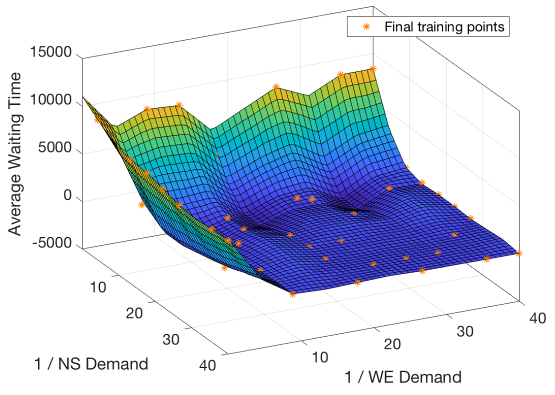


(a)

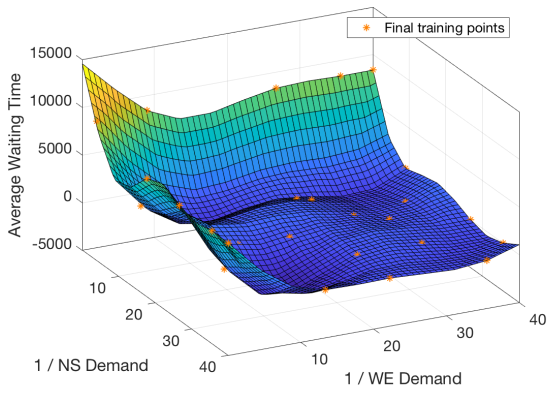


(b)

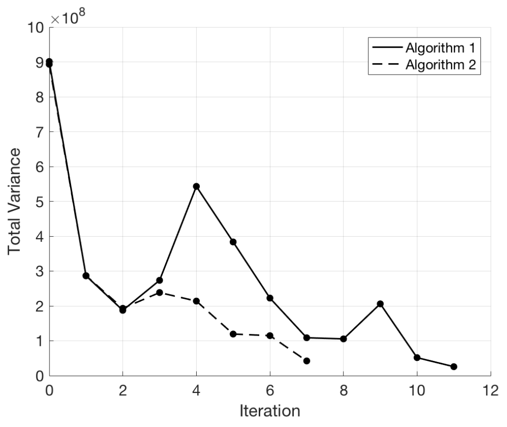
Figure 10: (b) Initial learning state for the traffic simulation data, where the first 10 training points were randomly scattered in the input space $[0, 40] \times [0, 40]$. This corresponds to iteration 0, which is shared by the four algorithms in study. (c) Variance behavior across the input region.



(a)

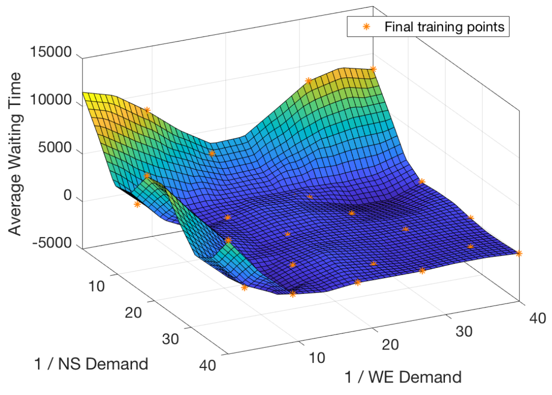


(b)

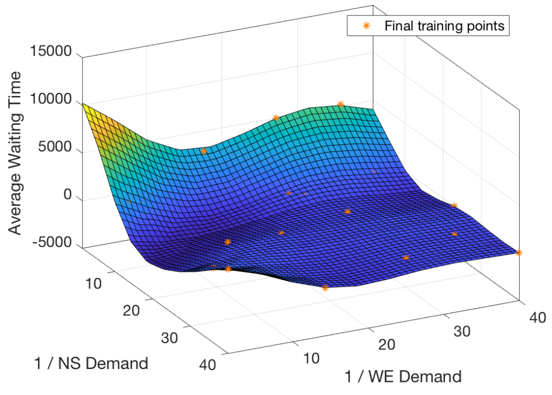


(c)

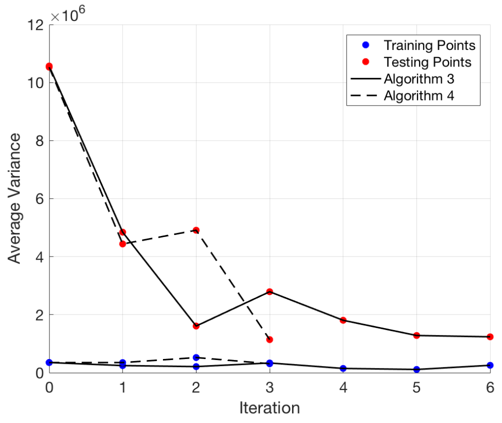
Figure 11: Final results for the traffic simulation data using (a) Algorithm 1 with user-defined parameters $\alpha_1 = 0.95$, $\beta = 0$ and $k = 3$, and (b) Algorithm 2 with $\alpha_1 = 0.95$, $\beta = 0.3$ and $k = 3$. Panel (c) shows the number of iterations for both algorithms.



(a)



(b)



(c)

Figure 12: Final results for the toy data set using (a) Algorithm 3 with user-defined parameters $\alpha_2 = 0.2$, $\beta = 0$ and $k = 3$, and (b) Algorithm 4 with $\alpha_2 = 0.2$, $\beta = 0.2$ and $k = 3$. Panel (c) shows the number of iterations for both algorithms.

Figure 11 and 12 present the final results, showing fairly identical GP approximations across the four algorithms. We observe that Algorithm 1 took 11 iteration to achieve a total variance reduction of 95%, against seven iterations from Algorithm 2 (see Figure 11(c)). Both are based on Criterion A and on the same stopping threshold. However, due to the proposed space restriction (imposed by $\beta = 0.3$), the latter presents a more efficient performance than the former, with a difference of $(11 - 7) \times 3 = 12$ simulation runs. Finally, we can see from Figure 12(c) that Algorithm 4 took three iterations to stop, whereas Algorithm 3, whose batch formation is not restricted by β, required seven to satisfy Criterion B with $\alpha_2 = 0.2$. Although these differences in the iteration numbers may seem to be of little significance for the current academic example, they can prove to be quite relevant in real-world and thus computationally heavy simulation models, which can take up to several days to accomplish.

5 Conclusion and Future Work

In this paper we presented a restricted batch-mode active learning approach, along with two practical user-defined stopping criteria, in the context of transport simulation metamodeling. The proposed algorithm seeks for the most informative test points in restricted batches. The parameter β, which represents a fraction of the maximum possible distance (diameter) within the input region of interest, controls the minimum distance between each gathered point. This prevents each batch from being formed with points from the same high variance regions, thus making the learning process faster and more efficient. Ultimately, our objective is to obtain a reasonable understanding of the simulator under study with as few simulation runs as possible.

The results obtained from three independent experimental settings show that the introduction of a spatial restriction, induced by β, in the formation of the batch seems to turn the metamodeling process more efficient. Additionally, we concluded that different data contexts and experimental settings require different parameter values in order to attain comparable results.

There are several ways in which this work can be improved. The procedure to select the algorithm parameters (α_1, α_2 and k) was conducted in a rather informal way, essentially for the sake of illustration. We intend to expand the current study not only with more parameter configurations but also explore their relationship with the size, shape and dimensionality of the input spaces of interest. Additionally, we aim to test our approach using a large-scale transportation problem in order to assess its feasibility in comprehensive real-world applications. Finally, other challenging problems such as developing strategies to fine tune these parameters, generalization to higher input space dimensions and sensitivity analysis, will also be addressed in future lines of work.

Acknowledgment

The Fundação para a Ciência e Tecnologia (FCT) is gratefully acknowledged for financing the first author with grant No. PD/BD/128047/2016 in the framework of the MIT-Portugal Doctoral Program in Transportation Systems. This work was also supported by the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Individual Fellowship H2020-MSCA-IF-2016, ID number 745673.

References

[1] M. C. Marengo, "Urban simulation models: Contributions as analysis-methodology in a project of urban renewal," *Current Urban Studies*, vol. 2, no. 03, p. 298, 2014.

[2] A. M. Law, W. D. Kelton, and W. D. Kelton, *Simulation modeling and analysis.* McGraw-Hill New York, 1991, vol. 2.

[3] C. E. Rasmussen and C. Williams, *Gaussian processes for machine learning (Adaptive computation and machine learning).* The MIT Press, 2005.

[4] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2009.

[5] X. Wang and J. Zhai, *Learning from Uncertainty.* CRC Press, 2016.

[6] A. Vlachos, "A stopping criterion for active learning," *Computer Speech & Language*, vol. 22, no. 3, pp. 295–312, 2008.

[7] W. Wang, W. Cai, and Y. Zhang, "Stability-based stopping criterion for active learning," in *Data Mining (ICDM), 2014 IEEE International Conference on.* IEEE, 2014, pp. 1019–1024.

[8] M. Ghayoomi, "Using variance as a stopping criterion for active learning of frame assignment," in *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing.* Association for Computational Linguistics, 2010, pp. 1–9.

[9] F. Laws and H. Schätze, "Stopping criteria for active learning of named entity recognition," in *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1.* Association for Computational Linguistics, 2008, pp. 465–472.

[10] S.-J. Huang, R. Jin, and Z.-H. Zhou, "Active learning by querying informative and representative examples," in *Advances in neural information processing systems*, 2010, pp. 892–900.

[11] J. Zhu, H. Wang, E. Hovy, and M. Ma, "Confidence-based stopping criteria for active learning for data annotation," *ACM Transactions on Speech and Language Processing (TSLP)*, vol. 6, no. 3, p. 3, 2010.

[12] K. Brinker, “Incorporating diversity in active learning with support vector machines,” in *ICML*, vol. 3, 2003, pp. 59–66.

[13] S. C. Hoi, R. Jin, and M. R. Lyu, “Large-scale text categorization by batch mode active learning,” in *Proceedings of the 15th international conference on World Wide Web.* ACM, 2006, pp. 633–642.

[14] Z. Xu, R. Akella, and Y. Zhang, “Incorporating diversity and density in active learning for relevance feedback,” in *European Conference on Information Retrieval.* Springer, 2007, pp. 246–257.

[15] L. W. Friedman, *The simulation metamodel.* Springer Science & Business Media, 2012.

[16] R. R. Barton, “Simulation metamodels,” in *Simulation Conference Proceedings, 1998. Winter*, vol. 1. IEEE, 1998, pp. 167–174.

[17] J. P. Kleijnen and W. C. Van Beers, “Application-driven sequential designs for simulation experiments: Kriging metamodelling,” *Journal of the Operational Research Society*, vol. 55, no. 8, pp. 876–883, 2004.

[18] J. P. Kleijnen, “Kriging metamodeling in simulation: A review,” *European journal of operational research*, vol. 192, no. 3, pp. 707–716, 2009.

[19] J. P. Kleijnen and R. G. Sargent, “A methodology for fitting and validating metamodels in simulation,” *European Journal of Operational Research*, vol. 120, no. 1, pp. 14–29, 2000.

[20] A. Boukouvalas, “Emulation of random output simulators,” Ph.D. dissertation, Aston University, 2010.

[21] T. Chen, K. Hadinoto, W. Yan, and Y. Ma, “Efficient meta-modelling of complex process simulations with time–space-dependent outputs,” *Computers & chemical engineering*, vol. 35, no. 3, pp. 502–509, 2011.

[22] S. Conti and A. OâĂŹHagan, “Bayesian emulation of complex multi-output and dynamic computer models,” *Journal of statistical planning and inference*, vol. 140, no. 3, pp. 640–651, 2010.

[23] B. Jones and R. T. Johnson, “Design and analysis for the gaussian process model,” *Quality and Reliability Engineering International*, vol. 25, no. 5, pp. 515–524, 2009.

[24] J. A. Christen and B. Sansó, “Advances in the sequential design of computer experiments based on active learning,” *Communications in Statistics-Theory and Methods*, vol. 40, no. 24, pp. 4467–4483, 2011.

[25] T. Pfingsten, “Bayesian active learning for sensitivity analysis,” in *European Conference on Machine Learning.* Springer, 2006, pp. 353–364.

[26] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[27] B. Kurlej and M. Wozniak, "Active learning approach to concept drift problem," *Logic Journal of IGPL*, vol. 20, no. 3, pp. 550–559, 2011.

[28] W. Chu, M. Zinkevich, L. Li, A. Thomas, and B. Tseng, "Unbiased online active learning in data streams," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 2011, pp. 195–203.

[29] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with evolving streaming data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 2011, pp. 597–612.

[30] I. Zliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2014.

[31] C. H. Park and Y. Kang, "An active learning method for data streams with concept drift," in *Big Data (Big Data), 2016 IEEE International Conference on.* IEEE, 2016, pp. 746–752.

[32] B. Ciuffo, J. Casas, M. Montanino, J. Perarnau, and V. Punzo, "Gaussian process metamodels for sensitivity analysis of traffic simulation models: Case study of aimsun mesoscopic model," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2390, no. 2390, pp. 87–98, 2013.

[33] L. Zhang, X. He, C. Xiong, Z. Zhu *et al.*, "Bayesian stochastic kriging metamodel for active traffic management of corridors," in *IIE Annual Conference. Proceedings.* Institute of Industrial and Systems Engineers (IISE), 2014, p. 1790.

[34] X. M. Chen, L. Zhang, X. He, C. Xiong, and Z. Li, "Surrogate-based optimization of expensive-to-evaluate objective for optimal highway toll charges in transportation network," *Computer-Aided Civil and Infrastructure Engineering*, vol. 29, no. 5, pp. 359–381, 2014.

[35] X. Chen, Z. Zhu, X. He, and L. Zhang, "Surrogate-based optimization for solving a mixed integer network design problem," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2497, no. 2497, pp. 124–134, 2015.

[36] W. Song, K. Han, Y. Wang, T. Friesz, and E. del Castillo, "Statistical metamodeling of dynamic network loading," *Transportation Research Procedia*, vol. 23, pp. 263–282, 2017.

[37] R. Gomes, J. P. de Sousa, and T. Galvão, “An integrated approach for the design of demand responsive transportation services,” in *Computer-based Modelling and Optimization in Transportation.* Springer, 2014, pp. 223–235.

[38] 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, 2012.