Active Learning Metamodels for ATM Simulation Modeling

Christoffer Riis
DTU Management
Technical University of Denmark
Kgs. Lyngby, Denmark
chrrii@dtu.dk

Francisco Antunes
DTU Management
Technical University of Denmark
Kgs. Lyngby, Denmark
franant@dtu.dk

Gérald Gurtner
School of Architecture and Cities
University of Westminster
London, England, UK
g.gurtner@westminster.ac.uk

Francisco Camara Pereira
DTU Management
Technical University of Denmark
Kgs. Lyngby, Denmark
camara@dtu.dk

Luis Delgado
School of Architecture and Cities
University of Westminster
London, England, UK
l.delgado@westminster.ac.uk

Carlos Azevedo
DTU Management
Technical University of Denmark
Kgs. Lyngby, Denmark
climaz@dtu.dk

***Abstract*—Transportation systems are particularly prone to exhibiting overwhelming complexity on account of the numerous involved variables and their interrelationships, unknown stochastic phenomena, and ultimately human behavior. Simulation approaches are commonly used tools to describe and study such intricate real-world systems. Despite their obvious advantages, simulation models can still end up being quite complex themselves. The field of Air Traffic Management (ATM) modeling is no stranger to such concerns, as it traditionally involves laborious and systematic analyses built upon computationally heavy simulation models. This rather frequent shortcoming can be addressed by employing simulation metamodels combined with active learning strategies to approximate the input-output mappings inherently defined by the simulation models in an efficient way.**

In this work, we propose an exploration framework that integrates active learning and simulation metamodeling in a single unified approach to address recurrent computational bottlenecks typically associated with intense performance impact assessments within the field of ATM. Our methodology is designed to systematically explore the simulation input space in an efficient and self-guided manner, ultimately providing ATM practitioners with meaningful insights concerning the simulation models under study. Using a fully developed state-of-the-art ATM simulator and employing a Gaussian Process as a metamodel, we show that active learning is indeed capable of enhancing both the modeling and performances of simulation metamodeling by strategically avoiding redundant computer experiments and predicting simulation outputs values.

***Keywords*—Active Learning, Simulation Metamodeling, Air Traffic Management Simulation Modeling, Gaussian Processes**

I. Introduction

Transportation systems are inherently characterized by overwhelming complexity and dynamism. In addition to involving numerous variables and their corresponding relationships, these systems often exhibit stochastic behavior and unknown random phenomena that are virtually impossible to encode into closed-form and tractable analytic expressions [1]. As a result simulation approaches are traditionally employed as the de facto modeling tools to characterize and study such systems and ultimately assess their performance. Both present and future transport systems can be successfully studied via simulation, additionally allowing for the assessment of current and planned interventions [2]. Due to its intrinsically exploratory nature supported by computer-based representations, simulation proves to be highly advantageous for planning and policy analysis [3].

Simulation modeling has a long tradition within the transportation community [2], in which many simulation-based analyses mostly rely on manual what-if approaches in an effort to characterize and ultimately discretize uncertainty into a finite and intelligible set of possible future system states. Since planning for the future inevitably involves risk and uncertainty, the use of scenarios, and associated solutions, as descriptive narratives of potential future states of the system under study is widely used in planning and robust decision-making processes [4].

The focus of this work lies in the field of Air Traffic Management (ATM), whose main objectives are to ensure the safety and efficiency of air traffic flows [5]. An ATM system encompasses an overwhelming number of elements and variables, from those related to the aircraft, their control, and associated technologies, to the interdependent policies and regulations, stakeholder and market conditions. The emergent behavior resulting from the interaction between all these elements makes ATM systems inherently hard to model [6]. Due to their complexity, ATM systems are typically approached via simulation modeling, as it constitutes the only reliable method capable of tackling its idiosyncrasies. Both micro and macroscopic models are commonly used. Whereas the former primarily focuses on behavioral foundations and individual entities at the disaggregated level, the latter tends to be employed for strategic decision-making due to their parsimonious and high-level properties.

Despite the indisputable advantages of the above-mentioned simulation modeling approaches, they do not come without their drawbacks: simulation models themselves can also become complex and thus computationally expensive to run. It is expected that with more detailed and realistic models, the potential for prohibitive simulation running times is highly likely, which can render further analysis unfeasible and the exploration of the simulator's behavior difficult to attain in a systematic manner. One of the most common challenging problems in terms of ATM modeling is the assessment of the performance impacts of new system-wide level solutions, as recently highlighted in [7].

In this paper, we propose an exploration framework that effectively integrates active learning and simulation metamodeling to address the limitations posed by computationally expensive simulation models in the context of ATM. On the one hand, active learning is a modeling paradigm that aims at attaining high predictive performance with few data points, while on the other hand, simulation metamodels are designed to approximate the input-output mappings inherently defined by the simulation models themselves in an effort to mimic the simulators' output behavior. Hence, our aim is to combine the best of both worlds into an integrated approach. To the best of our knowledge, such kind of solution is scarcely applied in transportation research and is nonexistent in the ATM field. Using a fully developed state of the art agent-based ATM simulator and employing a Gaussian Process as a metamodel, we show that active learning is indeed capable of enhancing both the modeling and exploration performances of simulation metamodeling by strategically avoiding redundant computer experiments and predicting, within a certain degree of confidence, simulation outputs values. Additionally, we show the computational advantages of using our metamodel by benchmarking the approach against traditional exploration of a simulator.

II. Background

A. Gaussian Processes

The Gaussian Processes (GPs) [8] are a widely applied tool within probability, statistics and machine learning fields. Their nonlinear and nonparametric abilities make the GPs a powerful modeling tool in a wide range of regression and classification problems. Moreover, its Bayesian properties enable the quantification of the uncertainty present in its own predictions and in the data itself.

Within any regression setting, a generic D-dimensional data set by $\mathcal{S} = \{(\mathbf{x}_i, y_i) | i = 1, \ldots, n\}$, where $\mathbf{x}_i \in \mathbb{R}^D$ is an input vector, y the output (or target) variable, n the number of observations, and D the size of the input feature space, is first considered. After aggregating the n input vectors, we obtain the $(D \times n)$ design matrix X. In a similar fashion for the n targets, $\mathbf{y}$ is also obtained. The original data set can now be rewritten as $(X, \mathbf{y}) \subseteq \mathbb{R}^{D\times(n+1)}$. The main objective of any regression approach is to infer the functional dependency between the input and output variables or, in other words, the conditional distribution of $\mathbf{y}$ given X, $p(\mathbf{y}|X)$.

As described by [8], a GP is a collection of stochastic variables $f = (f_t, t \in \mathcal{T})$, where any finite set of which jointly follows a Multivariate Gaussian Distribution (MGD), and $\mathcal{T}$ is a set of indices. In this sense, a GP can be trivially seen as a straightforward generalization of a MGD. At the core of any GP lies a pair of functions, namely, the mean and the covariance (also called the *kernel*), respectively denoted by $m_f(\mathbf{x})$ and $k_f(\mathbf{x}, \mathbf{x}')$, where $\mathbf{x}'$ is another input vector. Hence, a GP can be simply denoted as $\mathcal{GP}(m_f(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}'))$, where $m_f(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})]$, $k_f(\mathbf{x}, \mathbf{x}') = Cov(f(\mathbf{x}), f(\mathbf{x}')) = \mathbb{E}[(f(\mathbf{x}) - m_f(\mathbf{x}))(f(\mathbf{x}') - m_f(\mathbf{x}'))]$. In the modeling context previously mentioned, the GP-based regression approach assumes that the relationship between the involved variables is described by a GP, i.e., $y = f(\mathbf{x}) + \epsilon$, where $\epsilon \sim \mathcal{N}(0, \sigma^2)$ and $f(\mathbf{x}) \sim \mathcal{GP}(m_f(\mathbf{x}), k_f(\mathbf{x}, \mathbf{x}'))$.

The kernel has a critical role in determining the overall properties and behavior of the resulting GP, as it encapsulates the similarity between the points and its generalization capabilities. The kernels typically have a certain number of free parameters (hyperparameters of the GP) which can be estimated by minimizing the negative marginal likelihood subjected to the training data. The Squared Exponential (SE) is a common choice for the GPs' kernel. It is generically 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 $\boldsymbol{\sigma} = [\sigma_1, \sigma_2, \ldots, \sigma_D]^\top$. Here, f refers to the function intrinsically defined by the simulation model itself, which we aim to approximate via a GP.

After the optimal hyperparameters of the kernel are obtained, the posterior distribution for a single test point $\mathbf{x}_*$ is given by $\mathbf{f}_* | X, \mathbf{y}, \mathbf{x}_* \sim \mathcal{N}(\bar{\mathbf{f}}_*, Cov(\mathbf{f}_*))$, with $\bar{\mathbf{f}}_* \triangleq \mathbb{E}[f_* | X, \mathbf{y}, \mathbf{x}_*] = k_{f*}^\top [K_y]^{-1}\mathbf{y}$, and $Cov(\mathbf{f}_*) = \mathbb{V}[f_*] = k_{f**} - k_{f*}^\top [K_y]^{-1} k_{f*}$, where $k_{f*} = k_f(X, \mathbf{x}_*)$, $k_{f**} = k_f(\mathbf{x}_*, \mathbf{x}_*)$, and K_y is the kernel for noisy observations. We see that each GP prediction comes in the form of Gaussian distributions with specific mean and variance. Thus, each prediction corresponds to a whole range of values weighted by a well-defined probability density function. Often, $\bar{\mathbf{f}}_*$ and $Cov(\mathbf{f}_*)$ are deemed the predictive mean and variance of the GP, respectively.

Through Bayesian formalism, the GP framework is able to characterize the variance within its own predictions. In fact, the predictive variance is a critical component in GP modeling, and it can be used to maximize information acquisition if regarded as a measure of uncertainty. Note that any given GP prediction can be considered uncertain if its associated predictive variance is high. In other words, the higher the variance is, the more scattered the predictive distribution becomes around the mean value, consequently increasing the range of the most likely values to be predicted. Conversely, low variances imply narrower distributions concentrating most of the probability density in the vicinity of the expected value.

B. Active Learning

Active learning consists of an iterative learning approach that aims to attain high prediction performance with as few training samples as possible. This is achieved by designing into the underlying learning process the ability to - according to a certain criterion - actively select the data points from which it learns, proving to be particularly useful for modeling tasks where labeled data is difficult to obtain [9].

The general idea of active learning is to sequentially select the most informative data points that simultaneously boost the model training efficiency and improve its predictive performance. Hence, it is essentially focused on improving the overall model prediction performance, as well as on controlling the costs associated with acquiring new labeled instances. The entire learning process is guided by a label provider, also called the oracle, whose task is to provide labeled data instances which are then successively incorporated into the expanding training data set. Hence, a core assumption of active learning is that a label provider must be permanently accessible to be queried on-demand by the learning algorithm or model. The most important aspect of the oracle is its ability to successfully and consistently generate labeled instances from the ground truth function inherently defining the process under study.

As elegantly presented in [10], an arbitrary active learning system encompasses five fundamental entities, summarized in the following quintuple $(\mathcal{L}, \mathcal{U}, \mathcal{M}, \mathcal{O}, \mathcal{Q})$. First, $\mathcal{L}$ is the labeled training set, whereas the set of unlabeled data points is represented by $\mathcal{U}$. Generally, we have that the number of unlabeled points is significantly higher than the labeled ones, formally enclosing the earlier mentioned scenarios characterized by shortages of labeled data. The machine learning model is represented by $\mathcal{M}$, whereas the oracle by $\mathcal{O}$. Finally, $\mathcal{Q}$ is the query function that encodes the strategies and criteria for finding and selecting the most informative instances from $\mathcal{U}$ to be added to $\mathcal{L}$.

Different querying strategies can be encoded into $\mathcal{Q}$, essentially representing distinct approaches to the underlying modeling problem. [9] lists several query frameworks, such as uncertainty sampling, query-by-committee, expected model change and error reduction, variance reduction, or density-weighted methods. In sum, most of these frameworks rely either on measures of informativeness or on the identification of uncertainty regions. Whereas the former addresses the querying problem by measuring the potential degree of information contained within an unobserved instance, the latter explicitly defines regions of uncertainty where the model is less confident with respect to its own predictions. Moreover, and due to its inner iterative nature, active learning schemes must be stopped at a certain point in time. Regardless of the details of its definition, it is generally expected that the stated stopping criteria should take into account the constant trade-off between model performance and the cost of acquiring new labeled points. It is imperative to be able to identify those situations in which the model is reaching its theoretical performance threshold, that is to say, when the addition of new training points will potentially have a negligible effect on its performance improvement.

It is common to focus on a single output and sample one data point in each iteration since that allows for incorporating the information from the queried data point to decide on which data point to query next [11], but sometimes it is advantageous to query multiple data points at once. When creating a metamodel for a slow simulator, the computational time of the simulator is an important bottleneck to be aware of, but fortunately, most simulators can be either called in parallel or distributed across different computers. This gives rise to the trade-off between how many data points we are willing to query based on the current information versus how much time we are willing to wait to get more information. Often this trade-off is decided by the implementation of the simulator, how much time a simulation takes, and the hardware available.

As previously discussed, the core concept underlying active learning is that seeking out the most informative points and exploiting uncertainty regions of the input domain space reduces data redundancy and boosts model training efficiency, besides saving significant computational resources in the process. In this regard, the GP framework proves to be an appropriate modeling tool that, due to its nonparametric and Bayesian properties, allows it to handle both informativeness and uncertainty in a relatively natural and intuitive way. Active learning methodologies involving GPs are oftentimes associated with exploration-exploitation problems within Bayesian optimization contexts [12].

C. *Metamodeling*

The development and use of simulation metamodels [13] can be traced back at least to the beginning of the 70s. They constitute a type of auxiliary model that aims at approximating the function inherently defined by the simulation model and are explicitly defined by known and clear formula which can be evaluated at any given point in a rather effortless manner. Mathematical simplicity, speed, and interpretability are characteristics typically attributed to metamodels. Consequently, the use of metamodels within simulation analysis provides an additional level of understanding of the underlying system, as well as of the relationships between the system's input and output variables, while maintaining a computationally straightforward and economical approach to the problem [14].

As already mentioned, simulation metamodels are essentially input/output functions that are employed in such a way as to approximate the true, and usually much more complicated, function inherently defined by the simulation model itself. The metamodel's functional form is explicitly specified a priori, with many of its inputs being commonly shared with those of the simulation model. In this regard, the domain of applicability of the metamodel, or experimental region, should be clearly specified. This encompasses the definition of the input domain or region for which the metamodel should be a valid approximation.

The performance of simulation metamodels is inevitably defined by its modeling goals on an everlasting effort to balance the trade-off between computational speed and simplicity, and accuracy with regards to the simulation model. Therefore, it is equally important to specify the maximum or acceptable accuracy loss levels within the experimental region. The metamodel performance can be continually assessed via standard metrics such as the root relative squared error (RRSE). Given their modeling capabilities, the GPs have been widely used for metamodeling tasks in several fields [15][16].

III. Methodological Approach

Our approach to tackling the computational challenges often posed by simulation-based studies is twofold. First, we address the computational burden by providing a metamodeling framework for approximating the simulation results and enabling prediction over unlabeled regions of the simulation input space. By predicting simulation output values for unlabeled input data points, a reasonable amount of simulation runs can be skipped. Second, we adopt an active learning approach to specifically aim at carefully selecting the data to be simulated for posterior fitting in a more efficient way.

An overview of the methodology presented in this work is depicted in Figure 1, showing its iterative cyclic nature around the alternating phases of metamodeling and active learning. The a priori preparations for this approach include the specification of the relevant input variables to consider, and the corresponding output variables of interest, typically in the form of system Key Performance Indicators (KPIs). Along with it, the values ranges of input variables are set, and consequently, an unlabeled simulation input region for exploration, or search grid, is defined. This search region corresponds to the applicability domain in which we aim to explore the simulator's output behavior. Afterward, an initial data set comprising of a few previously run simulation results, i.e., labeled instances, is built, allowing the iterative process to initiate.

It is worthwhile mentioning that, in order to establish an operationally controlled experimental and replicable environment, we decided to follow an 'offline' pool-based active learning approach. This means that a labeled simulation pool consisting of pre-generated simulation results is used instead of accessing the simulation model directly. In this sense, this data set plays the role of the simulator by providing labeled instances on-demand whenever queried. From an active learning point-of-view, this constitutes a minor adaptation that, for all intents and purposes, has little to no impact on the analysis conducted hereafter.

The first step of this process involves the training of the metamodel using the above-mentioned initial set. Then, the prediction step follows, where the fitted metamodel is used to predict over the given unlabeled input region of interest or applicability domain. Prediction performance assessment is additionally carried out using a disjoint test set, effectively ending the metamodeling phase. Subsequently, the active learning procedure is initiated by requesting the labeled simulation pool for new data according to a pre-defined sampling strategy (essentially enclosing some information-based criterion) which is then added to the training data. Henceforward, the entire process is repeated cyclic and iteratively until a specific stopping criterion is satisfied. This criterion should

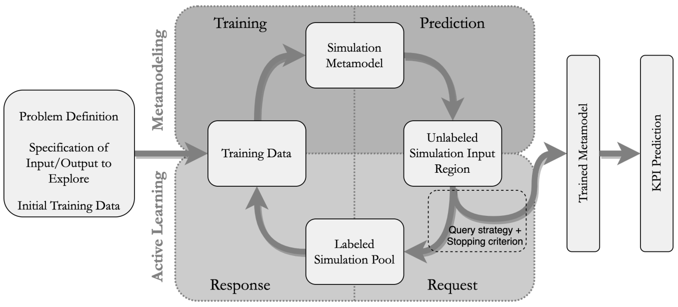

Figure 1. Main elements and steps of the proposed active learning metamodeling methodology.

consider the metamodel's relative performance and recognize when no further improvements are attainable solely via the expansion of the training set. Finally, the process ends with a trained metamodel specially tailored for the experimental input domain specified at the beginning.

IV. Experiments

A. Case Study

The study of ATM systems by means of agent-based simulation models is relatively recent [17][18]. Most of these models are designed to assess the impact of a limited number of airspace features, additionally covering small geographical areas. In the European context, simulation methods encompassing the entirety of the European civil air traffic are rare, confidential, both in terms of documentation and implementation, and oftentimes deterministic or driven by rule-based behaviors of their agents. Moreover, almost none of these simulators are able to generate performance metrics from the passengers' perspective.

The simulation model used in this work, henceforward referred to as *Mercury*, was previously developed in the framework of the H2020-SESAR JU ER3 Domino[1] project [19]. Mercury is a stochastic event-driven micro-level agent-based model designed to mimic the movements of both flights (consider reactionary effects) and passengers (including multi-leg itineraries with possible connections) for an entire day of operations at the European Civil Aviation Conference (ECAC) level. This type of approach provides ATM designers and practitioners with a better understanding of the involved systems and their relationships, which typically emerge in a given technological and operational setting. Mercury comprises a series of entities, namely, the airline operating center (AOC), individual flights, ground airport, network manager, arrival and departure managers (AMAN and DMAN, respectively), radar system, and flight air traffic management delay swapper. As passengers have limited decisions to make, they are not modeled as agents, although their preferences are indirectly encoded by the airlines via a logit utility-based soft cost. All the agents, particularly the AOC, are designed to estimate their operational costs and to act in order to tentatively minimize them. Different costs are considered, such as those

[1]https://domino-eu.com/

TABLE I. SUMMARY DESCRIPTION OF THE MERCURY INPUT/OUTPUT SIMULATION VARIABLES USED FOR METAMODELING.

var	name	description	values
x_1	compensation_magnitude_long1	Compensation between first and second thresholds for long-haul passengers in euros (€).	[0, 500, 1000,. . . , 5000]
x_2	first_compensation_threshold.	Threshold of arrival time after which the passengers receive a compensation in minutes.	[0, 60, 120,. . . , 480]
x_3	fuel_price	Price per kilogram of fuel in euros (€).	[0, 0.5, 1,. . . , 5]
y_1	arrival_delay_min_mean	Average arrival delay per flight in minutes.	real-valued
y_2	departure_delay_min_mean	Average departure delay per flight in minutes.	real-valued
y_3	total_cost_mean	Average total cost per flight operation in euros (€)	real-valued, positive
y_4	pax_tot_arrival_delay_mean	Average total arrival delay per passenger in minutes.	real-valued, positive
y_5	lcc_arrival_delay_min_mean	Average arrival delay per flight for low cost carriers in minutes.	real-valued
y_6	lcc_total_cost_mean	Average total cost per flight for low cost carriers in euros (€).	real-value, positive

TABLE II. A RANDOM LABELED RAW SAMPLE SHOWING THE TYPE AND STRUCTURE OF THE FEATURED DATA USED IN THIS WORK.

x_1	x_2	x_3	y_1	y_2	y_3	y_4	y_5	y_6
500	0	0.0	0.47	10.81	11335.41	151.74	6.96	8286.44
3500	360	3.0	0.12	11.88	120072.96	255.78	4.84	34846.82
500	60	1.5	-2.97	11.25	60685.01	206.22	5.12	17844.80
200	60	1.5	-3.62	11.22	61143.59	195.51	6.37	17552.87

in the form of costs of delays (including passenger and non-passenger related, reactionary, and curfews breaches costs), fuel consumption, and airspace charges.

Mercury's input data is essentially related to airline operations and flights, passenger itineraries, ATM operational data, fuel cost, and other charges. In terms of outputs, departing and arrival delays, taxi times, missed connections, and compensations are just some examples of detailed metrics produced per individual flight and passenger. After aggregation, different KPIs of the modeled system are computed. For more extensive details on the Mercury simulator, please refer to [20].

In this work, we focus on three input variables and six KPIs, as summarized in Table I, and on a scenario encompassing 1000 flights and their associated passengers. Besides allowing for the construction of an illustrative experimental design to assess the proposed active learning strategies, the choice for this set of variables originates from ongoing work regarding the impact analysis of Regulation 261, which essentially establishes common rules for passenger compensation and assistance in case of denied boarding, flight cancellation or long delays.

B. *Metamodeling framework*

We construct a metamodel consisting of six - due to the six KPIs of interest - independent GPs with a SE-ARD kernel [8] and a constant mean, all trained on the same data set. The input data is normalized to the unit cube, whereas the output data is standardized to have zero mean and unit variance. It is worthwhile remembering that in our modeling context, a labeled data point is a point from the simulation input space concatenated with its corresponding simulation output value. Conversely, an unlabeled data point corresponds to a simulation input point whose output value is unknown.

As previously mentioned, we use an 'offline' pool-based active learning approach and thus, we create a data set consisting of all the 1089 combinations of values for the inputs given in Table I. For each combination, we then run the simulator six times, giving a data pool $\mathcal{U}$ of 6534 possible data points. For the test set, we simulate each combination twice and take the average, yielding a test set with 1089 robust data points covering the full input space. The initial data set consists of four (one per input dimension plus one [21]) data points sampled by Latin Hypercube Sampling (LHS) from $\mathcal{U}$. Given the fact that we investigate six KPIs simultaneously, we utilize the available hardware to parallelize the simulator runs, querying six data points per iteration, and thereby cutting the required simulation time by six.

We adopt three different active learning strategies laid out in the following section. To benchmark the proposed method and the strategies, we implement the well-known linear regression with a degree of two as a metamodel and replace the active learning with LHS sampling the number of data points corresponding to that of the active learning framework. The performance of all the strategies is averaged across 30 runs and evaluated by the Relative Root Square Error (RRSE) on a separate test set. The active learning is run for 20 iterations corresponding to 120 simulations.

C. *Active Learning Strategies*

A central component in the active learning (AL) framework is the AL strategy which specifies how to choose the next points to be labeled by the oracle. Three different AL strategies are explored: random sampling and two kinds of variance-based sampling. The former consists of querying six data points at random. The first variance-based strategy is the common criterion when applying AL with a GP as the metamodel [16]. For a single output, it uses the inherent predictive variance (var) from the GP to query the data point with the largest variance, i.e. $x_{new} = \arg\max_x var(x)$ for $x \in \mathcal{U}$. We extend this to multiple outputs by applying the strategy for each output individually in each iteration.

However, some outputs may exhibit more variance than others and are thus more interesting. Therefore, we propose to allocate the available resources differently and therefore weigh the different outputs according to their average variance, such that for each data point to query, we choose, with a certain probability, an output to which the variance-based criterion described above is applied. In this way, the outputs with high variance are favored, although the ones with low variance still have a chance of affecting the process. The average variance is computed by taking the mean of the predictive variance for all the unique data points in the unlabeled data set $\mathcal{U}$ and for each output dimension. Afterward, the average variance across all the outputs is normalized such that they sum to 1, effectively converting it into a probability distribution, which is then used for choosing the output.

D. *Results*

The performances of the different AL strategies are shown in Figure 2. For all outputs, the baseline consisting of the linear regression with LHS is beaten by one of the approaches with GPs and AL, and it is the worst-performing approach for half of them. The outputs can be divided into two groups:

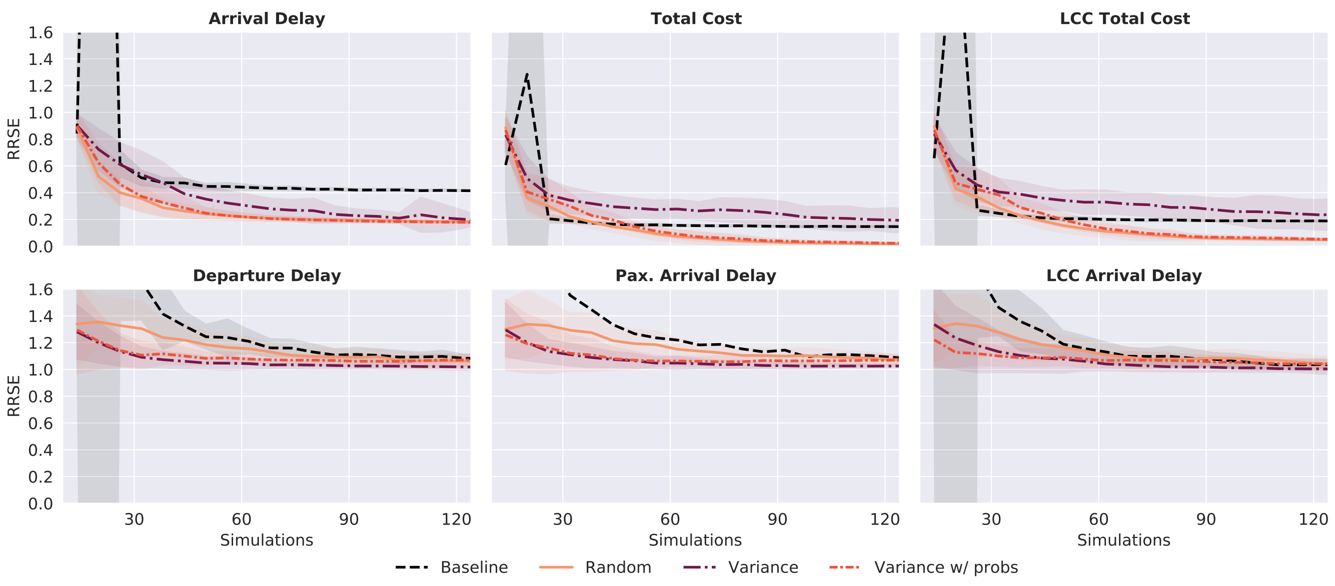

Figure 2. Performance of the baseline and three active learning strategies, averaged over 30 runs. The shaded regions show the ±1 standard deviations.

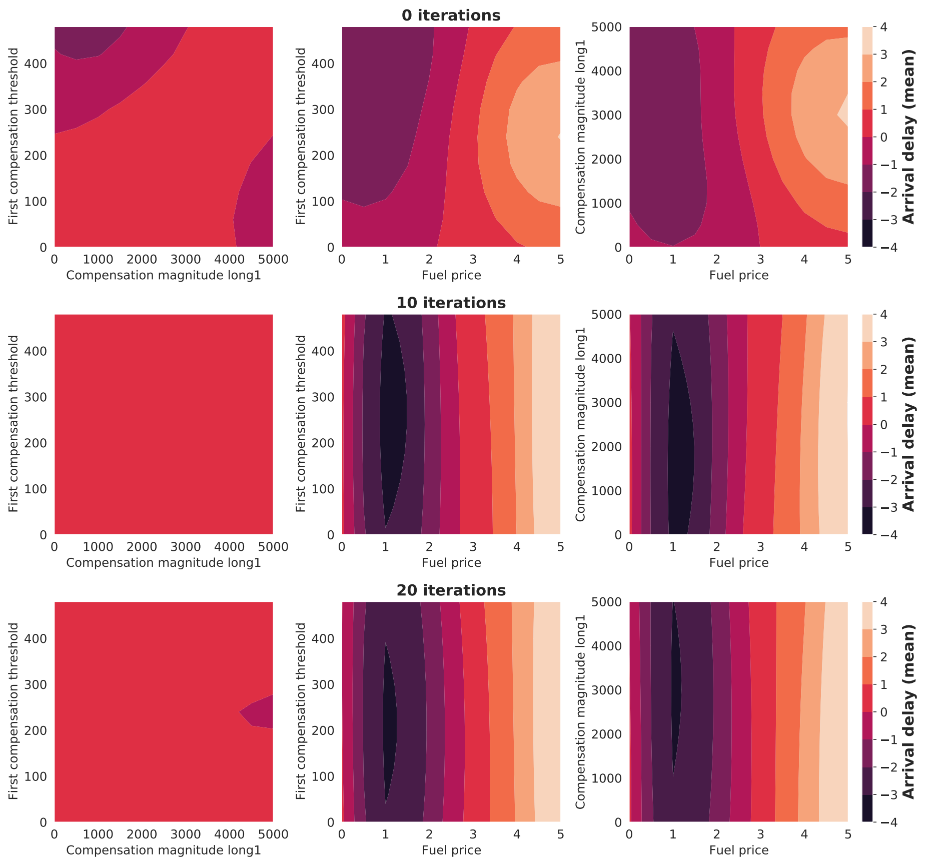

Figure 3. The predictions of *arrival delay* after 0, 10, and 20 iterations.

the outputs in the first row, where the metamodel achieves an RRSE significantly below one, and those in the second row, where it achieves a performance around one. In the former, the RRSE values below one indicate that the corresponding outputs follow a clear signal which is indeed a function of the inputs. Contrariwise, the RRSE values close to one, displayed in the latter case, indicate that the outputs are treated as random noise, yielding that they are being modeled indepen-

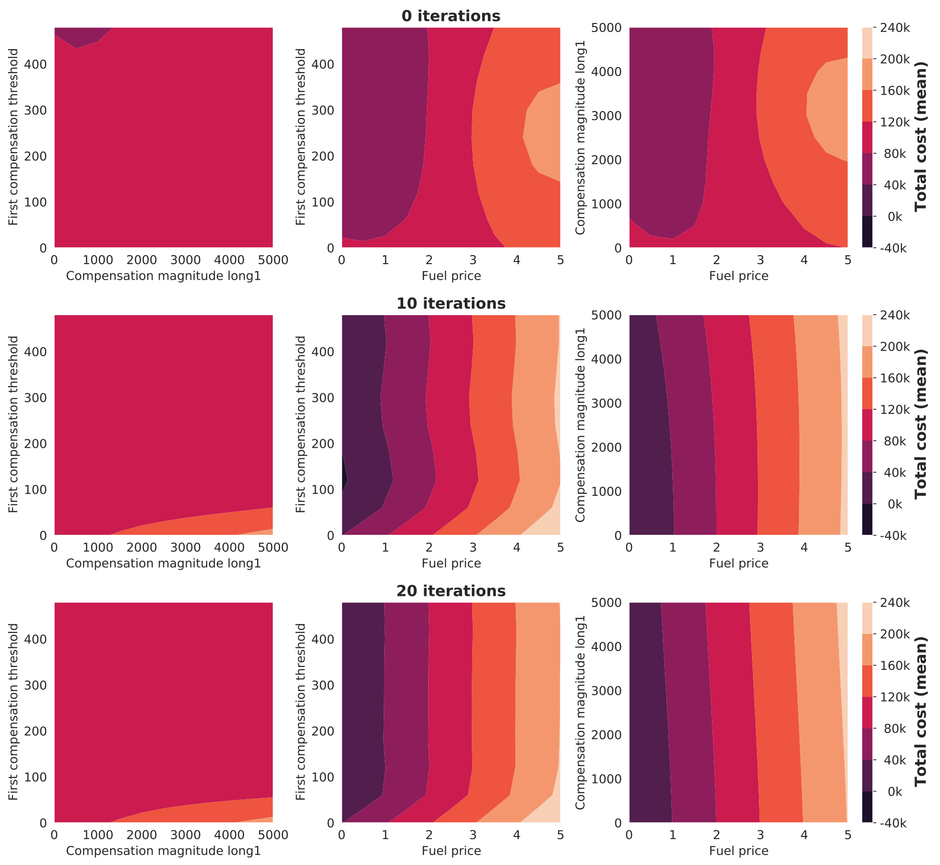

Figure 4. The predictions of *total cost* after 0, 10, and 20 iterations.

dently of the considered inputs. Regarding the AL schemes for the first group, the strategy *variance with probabilities* is on par with *random* but significantly better than *variance*. For the second group, *variance* is performing best, followed by *variance with probabilities* achieving almost the same performance. Thus across the six outputs, the best performing strategy is the variance-based criterion with probabilities since it is either comparable to the others or significantly better. In other words, it is a robust choice no matter the magnitude of noise in the output.

The curves in Figure 2 also show that the effect of querying more data diminishes through the AL iterations. The real impact of this appears when examining the predictions for the outputs after 0, 10, and 20 iterations. Given the three-dimensional inputs, we visualize the predictions for two inputs and then average the predictions across the third input, thus giving nine plots for an output. In Figure 3, the contours of the predictions of the *arrival delay* are shown. The predictions change significantly when going from 0 to 10 iterations, whereas they are almost identical after 10 and 20 iterations, showing that the metamodel almost does not change in the last ten iterations. Additionally, the predictions from the metamodels describe the behavior of the simulators, e.g., they show that the input *fuel price* has the largest effect on the *arrival delay*. Similar insights are obtained for the *total cost* in Figure 4, where only minor changes occur between iteration 10 and 20. Again, the *fuel price* has the largest effect on the *total cost*, though the interaction of the *first compensation threshold* and *compensation magnitude* have a significant impact too. In sum, these two figures show that fewer than 20 iterations might be sufficient in practice and how the metamodeling framework describes and reveals the simulator's behavior.

E. *Computational advantages*

In this section, we quantify the performance gains of employing a metamodeling strategy compared to using a more manual approach traditionally encompassing exhausting Monte Carlo simulation-based sampling schemes. We denote this setup as the *full system*, where we perform 10 simulations for each unique input data point. In other words, for each different combination of the input values, we run the simulator 10 times. In regard to quantifying the computational demand of the system, the 10 simulations per data point can be considered a conservative number since sometimes the data points are simulated 50 or 100 times [20], [22]. We can view the *full system* as a naive way of sampling simulation results.

We use the same input values as given in Table I with 1089 unique combinations of input values. Each combination is then simulated 10 times such that the full system requires 10890 simulation runs. Utilizing the same hardware specifications as for the metamodeling, each simulation takes approximately 20 minutes, and we are able to do six simulations in parallel. The simulation time for the full system is then

$$\text{Full system} : 10890 \text{ sim} \cdot 20 \text{ min}/6 = 25.2 \text{ days} \quad (1)$$

The resulting metamodels fitted to four initial data points and 20 iterations each with six data points, takes only 7 hours (the metamodels' training time plus the active learning process itself take less than five minutes in total):

$$\text{Metamodel} : 124 \text{ sim} \cdot 20 \text{ min}/6 = 7 \text{ hrs} \quad (2)$$

With this setup, the metamodel is ~ 86 times faster than the full system, though it should be kept in mind that the computational time of the full system is dependent on the number of combinations and repetitions. A plot showing the simulation time as a function of the number of different input values is seen in Figure 5.

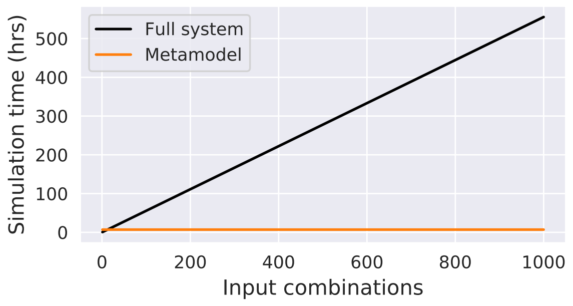

Figure 5. A comparison of the full system with 10 simulations per input combination and the metamodel fitted to 124 simulations.

Though the metamodel is much faster, it is not necessarily as accurate as the full system. For this reason, there is a constant concern for a balanced trade-off between computational speed and accuracy (or predictive performance), during the entire active learning process. On the one hand, we aim at bypassing the maximum number of simulation runs, while on the other hand, we want a metamodel capable of approximating the simulator's output behaviour with minor and controlled accuracy losses. These two objectives are of opposite natures, thus we should be able to recognize when the metamodel has reached its modelling performance plateau from which the cost of acquiring new simulation data points is likely a waste of time and computational resources.

In the following, we quantify the speed-up and accuracy of the two approaches by investigating the RRSE between the mean predictions from the full system and those of the metamodel. Often, not only the mean prediction is of interest but also the associated uncertainty which, by its turn, comes in the form of predictive variance. Thus, to account for this uncertainty, we additionally measure the Intersection Over Union (IOU) between the confidence intervals of the mean predictions given by ± 2 standard deviations. This will capture whether the mean predictions are off by imprecise predictions or due to high stochasticity present within the simulator's output. The standard deviations from the full system are calculated directly from the 10 simulations per unique input data point, and for the metamodel, they are taken from variance predictions directly provided by the Gaussian Process framework.

In Figure 6, we once again observe a clear distinction of the two types of outputs mentioned earlier, with the *arrival delay*, *total cost* and *lcc total cost* exhibiting almost no or little dispersion, in contrast to the remaining outputs. Nevertheless, in both cases the metamodel was able to successfully follow not only the mean but also the dispersion around it across the majority of the input combinations. These results brings to our attention the importance of considering a fitted metamodel that can explicitly model the expected values together with the corresponding variances of the output variables, given arbitrary simulation input combinations. While RRSE is a generally good metric to quantify the accuracy of a metamodel, it might be insufficient, or even misleading, when a high degree of random noise is present, which is the case of *departure delay*, *pax. arrival delay* and *lcc arrival delay*. It is true that we are interested in the average behaviour of the simulator, although it is equally important to ensure that the metamodel successfully captures most of the output variance generated by the simulation's inner stochastic nature.

Finally, we compare the accuracy of two metamodels M64 and M124 fitted to 64 and 124 simulations, respectively, using the AL strategy variance with probability. Their performance are evaluated against the mean and standard deviation from the full system. The results are averaged across 30 runs and are seen in Table III. The table shows two results: 1) running 60 simulations more give significantly better predictions, in terms of the RRSE, for the three outputs with low or almost no dispersion, whereas no significant improvements - rather worsening - are observed for the other three outputs, and 2) for the predictions of M124 either the RRSE is low, demonstrating that the mean predictions are very close to the ones of the full system, or the IOU is high, illustrating that even with high stochasticity in the output, making the mean prediction of the metamodel slightly off, the IOU is high.

V. Conclusion and Future Work

In this paper, we proposed an integrated modeling framework that integrates active learning with simulation metamodels in order to reduce the computational hindrances so often attributed to intense and systematic simulation-based analysis, particularly within the context of ATM systems.

Using an illustrative case study built upon a state-of-the-art ATM simulator and employing a Gaussian Process as a metamodel, we showed that active learning strategies are capable of increasing the modeling performances of simulation metamodeling approaches in a more efficient way. The results show that the baseline benchmark represented by the straightforward linear regression performed worse than the proposed approach. On the other hand, we found that the proposed multi-output metamodel together with the variance-based active learning strategy favoring the high-variance output dimensions is the overall best performing approach. Additionally, we were able to identify a strong association between

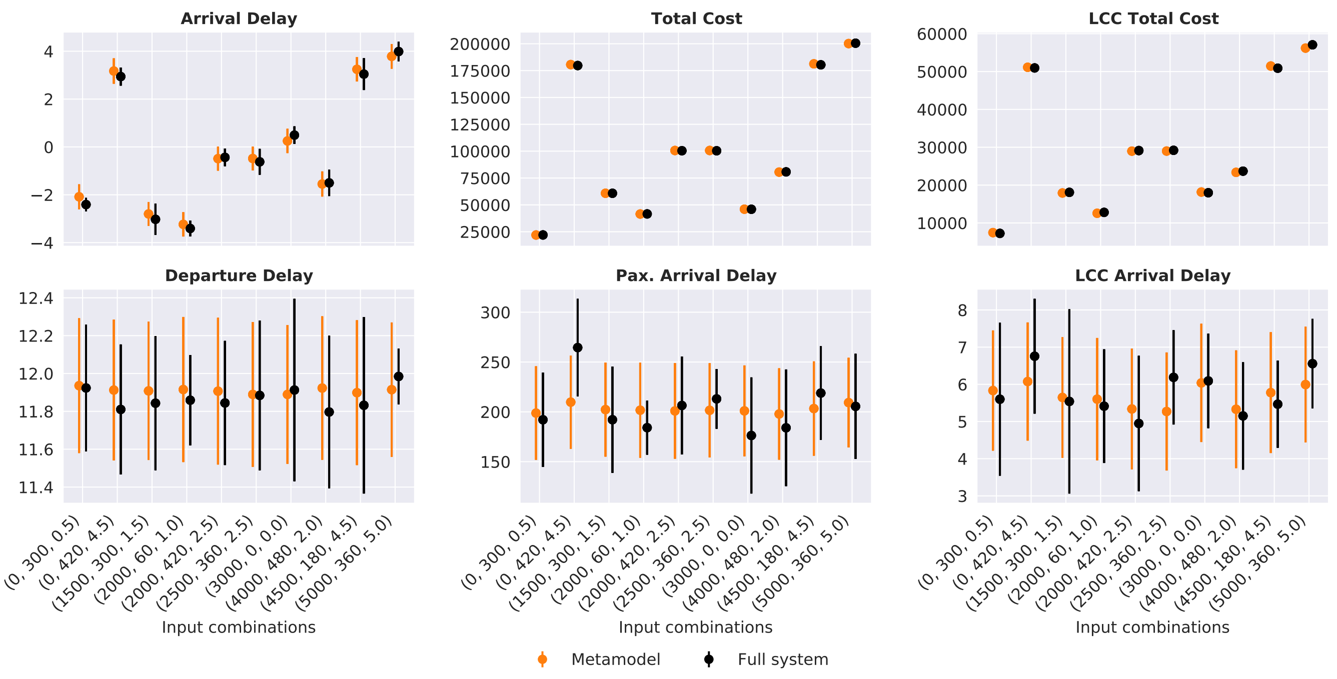

Figure 6. Comparison between the metamodel (variance w/ probabilities, fitted to 124 simulation results) and the full system. The dots are the mean predictions from the metamodel and the full system, and the lines are confidence intervals given by ±2 standard deviations. The 10 input combinations are chosen at random and are displayed as (compensation magnitude, compensation threshold, fuel price).

TABLE III. Comparison of the RRSE and IOU between the metamodel (variance w/ probabilities, fitted to 64 and 124 simulation results) and the full system.

Metamodel	Arrival Delay		Total Cost		LCC Total Cost		Departure Delay		Pax. Arrival Delay		LCC Arrival Delay	
#simulations	RRSE	IOU	RRSE	IOU	RRSE	IOU	RRSE	IOU	RRSE	IOU	RRSE	IOU
64	0.33 (±.07)	.50 (±.04)	0.28 (±.03)	.06 (±.01)	0.33 (±.04)	.15 (±.03)	1.13 (±.12)	.78 (±.01)	1.15 (±.12)	.78 (±.01)	1.17 (±.22)	.75 (±.02)
124	0.12 (±.01)	.73 (±.01)	0.02 (±.01)	.43 (±.02)	0.04 (±.01)	.45 (±.02)	1.25 (±.13)	.76 (±.02)	1.25 (±.12)	.77 (±.01)	1.10 (±.13)	.76 (±.02)

output variables that can be simply described as random noise and the ability of active learning to effectively improve the metamodel's learning. We also showed that, when compared to the more traditional ad-hoc and manual sampling approaches, the cojoint use of active learning and metamodeling strategies can greatly improve not only the exploration of the simulation input space but also reduce the computational burden often associated to the latter.

The core contributions of this work are part of the foundations of a broader aim encompassing the development of an auxiliary tool that enhances the simulation-based studies of ATM systems. The direct implication of this framework is to provide an auxiliary modeling tool capable of exploring the behavior of stochastic multi-KPIs simulators in an approximative but computationally fast fashion and eventually to guide the simulation analysis in a more efficient way facilitating the analysis work of both ATM researchers and practitioners. Notice that it is not our goal to dismiss the simulators entirely after the metamodels are obtained. Instead, the proposed methodology reflects our intentions of deploying the active learning-based metamodels along with the simulators themselves as a unified bundled modeling framework, as both types of models play a fundamental and complementary role within this integrated approach. While the metamodel aims at reducing the exploratory redundancy by trying to seek the most informative and distinct input data points, the simulation model ensures that this exploration process is maintained on the right track. In the end, the performance of the metamodel, and of the approach itself, will always be a function of the complexity of the simulator, the input ranges and amount of the input variables.

This work raised interesting questions that will definitely draw several lines of work in the near future. The relation between randomly distributed output variables and the efficacy of active learning strategies should be further investigated. Naturally, another possible step forward is to apply the proposed methodology to a wider range of inputs and outputs, possibly also taking into account discrete variables, in more complex and realistic simulation scenarios, for example, by considering a larger number of flights. The introduction of discrete variables is likely to lead to novel combinatorial challenges not present in this work. We also plan to employ multi-output GPs in the future, as they are able to identify correlations within the output variables via convolution, and experiment with alternative covariance functions. Furthermore, we will inevitably move to 'online' active learning frameworks where the simulation models are queried directly in lieu of pre-generated labeled pools.

VI. Acknowledgements

This work was supported by the ongoing NOSTROMO (Next-generation Open-Source Tools for ATM PeRfOrmance Modelling and Optimisation)[2] project funded by the SESAR Joint Undertaking[3] through the European Union's Horizon 2020 research and innovation programme under grant agreement No 892517.

References

[1] A. M. Law, *Simulation Modeling and Analysis*, 5th ed. McGraw-Hill Higher Education, 2015.
[2] P. T. Fishburn, J. Golkar, and K. M. Taaffe, "Simulation of transportation systems," in *Proceedings of the 27th conference on Winter simulation*, 1995, pp. 51–54.
[3] S. Bankes, "Exploratory modeling for policy analysis," *Operations research*, vol. 41, no. 3, pp. 435–449, 1993.
[4] D. Mietzner and G. Reger, "Scenario approaches: history, differences, advantages and disadvantages," in *EU-US Seminar: New Technology Foresight, Forecasting and Assessment Methods, Seville, May*, 2004, pp. 13–14.
[5] A. Cook, *European air traffic management: principles, practice, and research*. Ashgate Publishing, Ltd., 2007.
[6] A. Cook and D. Riva, *Complexity Science in Air Traffic Management*. Routledge London, 2016.
[7] SESAR Joint Undertaking, "Vision of the future performance research in sesar," Project PJ19 CI, Tech. Rep., 2018.
[8] C. E. Rasmussen and C. Williams, *Gaussian processes for machine learning (Adaptive computation and machine learning)*. The MIT Press, 2006.
[9] B. Settles, "Active learning literature survey," University of Wisconsin–Madison, Computer Sciences Technical Report 1648, 2010.
[10] X. Wang and J. Zhai, *Learning With Uncertainty*. CRC Press, 2016.
[11] *On Sequential Sampling for Global Metamodeling in Engineering Design*, ser. International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, vol. Volume 2: 28th Design Automation Conference, 09 2002.
[12] C. K. Ling, K. H. Low, and P. Jaillet, "Gaussian process planning with lipschitz continuous reward functions: Towards unifying bayesian optimization, active learning, and beyond." in *AAAI*, 2016, pp. 1860–1866.
[13] L. W. Friedman, *The simulation metamodel*. Springer Science & Business Media, 2012.
[14] L. W. Friedman and I. Pressman, "The metamodel in simulation analysis: Can it be trusted?" *Journal of the Operational Research Society*, vol. 39, no. 10, pp. 939–948, 1988.
[15] J. P. Kleijnen, "Kriging metamodeling in simulation: A review," *European journal of operational research*, vol. 192, no. 3, pp. 707–716, 2009.
[16] R. B. Gramacy, *Surrogates: Gaussian process modeling, design, and optimization for the applied sciences*. CRC Press, 2020.
[17] S. Bouarfa, H. A. Blom, R. Curran, and M. H. Everdij, "Agent-based modeling and simulation of emergent behavior in air transportation," *Complex Adaptive Systems Modeling*, vol. 1, no. 1, pp. 1–26, 2013.
[18] G. Gurtner, C. Bongiorno, M. Ducci, and S. Miccichè, "An empirically grounded agent based simulator for the air traffic management in the sesar scenario," *Journal of Air Transport Management*, vol. 59, pp. 26–43, 2017.
[19] L. Delgado, G. Gurtner, S. Zaoli, P. Mazzarisi, D. Valput, A. Cook, and F. Lillo, "Final tool and model description, and case studies results," *Domino Project, Deliverable 5.3*, 2019.
[20] L. Delgado, G. Gurtner, P. Mazzarisi, S. Zaoli, D. Valput, A. Cook, and F. Lillo, "Network-wide assessment of atm mechanisms using an agent-based model," *Journal of Air Transport Management*, vol. 95, p. 102108, 2021.
[21] D. Wu, C. T. Lin, and J. Huang, "Active learning for regression using greedy sampling," *Information Sciences*, vol. 474, pp. 90–105, 2019.
[22] G. Gurtner, L. Delgado, and D. Valput, "An agent-based model for air transportation to capture network effects in assessing delay management mechanisms," *Transportation Research Part C: Emerging Technologies*, vol. 133, p. 103358, 2021.

[2]https://nostromo-h2020.eu/
[3]https://www.sesarju.eu/