Directional Grid-Based Search for Simulation Metamodeling Using Active Learning

Francisco Antunes[1(✉)], Francisco Pereira[2], and Bernardete Ribeiro[1]

[1] University of Coimbra, Rua Sílvio Lima - Polo II, 3030-790 Coimbra, Portugal
fnibau@uc.pt

[2] Technical University of Denmark, Bygningstorvet, 2800 Kongens Lyngby, Denmark

Abstract. Within dense urban environments, real-world transportation systems are often associated with extraordinary modeling complexity. Where standard analytic methods tend to fail, simulation tools emerge as reliable approaches to study such systems. Despite their versatility, simulation models can prove to be computational burdens, exhibiting prohibitive simulation runtimes. To address this shortcoming, metamodels are used to aid in the simulation modeling process.

In this paper, we propose a directional training scheme, combining both active learning and simulation metamodeling, to address the challenge of exploring the input space, within the context of computationally expensive simulation models. Using a Gaussian Process (GP) as a simulation metamodel, we guide the exploration process towards the identification of specific regions of the input space that trigger a particular simulation output search value of interest defined a priori by the user, saving a significant amount of simulation time in the process.

The results obtained from applying our methodology to an Emergency Medical Service (EMS) simulator, show that it is capable of identifying such important input regions while minimizing the number of simulation runs at the same time, thus making the simulation input space exploration process more efficient.

Keywords: Machine learning · Active learning · Simulation metamodeling · Gaussian Processes

1 Introduction

Real-world urban transportation environments are systems often exhibiting overwhelming complexity and multidimensional dynamism. These intrinsic properties traditionally pose effective and practical constraints when it comes to the modeling process. Simulation tools are usually regarded as the only reliable approach to study such complex systems [22]. However, despite their obvious advantages, simulation models are not exempted from its drawbacks.

A. L. Martins et al. (Eds.): INTSYS 2019, LNICST 310, pp. 32–46, 2020.
https://doi.org/10.1007/978-3-030-38822-5_3

Perhaps the most evident and persistent disadvantage is that when designed with sufficiently high resolution and realism, simulation methods tend to exhibit prohibitive runtimes and massive computational workloads, even considering today's standards. A straightforward solution to this problem is to consider the use of simulation metamodels [11], which are specially conceived to approximate the simulation results and, consequently, the underlying function inevitably defined by the simulation model itself.

Within similar experimental setups, characterized by the lack or expensiveness of data, active learning emerges as a dominant modeling paradigm that tries to address this problem, being particularly popular among the machine learning community. Similarly to the simulation metamodels, the primary goal of active learning is to reduce the computational burden during the learning stage of a given machine learning model. It provides any model the ability to choose the most informative data points from which it learns, making it perform better with less training points and in a more efficient manner [28].

In this work, we propose a directional training scheme that combines the best of both worlds, active learning and simulation metamodeling, to address the challenge of exploring the input space, within the context of computationally expensive simulation models. A Gaussian Process (GP) is employed as a simulation metamodel, guiding the exploration process towards the identification of specific regions of the input space that trigger a particular simulation output search value of interest defined a priori by the user.

Using an Emergency Medical Service simulation model, the results show that the proposed approach can identify such important input regions while minimizing the number of required simulation runs at the same time, thus effectively making the exploration process of the simulation input space computationally more efficient.

2 Background

Simulation metamodeling [12,16,17,21] is quite an old topic among the simulation literature [5]. Its main purpose is to develop and provide approximation models for the simulation results, allowing for a systematic exploration of the functional behavior of complex and time-consuming simulation models in a rather less expensive way.

Simulation models are usually associated with computationally fast and structurally simple functions that map the same input/output domains of the original simulator. Hence, metamodels should reflect both the problem entity under study (e.g., some real-world system) and the simulation model itself. However, the validation degree is closely related to the accuracy requirements, which eventually depend on the metamodeling goals. In [19], four possible primary goals are identified, namely, problem entity understanding, simulation output prediction, optimization, and verification/validation. In this work, however, we are mostly concerned with understanding the real underlying system and with assessing the prediction performance of the metamodel. We assume that the

simulation model of interest is ideally validated, verified, optimized, and thus calibrated concerning the real problem under study.

The simplest applications of simulation metamodeling involved queuing systems with linear models as approximation functions [16]. Due to their simplicity and easy interpretation, GPs are also quite popular as simulation metamodels [6,8,10,18]. Although their application essentially started as deterministic simulation approximators, it was later extended to stochastic ones [3,7,20,29].

Similarly, as simulation metamodels are designed to reduce the computational burden of systematic and exhaustive computer experiments, active learning aims to increase the predictive performance of a given model with a few training points as possible. It does so by providing the model with the ability to actively choose the most informative data points that should be included in the training set, which iteratively expands as the fitting stage evolves.

As seen in [30], any active learning scheme is traditionally comprised of five key players, summarily presented in

$$(\mathcal{L}, \mathcal{U}, \mathcal{M}, \mathcal{O}, \mathcal{Q}).$$

The first two elements represent the labeled and unlabeled data sets, respectively. As active learning is often associated with modeling scenarios where labeled data is scarce or expensive to obtain, the size of $\mathcal{U}$ is oftentimes massively greater than that of $\mathcal{L}$. Next, we have $\mathcal{M}$, which denotes the machine learning model or any other kind of predictive algorithm, followed by $\mathcal{O}$, which represents the labeled instance provider, commonly known as the oracle. The only role of the latter is to provide labeled instances from the ground truth function underlying the process of the system under study. A human annotator traditionally played the role of the oracle. However, it can take several forms, as long as it constitutes a label provider, which trivially includes simulators, among others. Finally, $\mathcal{Q}$ is the query function, which essentially defines how the new data points should be selected from $\mathcal{L}$ to be labeled by $\mathcal{O}$. It commonly encompasses not only search strategies but also criteria to evaluate which are the most informative instances that best increase the performance of $\mathcal{M}$.

Closely associated with the query function is the definition of the stopping criteria. Being an iterative sampling method, active learning must be stopped at some point in time. As pointed out in [28], this point can be identified in two decisive situations: (a) when the cost of obtaining a new labeled point is higher than the model's errors and (b) when the model has reached a performance threshold, from which the addition of new training points will have no or almost no effect.

Due to its Bayesian properties, a GP can be easily implemented to follow an active learning scheme. As its predictions come in the form of fully-defined probability distributions, rather than single point-wise estimates, it accounts for data uncertainty in a quite intuitive way. Assuming that the predictive variance can be considered a proxy for informativeness, the GPs can be used to explore the most informative points within a given simulation input space. These approaches are commonly associated with exploration-exploitation strategies in Bayesian Optimization problems [15,23].

3 Approach

Our approach is based on an active learning scheme built on the top of a simulation metamodeling approach, and it is specially designed to extract relevant information regarding the underlying simulation model under study with as few simulation runs as possible. First, we introduce the GP modeling framework, acting as our simulation metamodel, and then move to the presentation of the proposed approach.

3.1 Gaussian Process

According to [27], a GP is a stochastic process in which any finite set of variables follows a multivariate Gaussian distribution. This collection of random variables is fully characterized by a pair of functions, namely, a mean and a covariance (or kernel) function, respectively represented and defined by $m_f(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$ and $k_f(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m_f(\mathbf{x}))(f(\mathbf{x}') - m_f(\mathbf{x}'))]$, with x and x' being two different D-dimensional input data points. Consequently, a GP is often denoted by $\mathcal{GP}(m_f(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}'))$. When applied to regression problems, the GP framework is known as Kriging [9]. Such denomination has its origins in the geostatistics field.

The GP framework places a prior over functions. Within a regression setup, this means that the functional relationship between the dependent variable $\mathbf{x}$ and the independent variable y is assumed to follow a GP. Formally put, we have $y = f(\mathbf{x}) + \epsilon$, where $f(\mathbf{x}) \sim \mathcal{GP}(m_f(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}'))$ and $\epsilon \sim \mathcal{N}(0, \sigma^2)$. Thus, the values of the signal function f are represented by the random variables comprising the GP and defined over an high-dimensional feature space, for example, $\mathbb{R}^D$.

For prediction purposes, the conditional distribution of a new test point $\mathbf{x}_*$ is given by

$$f_*|X, \mathbf{y}, \mathbf{x}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, cov(\mathbf{f}_*)),$$

with

$$\begin{aligned} \bar{\mathbf{f}}_* \triangleq &\mathbb{E}[f_*|X, \mathbf{y}, \mathbf{x}_*] = k_{f*}^\top [K_y]^{-1} \mathbf{y}, \\ cov(\mathbf{f}_*) = &\mathbb{V}[f_*] = k_{f**} - k_{f*}^\top [K_y]^{-1} k_{f*}, \end{aligned}$$

where $k_{f*} = k_f(X, \mathbf{x}_*)$, $k_{f**} = k_f(\mathbf{x}_*, \mathbf{x}_*)$, and $(X, \mathbf{y})$ representing the training data set. Here, notice that instead of a point-wise prediction, each GP prediction is associated with a completely defined Gaussian distribution, allowing it for a effective Bayesian treatment of the uncertainty not only present in the training data but also regarding its the predictions themselves.

Most of the functions used to define GPs have a set of free parameters (also called hyper-parameters), allowing for their optimization with respect to the training data, commonly via maximum likelihood estimation. However, mainly for simplicity reasons, the mean function can be set to zero for the majority of the applications. On the contrary, the covariance function plays a vital role in the modeling performance of the GP.

In this work, we use the Squared-Exponential with Automatic Relevance Determination (SE-ARD) function as the GP's kernel, generally defined as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\frac{1}{2}(\mathbf{x} - \mathbf{x}')^\top M(\mathbf{x} - \mathbf{x}')\right),$$

where σ_f^2 corresponds to the variance of the underlying signal function f, $M = diag(\boldsymbol{\sigma})^{-2}$ and $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_D]^\top$ is a positive real-valued vector. Each diagonal element of M represents the characteristic length-scale for each of the D input dimensions. These length-scales weight the importance of the each dimension during the inference process. The standard version of the SE version with isotropic distance measure can be obtained by setting $\sigma_1 = \sigma_2 = \ldots = \sigma_D$. This function is one of the most widely used kernel functions, not only for GPs but also for other well-known kernel-based machines, such as the Support Vector Machines (SVMs).

3.2 Directional Grid-Based Search

The modeling approach developed in this work is summarized in Algorithm 1, combining elements of both active learning and simulation metamodelling strategies.

Before running the algorithm, three entities must be defined a priori, namely, the search value (sV), the initial training grid ($trGrid$), and the search grid ($sGrid$). Whereas sV and $sGrid$ are fixed, $trGrid$ evolves over time. The basic idea of the presented methodology is to use a GP to guide the expansion process of $trGrid$ towards the identification of simulation input regions that trigger simulation output values close to sV. In other words, this methodology allows the user to search for sets of input values whose simulation results are close to a pre-specified output of interest, by conducting sequential predictions over $sGrid$. For the sake of simplicity, we focus on the two-dimensional case. An illustration of the grid-based training unit used in this work is depicted in Fig. 1(a).

We call it directional since it steers the simulation requests (or runs) exclusively in the direction of those input values that are more likely to assume output values similar to sV. On the other hand, it is grid-based as it comprises a series of iterative training grids used to locally approximate the simulation results within the neighborhoods of the search value. By proceeding in such a directional way, we can minimize the number of required simulation runs, therefore making the input space exploration process faster and computationally more efficient. This is particularly useful for those simulation models that exhibit prohibitive simulation runtimes and workloads.

After sV, $trGrid$ and $sGrid$ are set, the algorithm is ready to start. It does so by obtaining, via simulation requests, the simulation output results ($simR$) corresponding to the input values comprised in $trGrid$. Note that in this firstiteration, $trGrid$ matches the established training grid-based unit exactly,

Algorithm 1. Directional Grid-based Active Learning pseudo-algorithm.

1: **Inputs**: sV, $trGrid$, $sGrid$
2: **Repeat**
3: Obtain $simR$ corresponding to the input values in $trGrid$
4: Fit a GP to the training data set $(trGrid, simR)$
5: Use the fitted GP to make predictions over $sGrid$
6: Define training sub-grids according to the predicted values closest to sV
7: Expand $trGrid$ with the newly defined grids
8: Compute AAD between predictions and sV
9: **Until** AAD stabilizes
10: **Output**: $trGrid$

which in turn is defined over the simulation input space in which we believe that sV is triggered. Then, a GP is fitted to the training set $(trGrid, simR)$. The prediction stage follows this initial step. Here, the obtained GP is used to make predictions over $sGrid$. Remember that the latter does not contain any simulation result, only unlabeled instances.

Additionally, $sGrid$ should be sufficiently dense so that the GP can populate it with the associated predictions with enough detail. This does not constitute a computational hindrance since the GP, after training, is rather fast when making predictions. This is often the case for most machine learning frameworks. Afterward, the predictions obtained from the previous step are used to explore the simulation input space, especially to locate those that are value-wise closer to sV. At this point, new grids are defined, and $sGrid$ is expanded. Note that these newly defined grids are essentially sub-grids within the initial one with the same structure, as previously seen in Fig. 1(a). Lastly, we compute the Average Absolute Difference (AAD) between the GP predictions and sV.

The process is repeated until AAD shows no significant variation from iteration to iteration. We are not interested if the GP approximation is below or above the search value, but rather how close it is in absolute terms. We compute the average so that we have an indicator of the GP's overall fitting performance. As we expand the training set with data points whose simulation output values are successively closer to sV, it is expected that AAD decreases over time, eventually reaching a certain threshold. This threshold represents the point from which the GP can no longer improve its predictions, by merely adding more data points to the training set. Ultimately, the main goal of the algorithm is to provide a final mesh grid that delimits the input space region of interest that triggers explicitly the value we are searching for. In Fig. 1(b), a graphical depiction of the proposed approach is shown.

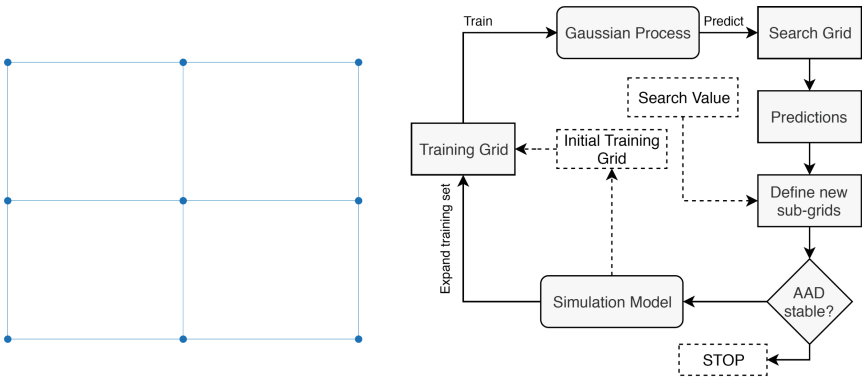


Fig. 1. (a) Grid-based training unit and (b) flow diagram of the proposed active learning metamodeling methodology.

4 Experiments

In this section, we illustrate the proposed approach using an Emergency Medical Service simulator. As for the software used, we implemented our approach using the free and open-source Matlab toolbox Gaussian Processes for Machine Learning (GPML), developed and maintained by [27]. As mentioned earlier, we choose the SE-ARD function as the GP kernel and set the GP mean as the average of the simulation output values within the training set.

4.1 EMS Simulation Model

According to a publication released by the United Nations, more than half of the world's population in 2016 lived in urbanized areas. By the year 2030, it is expected that around 60% will globally live within urban settlements, and 33% will live in cities with at least half a million inhabitants [25]. This migration from the countryside places unprecedented pressure on the existing urban infrastructures, consequently leading to further and unpredictable urban transformations. The impact of these transformations, and thus the future of our cities, are mostly dependent on decisions that are taken in the present day to prepare and plan for this inevitable urban growth [24].

Emergency Medical Services (EMS) play a particularly vital role within the exponential growth of today's cities. This kind of service ensures the safety and well-being of its citizens by promptly dispatching emergency vehicles to the locations of the life-threatening events, generally relying on public emergency phone lines. The quality of service is highly dependent on the information that the service operator obtains from the caller, the severity of the case and the accuracy of its medical needs assessment, as well as, obviously, on the final decision taken

and the corresponding actual response. Especially within high-density urban areas, EMSs are forcibly constrained by the city dynamics. Daily traffic and population changes are two fundamental forces that directly affect the outcome performance of medical services [2]. Hence, the planning of EMS is of utmost importance. Simulation modeling is a standard tool to design and explore the response of EMS since the evaluation of policy decisions or operational solutions are often deemed unfeasible in the real-world [4].

In this work, we used the agent-based EMS simulator developed by [1]. The underlying simulation model implements the vehicles' allocation and dispatching according to the closest dispatching rule [13,14,31], i.e., the emergency event is assigned by the closest idle vehicle. Moreover, the simulator encompasses three fundamental input dimensions, namely, location change probability, traffic error, and vehicle station locations. The latter also includes the number of vehicles per station. Whereas the latter is easy to interpret, the former two might not be so obvious. The location change probability is designed to induce a certain level of randomness to the emergency call's spatial distribution so that it differs significantly from the available historical data. On the other hand, the traffic error encompasses the uncertainty present in the difference between the predicted and the real traffic conditions.

Two main outputs are provided by this simulator, covering both the EMS vehicles' response times and the victims' survivability, which serve as performance metrics of the EMS. These are respectively represented by the average survival rate and the average response time. Whereas the latter trivially encodes the outcome of the emergency event, the former is defined by the time difference between the emergency call and the medical team's arrival.

The simulation model is configured to emulate an emergency system with real data from the city of Porto, Portugal, with 90 emergency vehicle locations. Furthermore, to present our methodology, we only consider the location change probability and the traffic error as inputs, and the average response time as a system performance metric. The first two assume real values in the interval $[0, 1]$. As for the response times, these can assume any positive value. The emergency station locations and their corresponding number of vehicles were maintained constant.

4.2 Results

Following the observations made in [26], which discusses the recommended guideline of a maximum response time of 8 min (480 s), we apply our methodology in order to search for the set of input simulation points that explicitly trigger this threshold within the mentioned EMS simulation model.

As previously seen in Sect. 3.2, several inputs for the proposed algorithm must be defined a priori. This led us to define 480 as our search value. Thus we have $sV = 480$. Next, we fixed $sGrid$ as a mesh grid of 10000 (unlabeled)

points scattered uniformly in $[0, 1] \times [0, 1]$, as this is the domain of the simulation input space under study. Lastly, the initial training grid, *trGrid*, corresponds to the first nine-point training unit, as depicted in Fig. 1(a), whose vertices exactly match those of the input domain. Recovering the notation associated to active learning presented in Sect. 2, we now have that $\mathcal{L} = trGrid$ and $\mathcal{U} = sGrid$. The GP is the oracle $\mathcal{O}$, and the query function $\mathcal{Q}$ is essentially represented by the way we select the new training grids that are added to the expanding training set.

The results are presented in Figs. 2, 3, 4 and 5. As mentioned earlier, the algorithm starts by requesting the simulation model to label the instances present in *sGrid*. Then, a GP is fitted to these simulation points. In Fig. 2(a), we can see the first GP approximation, here depicted by a three-dimensional surface defined over the two-dimensional simulation input space $[0, 1]^2$ using the unlabeled observations present in *sGrid*. After, the algorithm searches for the best candidate sub-grids within the initial grid. Such grids contain the GP predictions that are most similar to the search vale sV. This can observed in Fig. 3(b) and (c). The algorithm has detected that the most likely simulation input region to trigger sV is contained somewhere within $[0, 1] \times [0, 0.5]$. The AAD is then computed, and the algorithm proceeds.

The GP fitting, as well as the expansion of the training set, continues sequentially by sub-dividing the previous' iteration training grid into smaller and finer replicas of itself. In Fig. 3, we can clearly see the sequence of these grids. Due to paper space constraints, we only present part of the results, skipping iterations 7 to 13. Eventually, the process stops when the simple addition of new points does not alter AAD. Figure 5(a) shows that the algorithm took 15 iterations to stop. As a result of the continuous expansion of the training set, it is expected that the predictive variance associated with the data points lying near the input space region of interest tends to decrease. This decrease is depicted in Fig. 5. More than expected, this is the ultimate goal of the proposed methodology. Observe that the latter mentioned region, roughly approximated by the proposed grid-based training structure, gets narrower as the active learning process advances. Figure 4 clearly shows this evolution. Here, we depict the absolute difference between the GP predictions and the search value. Darker tones imply small differences. In the end, and by combining Figs. 3(h) and 4(h), we can observe that the points that are more likely to trigger the search value of interest, are concentrated in the input simulation region roughly by the grid contained in $[0, 1] \times [0.625, 0.750]$.

Note that we showed little concern regarding the prediction performance of the final GP approximation. Its main goal, more than being a reasonably good approximation of the underlying function defined by the simulation model itself, is to guide the active learning towards the most informative data points concerning the given output search value. As a consequent, this GP-based metamodeling approach ultimately leads us to the discovery of relevant input regions within the simulation space in a rather expeditious manner.

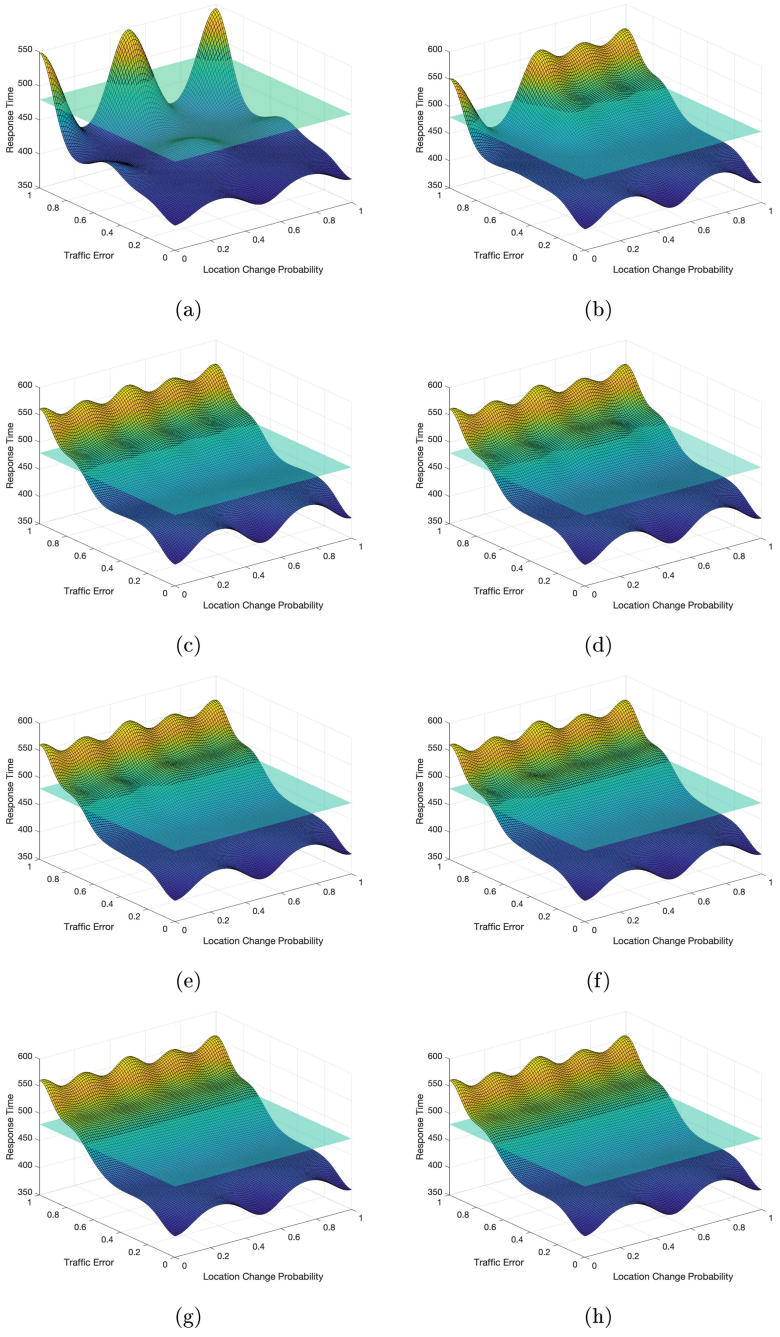


Fig. 2. Iterative GP surface approximations. Panels (a)–(f) and (g)–(h) correspond to iterations 1–6 and 14–15, respectively. The flat horizontal surface is located at z = 480.

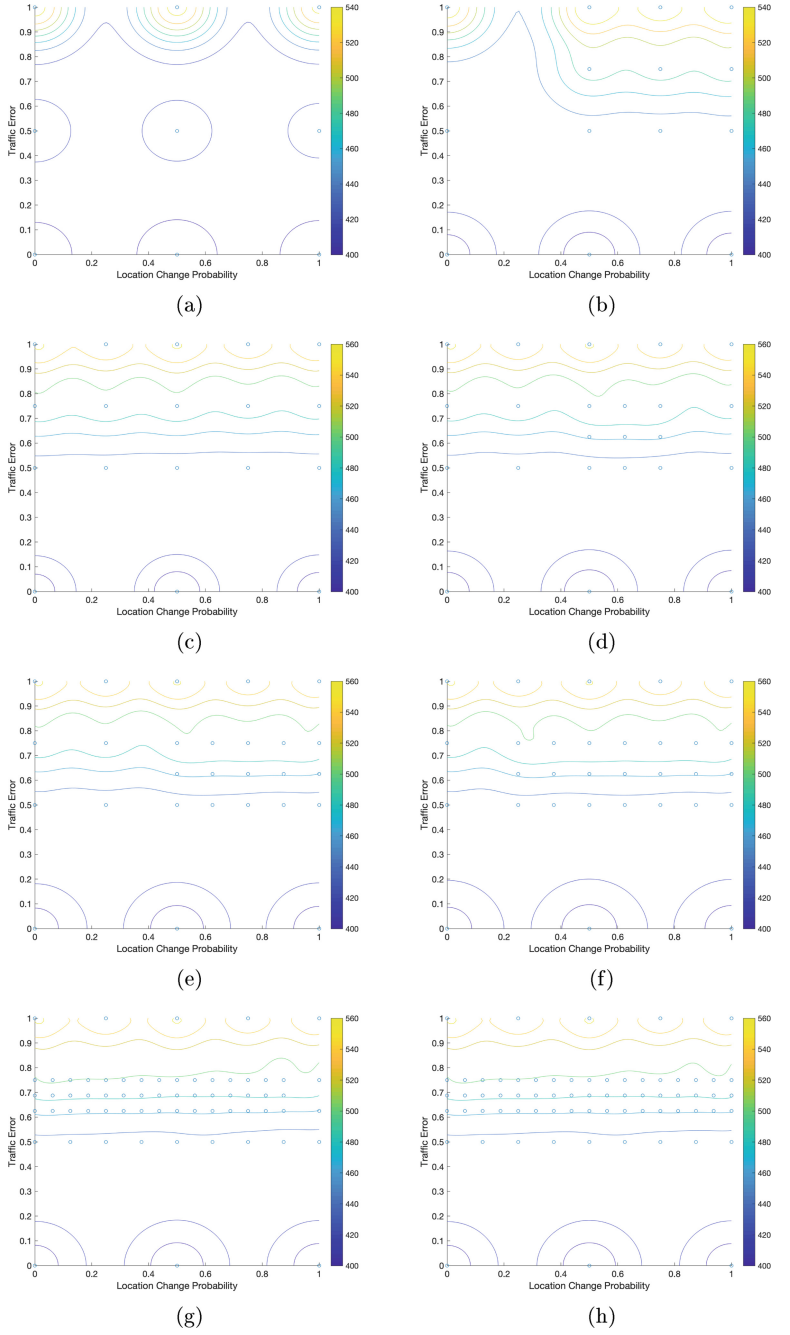


Fig. 3. Iterative training grids and associated GP surface approximation contours. Panels (a)–(f) and (g)–(h) correspond to iterations 1–6 and 14–15, respectively. The flat horizontal surface is located at z = 480.

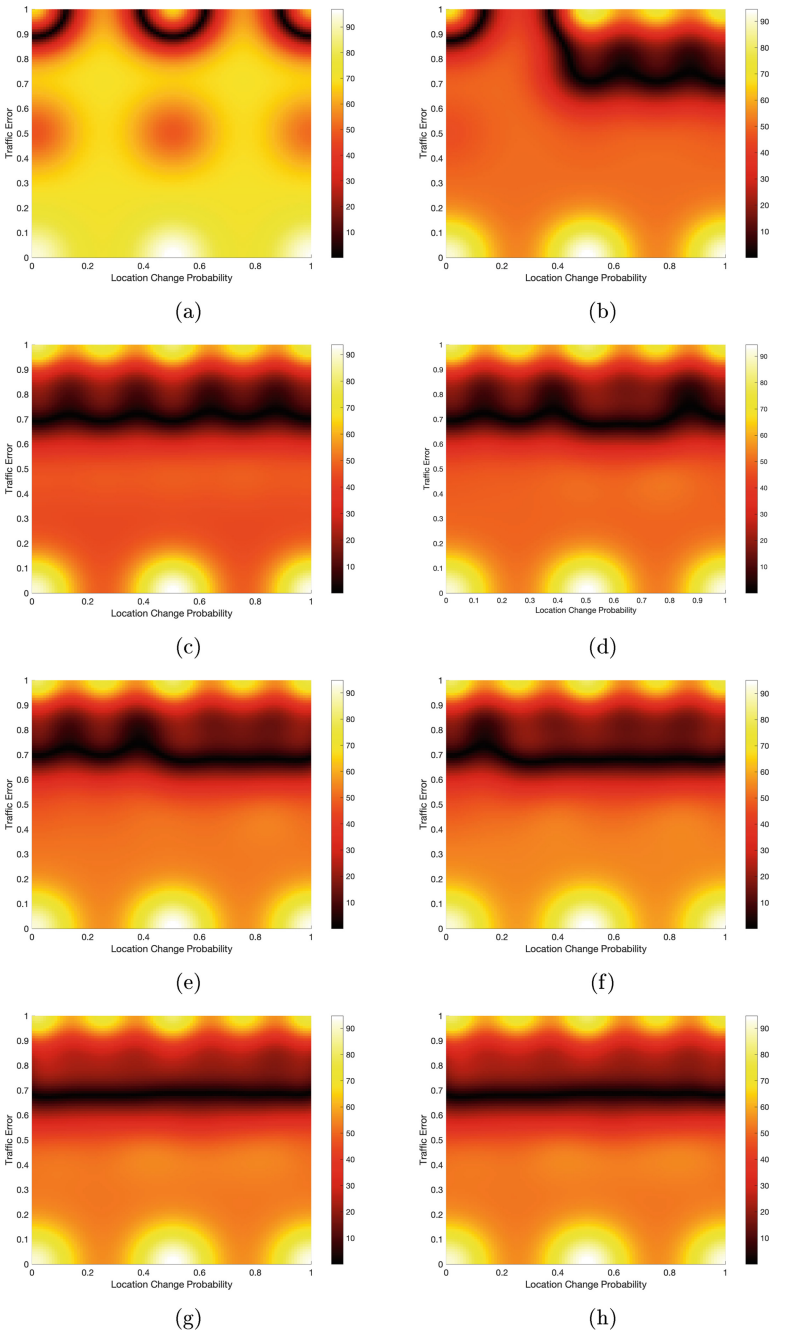
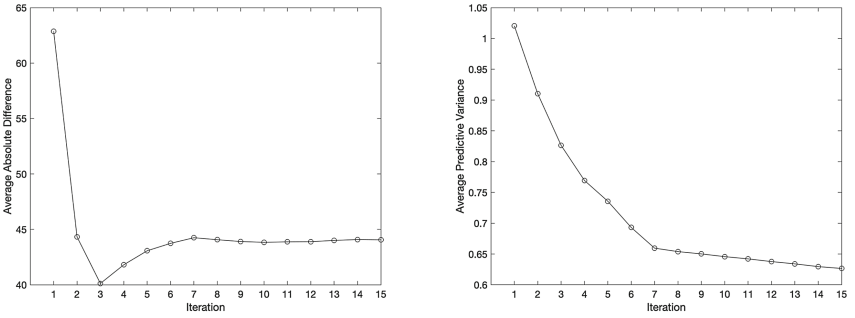


Fig. 4. Absolute difference between the GP surface approximations and the given search value (480). Panels (a)–(f) and (g)–(h) correspond to iterations 1–6 and 14–15, respectively. The flat horizontal surface is located at z = 480.



(a) (b)

Fig. 5. Evolution of (a) Average Absolute Difference and (b) Average Predictive Variance

5 Conclusion and Future Work

In this paper, we presented a methodology that combines active learning and simulation metamodeling to address the challenge of exploring the input space of any simulation model, particularly those that exhibit great runtimes and computational workloads. Starting from a simple training grid, and guided by a GP acting as a simulation metamodel, the proposed approach uses active learning to build an increasing mesh grid of training points iteratively. This mesh grid comprises a set of sequential subgrids that are steered towards specific regions of the simulation input space that trigger a specific simulation output value, defined a priori by the user. This directional scheme used to train the associated GP-based metamodel makes the exploration process more efficient, as only those input values whose simulation results are more likely to be closer to the search value of interest are considered to be included in the training set.

This work can be improved within several research directions. A straightforward expansion is to generalize our approach from a single search value to multiple search values or even sets of intervals. Additionally, it would be interesting to consider a multi-output approach, where multiple output performance measures could be explored simultaneously. This will not bring a significant challenge from the computational point-of-view, as, in principle, the output variables and associated metrics are always available throughout the entire simulation experiment workflow. Closely related to this, multiple output regression should be embraced in the future so that possible correlations among the output variables can be captured.

Increasing the dimensionality of the proposed approach should also be considered. Along with it, further numerical experiments and graphical representation challenges will emerge. The key contribution of our work is the identification of important regions within the simulation input space, which inherently implies graphical elements. Therefore, new ways of presenting the results, especially for hyper-dimensional spaces, must be explored and developed.

The grid-based unit used to train the simulation metamodel should be revised in the future. Maintaining the concept of grid, more appropriate geometric forms,

rather than square-based ones, should be taken into account in accordance with the characteristics of the simulation input space. As this training unit represents how we run the simulation experiments, new sampling strategies should be adopted in order to achieve improved fitting performance. For example, we plan to combine the current approach with statistical designs for computer experiments, such as the widely known Latin hypercube. This kind of scheme provides a systematic sampling framework that ensures the statistical significance as well as the prediction performance of the obtained simulation metamodels.

In our particular case of application, the simulation model did not exhibit very complex output behavior. Nevertheless, the results show the potential of this type of active learning metamodeling approach when searching for specific input regions. Hence, an important step is to apply our approach against new simulation models with an increased degree of complexity, especially with regards to the model's output behavior. This would not only provide further validation for the current work, but it would also stimulate more discussion on the topic.

Acknowledgments. The support of FCT (Portuguese national funding agency for science, research, and technology) under grant No. PD/BD/128047/2016 is greatly acknowledged. This work has additionally received funding from the People Programme (Marie Curie Actions) of 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. Amorim, M., Ferreira, S., Couto, A.: Emergency medical service response: analyzing vehicle dispatching rules. Trans. Res. Rec.: J. Transp. Res. Board **2672**(32), 10–21 (2018). https://journals.sagepub.com/doi/abs/10.1177/0361198118781645. https://journals.sagepub.com/toc/trra/2672/32
2. Amorim, M., Ferreira, S., Couto, A.: Corrigendum to how do traffic and demand daily changes define urban emergency medical service (uEMS) strategic decisions?: A multi-period survival approach. J. Transp. Health **12**, 60–74 (2019). p. 100570
3. Ankenman, B., Nelson, B.L., Staum, J.: Stochastic kriging for simulation metamodeling. Oper. Res. **58**(2), 371–382 (2010)
4. Antunes, F., Amorim, M., Pereira, F., Ribeiro, B.: Active learning metamodeling for policy analysis: application to an emergency medical service simulator. Simul. Model. Pract. Theory **97**, 101947 (2019)
5. Barton, R.R.: Simulation metamodels. In: Simulation Conference Proceedings, Winter, vol. 1, pp. 167–174. IEEE (1998)
6. Boukouvalas, A.: Emulation of random output simulators. Ph.D. thesis, Aston University (2010)
7. Boukouvalas, A., Cornford, D., Singer, A.: Managing uncertainty in complex stochastic models: design and emulation of a rabies model. In: 6th St. Petersburg Workshop on Simulation, pp. 839–841 (2009)
8. Chen, T., Hadinoto, K., Yan, W., Ma, Y.: Efficient meta-modelling of complex process simulations with time-space-dependent outputs. Comput. Chem. Eng. **35**(3), 502–509 (2011)
9. Chilès, J.-P., Desassis, N.: Fifty years of Kriging. In: Daya Sagar, B.S., Cheng, Q., Agterberg, F. (eds.) Handbook of Mathematical Geosciences, pp. 589–612. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-78999-6_29

10. Conti, S., O'Hagan, A.: Bayesian emulation of complex multi-output and dynamic computer models. J. Stat. Plan. Infer. **140**(3), 640–651 (2010)
11. Friedman, L.W.: The Simulation Metamodel. Springer, Heidelberg (2012)
12. Friedman, L.W., Pressman, I.: The metamodel in simulation analysis: can it be trusted? J. Oper. Res. Soc. **39**(10), 939–948 (1988)
13. Haghani, A., Yang, S.: Real-time emergency response fleet deployment: concepts, systems, simulation & case studies. In: Zeimpekis, V., Tarantilis, C.D., Giaglis, G.M., Minis, I. (eds.) Dynamic Fleet Management. Operations Research/Computer Science Interfaces Series, vol. 38, pp. 133–162. Springer, Boston (2007). https://doi.org/10.1007/978-0-387-71722-7_7
14. Jagtenberg, C., van den Berg, P., van der Mei, R.: Benchmarking online dispatch algorithms for emergency medical services. Eur. J. Oper. Res. **258**(2), 715–725 (2017)
15. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient global optimization of expensive black-box functions. J. Global Optim. **13**(4), 455–492 (1998)
16. Kleijnen, J.P.: A comment on blanning's "metamodel for sensitivity analysis: the regression metamodel in simulation". Interfaces **5**(3), 21–23 (1975)
17. Kleijnen, J.P.: Regression metamodels for generalizing simulation results. IEEE Trans. Syst. Man Cybern. **9**, 93–96 (1979)
18. Kleijnen, J.P.: Kriging metamodeling in simulation: a review. Eur. J. Oper. Res. **192**(3), 707–716 (2009)
19. Kleijnen, J.P., Van Beers, W.C.: Application-driven sequential designs for simulation experiments: Kriging metamodelling. J. Oper. Res. Soc. **55**(8), 876–883 (2004)
20. Kleijnen, J.P., Van Beers, W.C.: Robustness of Kriging when interpolating in random simulation with heterogeneous variances: some experiments. Eur. J. Oper. Res. **165**(3), 826–834 (2005)
21. Kleijnen, J.: Model behaviour: regression metamodel summarization. Encycl. Syst. Control **5**, 3024–3030 (1987)
22. Law, A.M.: Simulation Modeling and Analysis, 5th edn. McGraw-Hill Higher Education, New York City (2015)
23. Ling, C.K., Low, K.H., Jaillet, P.: Gaussian process planning with Lipschitz continuous reward functions: towards unifying Bayesian optimization, active learning, and beyond. In: AAAI, pp. 1860–1866 (2016)
24. Martine, G., Marshall, A., et al.: State of world population 2007: unleashing the potential of urban growth. In: State of World Population 2007: Unleashing the Potential of Urban Growth. UNFPA (2007)
25. United Nations: The World's Cities in 2016, Data Booklet, ST/ESA/SER.A/392. Department of Economic and Social Affairs, Population Division (2016)
26. Pons, P.T., Markovchick, V.J.: Eight minutes or less: does the ambulance response time guideline impact trauma patient outcome? J. Emerg. Med. **23**(1), 43–48 (2002)
27. Rasmussen, C.E., Williams, C.: Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, Cambridge (2006)
28. Settles, B.: Active Learning: Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Clay Pool, Long Island (2012)
29. Van Beers, W.C., Kleijnen, J.P.C.: Kriging for interpolation in random simulation. J. Oper. Res. Soc. **54**(3), 255–262 (2003)
30. Wang, X., Zhai, J.: Learning from Uncertainty. CRC Press, Boca Raton (2016)
31. Yang, S., Hamedi, M., Haghani, A.: Online dispatching and routing model for emergency vehicles with area coverage constraints. Transp. Res. Rec. **1923**(1), 1–8 (2005)