Taming Model Uncertainty in Self-adaptive Systems Using Bayesian Model Averaging

Matteo Camilli
Free University of Bozen-Bolzano
Italy
matteo.camilli@unibz.it

Raffaela Mirandola
Politecnico di Milano
Italy
raffaela.mirandola@polimi.it

Patrizia Scandurra
University of Bergamo
Italy
patrizia.scandurra@unibg.it

ABSTRACT

Research on uncertainty quantification and mitigation of software-intensive systems and (self-)adaptive systems, is increasingly gaining momentum, especially with the availability of statistical inference techniques (such as Bayesian reasoning) that make it possible to mitigate uncertain (quality) attributes of the system under scrutiny often encoded in the system model in terms of model parameters. However, to the best of our knowledge, the uncertainty about the choice of a specific system model did not receive the deserved attention.

This paper focuses on self-adaptive systems and investigates how to mitigate the uncertainty related to the model selection process, that is, whenever one model is chosen over plausible alternative and competing models to represent the understanding of a system and make predictions about future observations. In particular, we propose to enhance the classical feedback loop of a self-adaptive system with the ability to tame the model uncertainty using Bayesian Model Averaging. This method improves the predictions made by the analyze component as well as the plan that adopts metaheuristic optimizing search to guide the adaptation decisions. Our empirical evaluation demonstrates the cost-effectiveness of our approach using an exemplar case study in the robotics domain.

CCS CONCEPTS

• **Software and its engineering** → *Software system models*; *Software functional properties*; • **Computer systems organization** → *Self-organizing autonomic computing*.

KEYWORDS

Self-adaptation, Model uncertainty, Statistical inference, Bayesian model averaging

ACM Reference Format:
Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2022. Taming Model Uncertainty in Self-adaptive Systems Using Bayesian Model Averaging. In *17th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '22), May 18–23, 2022, PITTSBURGH, PA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3524844.3528056

SEAMS '22, May 18–23, 2022, PITTSBURGH, PA, USA

ACM ISBN 978-1-4503-9305-8/22/05…$15.00
https://doi.org/10.1145/3524844.3528056

1 INTRODUCTION

Research on uncertainty understanding, quantification and mitigation in software-intensive and (self-) adaptive systems has been the focus of several work in the last few years (e.g., [1–3]). Mainstream approaches to represent and handle uncertainty involve the enhancement of the analysis and plan activities of a feedback control loop architecture for adaptation (e.g., the well known MAPE-K loop [4, 5]) with model-based inference techniques in order to make predictions about future observations and plan actuation. These techniques are complemented with the calculation of confidence (or credible) intervals that incorporate uncertainty about parameter estimates, under the assumption that the "best" model representing the system (the phenomena of interest) has been selected [6, 7]. However, this approach could lead to overconfident assumptions because it tends to ignore the existence of different plausible alternative models to represent the understanding about a system and/or a process [8]. This condition is henceforth referred to as *model-selection uncertainty*. In statistics, this leads to the bias-variance problem, that is, parameter bias in under-specified models or to over specified models possibly leading to poor predictions [9].

In self-adaptive systems poor predictions yield failures in recognizing the need of an adaptation or unnecessary adaptations affecting dependability attributes [10]. Therefore, with the growing uncertainty surrounding modern self-adaptive systems (e.g., robotic software, autonomous vehicles), the ability to tame model uncertainty is becoming a concern of primary importance. Recent pieces of work demonstrate some effort in this direction, for example, including learning capability in the feedback loop to improve the model selection or definition step (e.g., [11, 12]). However, according to [9], settling on a single model (even the best one) could involve the risk of neglecting possible scenarios. To this end, a possible solution is represented by the adoption of a *Bayesian Model Averaging* (BMA) approach [13, 14]. This approach has been adopted in a range of disciplines to incorporate model-selection uncertainty into statistical inference and prediction [9, 13, 15–17], but, to the best of our knowledge, it has never been exploited to enhance analysis and plan activities in self-adaptive systems.

As illustrative example, let us consider a search and rescue robotic system whose goal is helping to discover people in emergency situations (more details are in Sec. 3). Due to little knowledge of the environment to be explored, the robot may be unable to go easily through a target location where a victim was remotely detected, possibly not overcoming all obstacles in non-patterned positions along the trail. The model that is often adopted in this situation is the one that conjectures a problem in the sensorial perception of the robot and drives adaptation to steer the robot's behavior. However, different scenarios could have transpired to

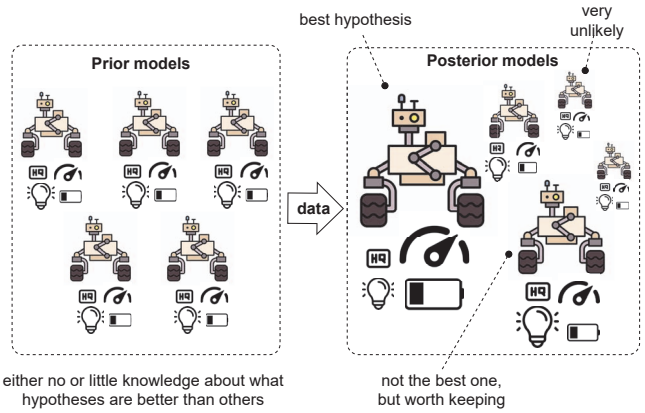


Figure 1: High-level interpretation of BMA: each model is shown as a robot holding its own beliefs (inspired by [15]).

cause safety issues. For example, ($scenario_1$) there may have been a problem mainly related to the speed, combined with high latency of the breaking system: the robot was too fast to be able to avoid the obstacle after a late detection. Or ($scenario_2$) the low light intensity (illuminance) of the search area could have prevented the detection of small obstacles, or alternatively ($scenario_3$) a low quality of the sensors (e.g., camera, LiDAR, ultrasonic range finder, etc.) could have been the main cause. Each $scenario_i$ corresponds to a different model of the world M_i associated with a different estimation of the occurrence of negative events, such as a safety violation σ, denoted by $p(\sigma|M_i)$. We do not know the true scenario **M**, but we do not really care about it either. Indeed, all that is relevant for our decision about the right action to undertake is the estimation of $p(\sigma)$ unconditional on any particular model of the world M_i.

The probability $p(\sigma)$ is referred to as the BMA estimate [17]. Adopting BMA means changing the modelling perspective: rather than first selecting the most plausible scenario M^* and then using the related estimation $p(\sigma|M^*)$, the likelihood of safety violations is assessed taking into account all scenarios simultaneously. This is performed by computing a weighted average, accounting for the plausibility of each scenario with a dynamic weighting scheme, adjusted as more information becomes available. Figure 1 gives a high-level overview of the approach. Each scenario's model is represented by a robot with its own beliefs about its parameters. Each robot and its parameters initially have the same size (left side) indicating no knowledge about the true scenario **M**. By gathering runtime evidence, the size of the robots changes (right side): they grow if they predict the data relatively well and shrink if they predict the data relatively poorly. In decision making, it is prudent to be informed by all the models (weighted by their importance), not just the best (largest) one.

In this paper, we present TUNE (Taming model UNcErtainty), a novel approach that augments the MAPE-K architecture with components that take into account not only the uncertainty about the parameters given a particular model, but also uncertainty across all candidate models. TUNE is fully automated and works at runtime along with the managed system in production. The contribution of the paper can be described as follows: we introduce TUNE that (1) enhances the MAPE-K feedback loop with the ability of mitigating model uncertainty using BMA; and (2) improves the adaptation decisions through metaheuristic optimization leveraging the BMA estimates. We present (3) an empirical evaluation that shows the accuracy, the effectiveness, and the cost of the proposed approach.

The remainder of this paper is as follows. In Sec. 2 we introduce some preliminaries. In Sec. 3 we describe the case study used to exemplify and evaluate the proposed approach. The whole TUNE approach is described in Sec. 4. We illustrate our empirical evaluation in Sec. 5, and we discuss advantages, shortcomings, and threats to validity in Sec. 6. An overview of related work is presented in Sec. 7. We then draw our conclusion in Sec. 8.

2 PRELIMINARIES

This section recalls the necessary background concepts and a comprehensive characterization of the problem that has been addressed.

2.1 Generalized Linear Models

In this work, we consider a class of statistical models that is a natural generalization of ordinary linear regression. *Generalized Linear Models* (GLMs) include, for instance, linear regression, logistic models, multinomial response model for counts and models for survival data. In a GLM, each response or outcome Y (i.e., dependent variables) is assumed to be generated from a distribution in the exponential family (e.g., normal, binomial, Poisson, and gamma distributions, among others). Given the independent (or explanatory) variables X, the mean of the distribution is:

$$E(Y|X) = \mu = g^{-1}(X\beta) \tag{1}$$

where $E(Y|X)$ is the expected value of Y conditional on X, the predictor $X\beta$ is a linear combination of uncertain/unknown parameters β typically estimated with maximum likelihood, and g is the *link function*. In GLMs, the relationship between the response and explanatory variables is not necessarily linear. The link function provides the relationship between the linear predictor and the mean of the response distribution. The choice of the link function depends on the nature of the response Y. According to [18], there exist common exponential-family distributions associated with the data they are typically used for. Given the family, a canonical well-defined link function is then derived.

As an example, in the case of Bernoulli and binomial distributions, the predicted parameter is one or more probabilities (i.e., real value in $[0, 1]$). The resulting model is known as logistic regression. The canonical link function is the *logit* function defined as follows:

$$X\beta = ln\left(\frac{\mu}{n-\mu}\right) \tag{2}$$

with n number of trials ($n = 1$ in the case of Bernulli distribution). In both cases, the mean of the distribution is computed as follows:

$$\mu = \frac{1}{1+e^{-X\beta}} \tag{3}$$

The expected value μ is a probability indicating the likelihood of occurrence of a single event. For instance, in the case of Bernulli distribution, we have a single boolean outcome, i.e., either 1 (True) or 0 (False). Thus, the expected value is the probability of occurrence of a positive (or negative) outcome.

2.2 Bayesian Inference

The Bayesian approach is a very popular framework for inference and prediction [19]. In particular, a very common goal in statistics is to learn about one (or more) uncertain/unknown parameters θ describing some details of a stochastic phenomenon of interest. To infer the parameters θ, we observe the phenomenon of interest and we collect observations $y = (y_1, y_2, ..., y_k)$ in order to compute the conditional density $p(y|\theta)$ of the observed data given θ, i.e., the likelihood function. The Bayesian approach also takes into account the hypothesis about θ. This information is often available from external sources such as operational profiles and expert information based on past experience or previous studies [20]. This information is represented by the *prior* distribution $p(\theta)$. By combining the prior and the likelihood using the Bayes' theorem (see Eq. 4) we obtain the *posterior* distribution $p(\theta|y)$, describing the best knowledge of the true value of θ, given the data sample y.

$$p(\theta|y) \propto p(y|\theta) \cdot p(\theta) \tag{4}$$

The Posterior distribution can be used in turn to perform point estimates of the uncertain parameters. This is typically addressed by summarizing the distribution through the posterior *mean*.

2.3 Bayesian Model Averaging

Let us assume σ is the quantity of interest we need to evaluate. To this end, BMA takes into account all the possible scenarios M_i related to the quantity of interest σ.

The BMA estimate is obtained through the equation:

$$p(\sigma) = \sum_i p(\sigma|M_i) \cdot p(M_i) \tag{5}$$

and adjusted as new information becomes available.

This information can be referred to collectively as "data," and the BMA estimate becomes:

$$p(\sigma|data) = \sum_i p(\sigma|M_i, data) \cdot p(M_i|data) \tag{6}$$

The observations influence not only the predicted quantity σ for each scenario, but also the probability of the scenarios themselves. The resulting distribution $p(\sigma|data)$ represents the beliefs about σ, after having made observations, while simultaneously taking all models into account. The model-averaged estimate of retains the uncertainty about the possible scenarios that might explain σ itself. This formulation provides a more nuanced and prudent estimate than simply assuming a single scenario as the true one.

3 CASE STUDY

To illustrate our approach we use a self-adaptive search and rescue robotic system, inspired by the case study in [21]. The system aims at supporting emergency circumstances such as fire, hurricane, or earthquake. Figure 2 shows a high-level overview of its main architecture components using an informal notation. The managed software includes proper abstractions of the physical interfaces (sensors and actuators), and key functions to carry out the rescue tasks, such as the navigation as well as the obstacle/human detection. The sensors (e.g., camera, LiDAR, ultrasonic range finder) feed the component *obstacle detector*, which detects and classifies obstacles or human beings. This information is used by the *navigator* to plan the steering angle, the acceleration, and the braking, which are then put into effect by the *controller*.

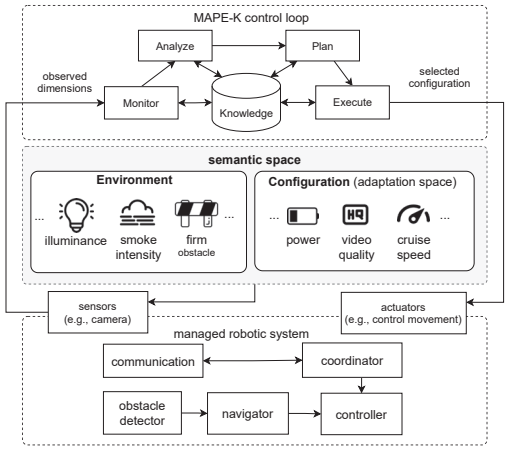


Figure 2: Main components of the self-adaptive robotic system and its semantic space.

Table 1: Semantic space of the search and rescue robot.

variable	space	type	domain
power	configuration	integer	[0, 100] %
cruise speed	configuration	continuous	[0, 5] m/s
bandwidth	configuration	continuous	[10, 50] Mbit/s
quality	configuration	categorical	{*low, mid, high*}
illuminance	environment	continuous	[40, 120000] lux
smoke intensity	environment	categorical	{*none, thin, thick*}
obstacle size	environment	continuous	[0, 120] ft^3
obstacle distance	environment	continuous	[0, 10] m
firm obstacle	environment	Boolean	Yes/No

The main components of the robotic software system can be configured at runtime to adapt the mode of operation, in the presence of uncertain changes in its environment (e.g., available bandwidth) and resource variability (e.g., estimated battery life). These configuration changes are enacted by engineering the controller with an *adaptation layer* embedding a typical MAPE-K feedback control architecture responsible for changing the system's operation mode and meeting dependability requirements at runtime. The configuration space of the system includes multiple dimensions. For instance, *power* is a discrete configuration dimension for the controller that may operate from *energy saving* to *full power* mode. The *cruise speed* represents a continuous configuration dimension for the navigator.

The set of configuration and environment dimensions is referred to as *semantic space*. For our illustrative example, they are roughly depicted in Fig. 2, while Table 1 lists all the dimensions that collectively compose the semantic space in our case study. More precisely, the semantic space is defined by a set of variables, each one having a space (either configuration or environment), a type, and a domain. For instance, *power* is an integer configuration variable that ranges in the interval [0, 100] percentage level. The *illuminance* represents instead a continuous environment variable ranging from 40 lux (for sunset/sunrise) to 120k lux (for brightest sunlight).

Table 2: Example of system-level requirements.

label	type	natural language statement
R_1	safety	The robot shall maintain a given protective distance between the actuated physical elements and possible obstacles in the search area.
R_2	safety	Misclassifications between objects and human beings shall not occur.
R_3	energy consumption	The robot shall reach the target location in the search area with enough power level.

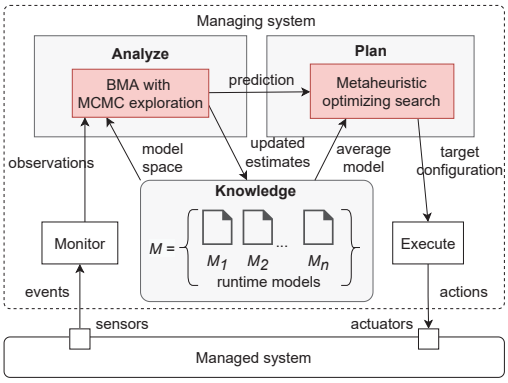


Figure 3: TUNE within a MAPE-K adaptation control loop.

The behavior of the rescue robot comprises different execution scenarios in which the robot shall carry out specific tasks yet satisfying the system-level requirements reported in Table 2. The table reports an extract as an illustrative example. The explanatory variables in the semantic space X may affect the ability of satisfying R_1–R_3. For instance, low video quality as well as high cruise speed can yield issues for the obstacle detector component that might lead to invalidate R_1 in case of hostile environment conditions (such as heavy rain or presence of smoke). At runtime, during the execution of the system, the likelihood of negative events invalidating the requirements can be modeled by adopting a logistic regression which depends on X. As introduced in Sec. 2, the link function is the *logit* and the mean μ of the distribution in Eq. 3 indicates the likelihood of observing a negative outcome, meaning that the system is not able to satisfy all the requirements.

4 THE TUNE APPROACH

In this section we describe an overview of TUNE (Sec. 4.1) and its integration into the analyze (Sec. 4.2) and plan (Sec. 4.3) components of a MAPE-K adaptation control loop.

4.1 Overview

Figure 3 illustrates a high-level overview of TUNE and its integration into a MAPE-K adaptation control loop. TUNE adopts GLMs as runtime models in the shared *knowledge*. According to our context discussed in Sec. 3, each GLM is a logistic model describing the phenomenon of interest affected by the variables in the semantic space. A given logistic model predicts the ability of satisfying system-level requirements (i.e., positive or negative outcome) according to Eq. 3. The model space M maintained by the knowledge component includes $n = 2^{|X|}$ possible models, that is, $M = \{M_1, ..., M_n\}$, where the i^{th} model M_i is obtained by including in the predictor only a subset of the explanatory variables in X. Under such circumstances, model uncertainty is commonly handled by performing model selection, as described for instance in [11]. Our analyze component adopts instead a model averaging approach that is often more satisfactory to inference because it takes explicitly into account the uncertainty in both the estimation of the model parameters and the model selection itself. The BMA module in Fig. 3 first estimates the parameters of interest conditional on each sampled model in M, then it computes an unconditional estimate using a weighted average of these conditional estimates according to the posterior probability of the corresponding models. Since the total number of models can be large, our approach adopts Markov Chain Monte Carlo (MCMC) exploration [22], to sample the model space. MCMC is, indeed, a popular exploration method which relies on the efficient generation or repeated random draws from a high-dimensional space. Such a stochastic exploration feeds BMA to estimate the quantities of interest. BMA estimates are used to predict possible violations of requirements. If a violation is predicted with high probability, the plan component is triggered. The plan makes use of metaheuristic optimizing search, as proposed in [23]. In our approach, we explore the semantic space by keeping the current assignment of the environment variables (i.e., observable phenomena) and looking for alternative assignments to configuration variables (i.e., controllable phenomena) that minimize the cost of the adaptation and, at the same time, maximize the likelihood of satisfying the requirements.

TUNE is fully automated and supported by a software implementation we used to carry out an empirical evaluation, as further detailed in Sec. 5.

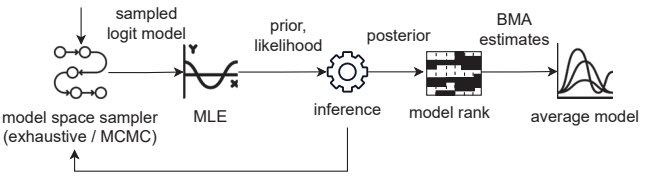


Figure 4: Zoom into the BMA process.

4.2 Analyzing with BMA

Figure 4 shows the overall process generating the average logistic model. It consists of the following components: the `model space sampler`, the `MLE`, the `inference`, and the `model rank` finally producing the average model based on the BMA estimates. The process receives as input the observations produced by the monitor $y = \{y_1, ..., y_k\}$, or simply *data* when the notation gets cumbersome. The data is a set of values assigned to the explanatory variables X (semantic space) associated with the corresponding response Y (either positive or negative outcome), observed by the system in production.

The `sampler` drives the whole process by exploring the model space either exhaustively or using a stochastic approach based on

Table 3: Example of logistic regression results using MLE.

logit regression results			
#observations	**log-likelihood**	**LL-Null**	**LLR p-value**
225	−213.58	−260.19	2.701×10^{-19}
estimations			
variable	**space**	**coefficient**	**p-value**
intercept	-	−5.9814	0.000
power	configuration	0.0409	0.001
cruise speed	configuration	−0.0532	0.000
smoke intensity	environment	−0.0806	0.004
firm obstacle	environment	0.9603	0.000

MCMC. For each sampled model M_i, and the data under consideration, we use Maximum Likelihood Estimations (MLE) [24] to make inference about the unknown population that is most likely to have generated the data. This means that the parameters of each model are estimated by using MLE, while the weighting scheme is developed under a Bayesian perspective introduced in Sec. 2.

The goal of `MLE` is to find the values of the model parameters $\hat{\beta}$ that maximize the likelihood function over the parameter space:

$$\hat{\beta} = \arg\max_{\beta} p(data \mid \beta, M_i) \tag{7}$$

Example 1 (MLE results). Table 3 contains an example of logistic regression using the following subset of the semantic space: *configuration* = {*power*, *speed*}, *environment* = {*smoke intensity*, *firm obstacle*}. The results have been obtained by applying the MLE method on 225 observations of the rescue robot system. The results show that the log-likelihood[1] of the model −213.58 is higher compared to the "null" model M_0 (i.e., the model containing the intercept only as parameter) that has a log-likelihood of −260.19 (LL-Null column). The result of the likelihood-ratio test (LLR p-value column) meets the common threshold for statistical significance 0.005. This indicates that including the aforementioned selected variables significantly improves model fit compared to M_0. The table also shows the estimated parameters. For instance, the coefficient for the configuration variable *speed* is −0.0532, with a p-value of 0.000. This means that the variable is significant, and that, on average, higher speed of the robot corresponds to lower probability of satisfying the requirements.

After MLE, the `inference` component computes the posterior probability used to rank M_i adopting a Bayesian perspective. To this end, we exploit the Bayesian Information Criterion (BIC) that, according to [14], is a popular criteria adopted to carry out Bayesian model selection. The BIC is approximately proportional to the likelihood that data are produced under the model M_i, as follows:

$$BIC \approx -2\, ln(p(data \mid M_i)) \tag{8}$$

Note that Eq. 8 involves the marginal likelihood of the data under the model M_i, that is $p(data \mid M_i)$. This quantity is used by the *Bayes factor* to compare two alternative hypotheses, as follows:

$$BF[M_i : M_j] = \frac{p(data \mid M_i)}{p(data \mid M_j)} \tag{9}$$

[1]The log-likelihood is often preferable compared to the likelihood, because of the mathematical properties of the natural logarithm (i.e., simpler to differentiate).

If the Bayes factor is greater than 1, the data under consideration strongly supports M_i rather than M_j. This means that the BIC information can be used to retrieve the best model having the largest log of marginal likelihood, which corresponds to the smallest BIC. If we stop the process to model selection (i.e., we pick up only the smallest BIC), we essentially ignore the presence of model uncertainty. Indeed, according to the data other models may have similar BIC and therefore may be considered equally good.

To account for the model uncertainty, our approach exploits the posterior probability of all the candidate models in the set M to produce a total order and quantify the extent to which the beliefs expressed by a given model are better or worse than others. To obtain the posterior probability of each model $p(M_i|data)$, the Bayes rule in Eq. 4 tells that we need to assign the prior probability $p(M_i)$ to each model and then update it after collecting the data by multiplying the marginal likelihood. The marginal likelihood is used to weight the prior probability so that models with higher likelihood have larger weights, and models with lower likelihood receive smaller weights. Thus, our framework computes the posterior, for each model, approximating the marginal likelihood based on the BIC defined in Eq. 8.

The `model rank` component quantifies the existing uncertainty by producing a total order in M according to the log of the *posterior odd* over the "null" model M_0. Intuitively, this quantity measures the likelihood of a given model compared to M_0 given the data. The posterior odd is computed as follows:

$$PO[M_i : M_0] = BF[M_i : M_0] \cdot O[M_i : M_0] \tag{10}$$

where $O[M_i : M_0]$ represents the *prior odd* defined as the ratio $p(M_i)/p(M_0)$.

Example 2 (Model uncertainty). In our rescue robot example, the semantic space contains 9 possible explanatory variables (see Table 1), thus 512 models using all possible combinations of the variables. Figure 5 visually represents the model uncertainty, after collecting 225 observations. The y-axis lists the variables (including the intercept), while the x-axis shows the rank of the first 20 models based on the posterior probability. Each vertical column corresponds to a logistic model. Black areas correspond to variables not included in a model. For instance, the first one in the rank includes 6 variables: the *intercept*, *smoke intensity*, *obstacle size*, *firm obstacle*, *power*, and *cruise speed*. The rank is produced by ordering the models according to Eq. 10. In our example, we started from the same prior probability for all models (i.e., uninformative prior). In this case, the prior odd is equal to 1 and the posterior odd is the same as the Bayes factor. The color associated with each column in Fig. 5 has been generated according to the log of the posterior probability. Since models with same colors have similar posterior probabilities, we can see clusters of models that are equally good based on the existing model uncertainty.

The posterior probability of each model, is then used to calculate weighted averages of quantities of interest as defined by the BMA weighting scheme in Eq. 6. Essentially, the next prediction $\hat{Y}^*$ after seeing the data can be calculated as a weighted average of the prediction of next observation under each model M_i, with the posterior probability of M_i being the weight. Models with higher posterior

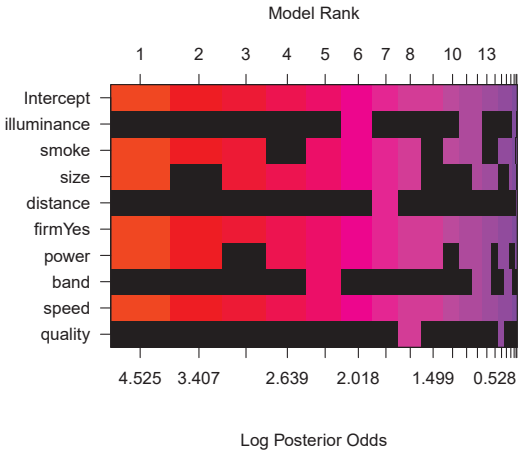


Figure 5: Model uncertainty after 225 observations.

receive higher weights, while models with lower posterior receive lower weights.

Example 3 (BMA estimates). Let us consider the model rank described in Example 2. According to the posterior probability of each model M_i, the BMA module computes the estimates reported in Table 4. The table lists, for each possible explanatory variable in the semantic space, the posterior mean and the posterior inclusion probability $p(x \neq 0)$ of each coefficient under BMA. Assume the following valid assignment leading to breaking R_1: {*power* = 52, *cruise speed* = 4.6, *bandwidth* = 28.6, *quality* = *low*, *illuminance* = 11800, *smoke intensity* = *none*, *obstacle size* = 5.38, *obstacle distance* = 2.58, *firm obstacle* = No}. According to the logistic model in Table 3, the expected value $E(Y|X)$ is 0.57, meaning that the model predicts the absence of negative events with high probability (i.e., greater than 0.5). However, this prediction is purely based on assumptions that ignore the presence of model uncertainty. For instance, the selected logistic model ignores the *obstacle size* that has high inclusion probability according to the BMA results in Table 4. Using BMA, the prediction $\hat{Y}^*$ is 0.47, meaning that the average model is more pessimistic (i.e., probability lower than 0.5).

When the total number of models is relatively small, the `model space sampler` can exhaustively enumerate them all to calculate the BMA estimates. In general, we may have a large number of variables, which often lead to long computation time. In this case, the sampler uses MCMC exploration based on the *Metropolis-Hastings* algorithm [25]. In the following, we limit the description to a high-level overview and we let the reader refer to [25] for further details. The algorithm produces a sample of the model space where the relative frequency of occurrence of each model in the sample represents a good proxy of its posterior probability. Starting from an initial model M_0 the exploration proceeds by randomly picking the next model M_1 and by checking whether this model improves the posterior probability computing the posterior odd $PO[M_1 : M_0]$ based on the Bayes factor and the prior odd, as in Eq 10. If the value is greater than 1, M_1 improves the posterior compared to M_0. In this case M_1 is included in the final sample. If the posterior odd is less than 1, the final sample still needs some representatives

Table 4: BMA estimates and inclusion probability for each explanatory variable after 225 observations.

variable	space	posterior mean	$p(x \neq 0)$
intercept	–	−6.1402	1.0000
power	configuration	0.0338	0.9032
cruise speed	configuration	−0.0514	0.9993
bandwidth	configuration	−0.0037	0.0994
quality	configuration	0.0001	0.0438
illuminance	environment	0.0004	0.0765
smoke intensity	environment	−0.0734	0.9094
obstacle size	environment	0.1401	0.8478
obstacle distance	environment	0.0001	0.0552
firm obstacle	environment	0.9097	0.9958

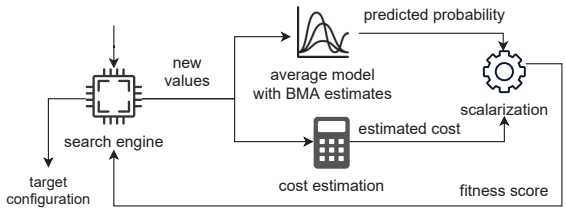


Figure 6: Zoom into the optimizing search process.

of this model. In this case, the posterior odd reflects the chance that M_1 is included in the final sample. After generating m models $M_0, M_1, ..., M_m$, the posterior of each model M_i can be estimated by using the relative frequency of occurrence of M_i in the sample. These probabilities can be used to calculate the BMA estimates.

4.3 Planning with BMA Estimates

The plan component takes as input the predictions produced by the analyze component as well as the average logistic model computed by applying BMA. An individual prediction $\hat{Y}^*$ triggers the adaptation when the probability of satisfying the system-level requirements is lower than 0.5. In this case the the plan component searches for an alternative configuration of the system that ensures a prediction greater than 0.5, while minimizing the cost of the adaptation. It is worth noting that a configuration is specified by an assignment of values for all the configuration variables (i.e., a subset of the semantic space including the dimensions that can be controlled). Therefore, the configuration variables collectively compose the adaptation space of the managed system. A target configuration (if any) represents the adaptation selected by the plan that is then executed in the next step of the MAPE-K loop, as shown in Fig. 3. The plan component makes use of a metaheuristic optimizing search approach [26] to reduce the cost of enumerating all possible valid configurations.

Figure 6 shows an overview of the automated process that selects a target configuration. It is composed of four main elements: the `search engine`, the `average model` (produced by the analyze component), the `cost estimation`, and the `scalarization`. The iterative process starts with the `search engine` generating a set of new values for all the configuration variables (e.g., modified cruise speed or power). The new values are used by the average model to predict the probability of satisfying the requirements and

by the `cost estimation` that calculates the cost of the adaptation according to the initial and the new values. The `scalarization` computes the *fitness score* depending on these two latter outcomes. Then the fitness score is fed back to the search engine to generate new (and better) input values over the next iterations. This process continues until a given time/iteration budget runs out.

The `search engine` drives the whole iterative process through a *genetic algorithm* to create new individuals (i.e., configuration values) using typical *crossover* and *mutation* operators to find better candidates while promoting diversity. The search is driven by the following two objectives: (1) maximize the predicted probability; and (2) minimize the cost of the adaptation. The two objectives map to two fitness functions f_1 and f_2 that are then combined by using *linear scalarization* [27]. The scalarization is an *a priori* method which reduces the multi-objective optimization problem to single-objective by expressing preference information before starting the search. The scalarization applies the following weighted sum to obtain the fitness score:

$$score = \omega_1 \, f_1(\cdot) + \omega_2 \, f_2(\cdot) \tag{11}$$

with the weights ω_1, ω_2 in $[0, 1]$ such that $\omega_1 > \omega_2$, as the search process penalizes high costs yet giving more value to high predicted probability. Since the search aims at minimizing the score, we define the two fitness functions in a way that the lower the result, the better the adaptation.

The function f_1 is as follows:

$$f_1(p) = \begin{cases} 1 - p & p > 0.5 \\ (1 - p) \cdot \lambda & otherwise \end{cases} \tag{12}$$

with p probability value given by the average model, and λ constant penalty term that discourages values lower than 0.5.

Given a static cost profile $u = \{u_1, ..., u_{|X|}\}$ such that u_i in $[0, 1]$ for all i, the function f_2 is defined as follows:

$$f_2(C, C') = \sum_i u_i \, \frac{|C_i - C'_i|}{ub(X_i) - lb(X_i)} \tag{13}$$

with C and C' vectors of initial and new values, respectively, $ub(X_i)$ and $lb(X_i)$ upper-bound and lower-bound of the corresponding variable X_i, respectively, u_i relative cost of changes for X_i. The idea of f_2 is to estimate the cost by accounting for the weighted magnitude of the change considering all the configuration dimensions. The meaning of the weight u_i for all i is to rank the configuration dimensions according to the cost of changing them. Intuitively, changing the cruise speed of the robot is cheaper than changing the power level, which implies a potentially expensive charging process. In this case, the weight associated with the variable *power* should be greater than the one associated with *cruise speed*.

Example 4 (Adaptation). Let us consider the BMA estimates introduced in Example 3 as well as the initial logistic model in Table 3. In the previous example we saw that the initial model leads to a false positive prediction. According to the average model, we observe instead a true negative triggering the adaptation process. Starting from the assignment in Example 3, the search engine executes the genetic algorithm that emits the following target adaptation: {*power* = 52, *cruise speed* = 2.4, *bandwidth* = 28.9, *quality* = *low*}. Since *power* has a very high weight compared to the other variables (e.g., 70% higher than *speed* in this example), its assignment remains the same. The *quality* configuration does not change as well. Indeed, according to the posterior mean and the inclusion probability in Table 4, this explanatory variable has low influence on the outcome. The *cruise speed* has instead higher influence. Here, we can observe a substantial change that causes the robot to slow down. According to the average model, this adaptation increases the predicted probability from 0.47 to 0.56.

5 EMPIRICAL EVALUATION

In this section we report on the empirical evaluation of TUNE. We introduce our research questions and design of the evaluation (Sec. 5.1) and then we present the major results (Sec. 5.2).

5.1 Research Questions and Design

The purpose of the evaluation is to study the extent to which TUNE can improve the predictions of the analyze component and the adaptation decisions of the plan component. In particular, we aim at answering the following three research questions:

- **RQ1:** What is the prediction accuracy obtained through BMA estimates adopted by the TUNE analyze component?
- **RQ2:** What is the effectiveness of the adaptation decisions taken by the TUNE plan component through BMA estimates?
- **RQ3:** What is the cost of calculating the BMA estimates in TUNE to mitigate the model uncertainty?

To answer the research questions, we designed a set of controlled experiments using the case study introduced in Sec. 3. In particular, we simulated multiple runs of the system to sample the semantic space X (see Table 1) and observe the response Y, i.e. a boolean outcome representing the satisfaction of the requirements in Table 2. The system response has been generated according to a ground truth baseline built by defining an *oracle* logistic model.

Then, we executed the analyze component multiple times adopting alternative competing models in the model space M as well as the BMA estimates built by using the same data sample, composed of 450 observations. We then used all the models to make predictions and compare their accuracy.

The predictions have been used as input to run the plan component and study the effectiveness of the adaptation decisions driven by alternative models in M and the BMA estimates. The cost of calculating the BMA estimates has been assessed measuring the execution time by varying the sample size and the number of potential explanatory variables in the semantic space.

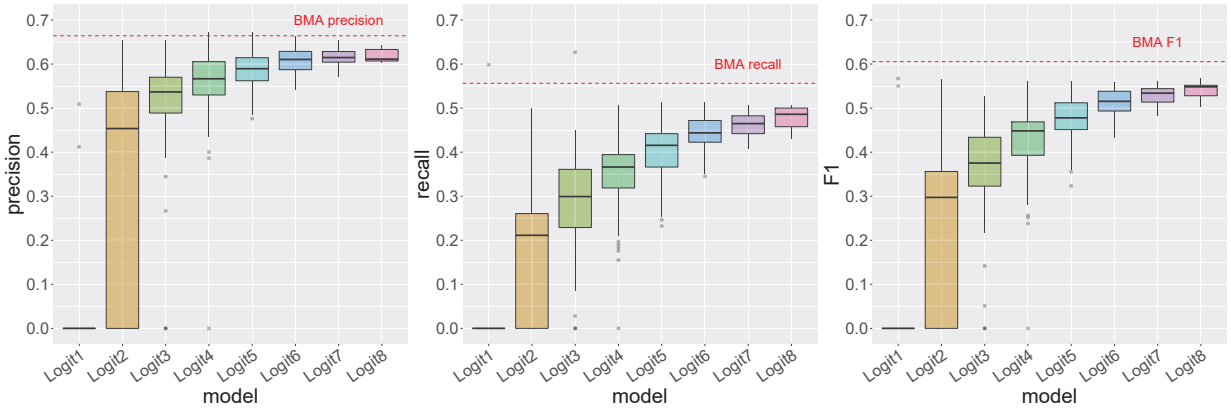
All the experiments have been conducted by using a commodity hardware machine equipped with: 2.3 GHz Dual-Core Intel Core i5 CPU, and 8 GB 2133 MHz LPDDR3 RAM.

Hereafter, we discuss the most relevant results and we refer the reader to our dataset for the replicability of the experiments[2].

5.2 Results

5.2.1 RQ1 (prediction accuracy). To address RQ1 we executed the analyze component described in Sec. 4.2 by changing the model used to make predictions. Specifically, for the semantic space in Table 1 (9 variables) we built all possible 512 logistic models fitted

[2]Replication package available at https://github.com/matteocamilli/bma-package.



(a) Precision **(b) Recall** **(c) F1 score**

Figure 7: Accuracy comparison between BMA estimates and all the other 512 models.

using MLE as well as the BMA estimates based on the same training set with 450 observations generated through the oracle model.

We assessed the accuracy comparing the classical metrics *precision*, *recall*, and *F1 score* [28] for all the models using a validation set with other 450 observations. The precision is the number of outcomes correctly classified as "requirements satisfied" (i.e., true positives) divided by the total number of positive outcomes in the evaluation set (i.e., sum of true positives and false positives). The recall measures the number of true positives divided by the sum between true positives and false negatives, that is, the outcomes wrongly classified as "requirements unsatisfied". High recall but low precision means many positive predictions, most of them incorrect. High precision but low recall means very few positive predictions, most of them correct according to the oracle. Ideally, the analyze component should use a model with high precision and high recall that are combined in the F1 score defined as the harmonic mean of the two quantities (the higher the F1 score, the better).

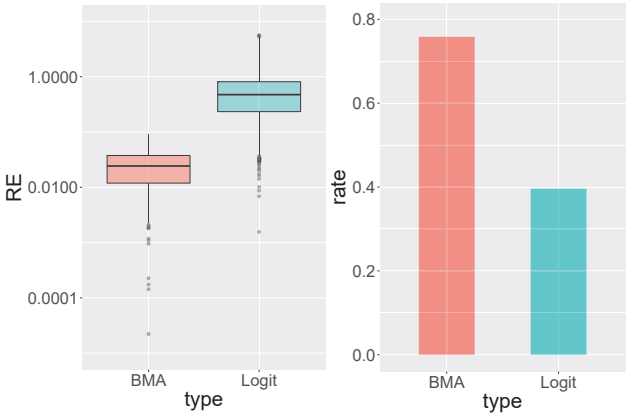
Figure 7 shows the results of the prediction accuracy by using box plots. The plots compare the three metrics precision (Fig. 7a), recall (Fig. 7b), and F1 score (Fig. 7c) using the BMA estimates and all the other 512 models clustered together by number of explanatory variables (Logit1–Logit8). Overall, we can observe that by increasing the number of explanatory variables in the models, we improve the median value of the three metrics and we also decrease the dispersion, as measured by the interquartile range $Q3 - Q1$ (i.e., the distance between the upper and lower quartiles). Excluding the cluster Logit1, which yields consistent and very low values (close to zero), the median varies from ~ 0.3 to ~ 0.55, while the interquartile range of the F1 score varies from ~ 0.35 to ~ 0.05. This means that, in our case study, a model selection in Logit8 would generally lead to better results than considering the other clusters (Logit1–Logit7). However, the data indicates the average model as the best one considering both prediction and recall, meaning that model averaging adopted by TUNE is always superior than model selection. Indeed, the F1 score obtained by using the BMA estimates is consistently higher than the median as well as the Q3 values of all the clusters. Considering the median of Logit2–Logit8, the BMA approach improves the F1 score from $\sim 5\%$ to $\sim 30\%$.

RQ1 summary: the prediction accuracy of TUNE is always higher using BMA than other competing models. This means that model averaging is superior than model selection. By applying model uncertainty mitigation through BMA, the TUNE analyze component improves the F1 score up to 30%.

5.2.2 RQ2 (effectiveness of the adaptation decisions). To address RQ2 we conducted additional controlled experiments with the plan component introduced in Sec. 4.3 by changing the model used to drive the adaptation decisions. We considered the same setting introduced in Sec. 5.2.1, that is, the same semantic space, and all the 512 logistic models fitted using MLE, as well as the BMA estimates, based on 450 observations of the search and rescue robot generated through the oracle model. Then, from the validation set, we selected 302 observations associated with the negative outcome "requirements unsatisfied". Starting from these valid assignments, we executed the planning process for each model and each negative outcome (154624 executions in total).

To measure the effectiveness of the adaptation decisions, we considered the *Relative Error* (RE) and the *Success rate* after producing the selected adaptation (i.e., a new assignment to configuration variables). The RE measures the magnitude of the difference between the prediction made by the adopted model and the prediction made by the oracle model after applying the selected adaptation. Intuitively, the higher the RE, the lower the chance to see a *success*, that is, select an adaptation leading to the positive outcome "requirements satisfied". Thus, we measure the success rate as the ratio between the number of successful adaptations and the total number of planned adaptations.

Figure 8 shows the results of the effectiveness of adaptation decisions in terms of RE (box plot in Fig. 8a) and success rate (bar plot in Fig. 8b). The two plots compare the results obtained by using the BMA estimates and all the other logistic models. We can observe that BMA estimates lead to lower RE, and therefore, higher success rate. According to our results in Fig. 8a, the median RE measured with BMA (0.0242) is two orders of magnitude lower than the median RE measured with all the other models (0.4734).



(a) Relative error (log scale) **(b) Success rate**

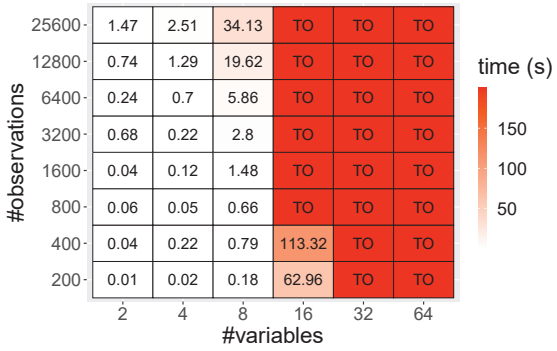
Figure 8: Effectiveness comparison between BMA estimates and all the other 512 models.

Figure 8b shows that the success rate with BMA (0.7582) is 36% higher than the the success rate with all the other mdoels (0.3624).

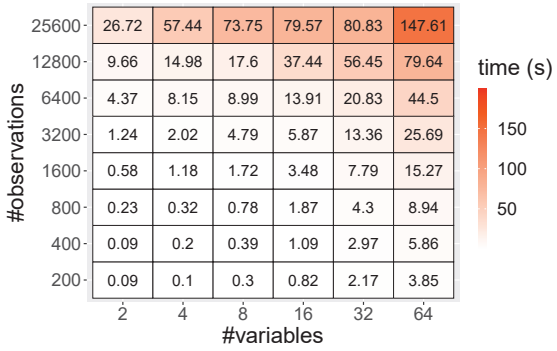
RQ2 summary: the effectiveness the adaptation decisions of TUNE using BMA is higher than other competing models. The BMA estimates reduce the RE by two orders of magnitude and lead to observe a 36% higher success rate.

5.2.3 RQ3 (cost of BMA estimates). To address the last research question we carried out controlled experiments with the BMA module introduced in Sec. 4.2 by controlling the number of explanatory variables in the semantic space (thus, the size of the model space), and the number of observations. We repeatedly calculated the BMA estimates by varying the variables from 2 to 64 (i.e., from 4 to 1.8×10^{19} models) and varying the number of observations from 200 to 25600. We repeated each experiment 100 times and we measured the effort as the wall-clock time (seconds) required by the BMA module to calculate the estimates. Being our approach meant to be executed at runtime along with the managed system, we would like to see a negligible cost (the lower the time, the better).

Figure 9 contains two heatmaps showing the mean cost. In all the runs, we set a timeout (TO) of 200 seconds. Figure 9a shows the cost when adopting an exhaustive enumeration of all the models in the model space, while Fig. 9b shows the cost when using MCMC exploration. We can observe that for low-dimensional model spaces (from 2 to 8 variables) the cost is generally lower when using exhaustive enumeration, even when increasing the number of observations. In both cases the time is less than a second from 200 to 800 observations and increases up to 0.56 and 1.2 minutes with $25k$ observations, respectively. However, the MCMC exploration yields better scalability. With high-dimensional model spaces (from 16 to 64 variables), we can observe that the exhaustive exploration leads to higher costs, exceeding the TO in almost all the cases. The MCMC never reaches the TO even in the worst case, when the model space has an order of magnitude of 10^9.



(a) Cost with exhaustive enumeration.



(b) Cost with MCMC exploration.

Figure 9: Cost of calculating the BMA estimates.

RQ3 summary: for low-dimensional model spaces (up to 8 variables), the cost is generally lower when using exhaustive enumeration. The MCMC exploration yields better scalability and it never reaches the given TO. Even with very high-dimensional spaces (10^9 models), the time varies from 3.8 to 147.6 seconds when increasing the observations from 200 to $25k$.

6 DISCUSSION AND THREATS TO VALIDITY

In this section we briefly discuss major advantages and disadvantages of TUNE, as well as how we mitigated threats to validity.

6.1 Advantages and Disadvantages

According to our experience, TUNE can be effectively used to account for, and then mitigate, uncertainty across models. Thus, its adoption offers the following advantages:

- TUNE deals with the oversimplified view taken by model selection which ignores model uncertainty and assumes zero probability for all but one model.
- Model averaging improves classical hypothesis testing in which a model is either accepted or rejected wholesale (i.e., the so-called "all-or-nothing" mentality). BMA retains instead all plausible models and ranks them all for the final inference stage.
- BMA makes adaptation decisions robust to misspecification. In case one model is selected, we assume the modeler is 100% sure of being correct, that is often not realistic. In TUNE, a range of

competing models are considered, thus increasing the chance that at least one of the models is approximately correct.
- TUNE gracefully updates estimates in the shared knowledge as the data accumulates. The model weights are continually adjusted preventing model lock-in when circumstances change.
- TUNE is fully automated and works at runtime along with the managed system in production. It does not require heavy training (offline) stages.

The aforementioned advantages do not come for free. Based on our evaluation, a few disadvantages may arise:

- Since a set of models is maintained, there exists the risk of assigning nonzero probability to some "wrong" models, especially in a cold start.
- The model space exploration introduces a computational overhead especially with high-dimensional model spaces. Stochastic sampling like MCMC can be used to mitigate this drawback.

6.2 Threats to Validity

We mitigated *construct validity* threats by assessing the appropriateness of our claims on the basis of our measurements. We measured the prediction accuracy using classical quality metrics in this context: precision, recall, and F1 score. The RE and success rate are common metrics to quantify the effectiveness of adaptation decisions. The execution time is again a natural measure of the cost.

To reduce threats to *internal validity*, we designed a set of controlled experiments detailing the independent variables of interest. In particular, our experimental setting allows for direct access to: the oracle model, the variables in the semantic space, its size, and the number of observations. This direct manipulation has been fundamental to assess cause-effect relations between external factors and both benefits and costs of our approach. We also enable replication by making our implementation and the experimental results publicly available.

External validity threats may exist if the characteristics of the system in our case study are not indicative of the characteristics of other systems. We limited these threats by adopting an established case study in self-adaptive systems having a nontrivial semantic space [21]. For practical reasons, we conducted the evaluation by simulating our case study. The application of our approach to additional case studies in other domains is part of our future work.

Threats to *conclusion validity* exist since BMA with MCMC is guided by a stochastic sampling of the model space. To avoid the risk of obtaining results by chance, we repeated our experiments multiple times. In particular, accuracy and effectiveness has been measured calculating the estimates with a validation set of 450 observations. Furthermore, each experiment to measure the cost of BMA with MCMC has been repeated 100 times.

7 RELATED WORK

Over the past years, researchers recognized the importance of studying the notion of uncertainty in the context of self-adaptation [2, 8, 21, 29]. Thus, a number of techniques explicitly integrating uncertainty in modeling recently emerged in different stages of the engineering life-cycle, such as design-time specification [30], testing [31–33], and runtime verification [6]. The mainstream approach to express uncertainty is by using model parameters, that are then handled using alternative methods including: probabilistic methods [34], where parameters are described by probability density functions; fuzzy sets [35], in which uncertain parameters are described with fuzzy boundaries; and interval analysis that aim at studying the limits of variation to detect either best/worst cases or violated requirements [3, 6, 7].

The Bayesian perspective in analyzing the variation of uncertain parameters is often preferred than the frequentist one, since it embeds the probability of the initial hypothesis in the inference framework [36, 37]. Under this setting, runtime analysis techniques supporting adaptation decisions usually keep alive Markov models (e.g., Discrete/Continuous Time Markov Chains) and calibrate their transition probabilities (representing uncertain QoS properties) using Bayesian inference [36, 38]. Improvements of these approaches have been proposed to alleviate the negative effect of historical data on the estimation by using aging mechanisms (e.g., Kalman filters [39]) to discard old information [37, 40]. To quantify the degree of uncertainty as the deviation from initial beliefs, the notion of Bayesian surprise [41] has been introduced to measure the distance from a prior to a posterior and support decision making when unexpected circumstances possibly require adaptation.

All the aforementioned approaches work under the overconfident assumption that the best model representing the phenomena of interest has been selected over plausible alternative models. The importance of this type of model uncertainty has been recently discussed in [11]. This latter approach applies model selection that aims at finding optimal hyper-parameter settings of Deep Neural Network models over alternative choices. The model is then used to predict the adaptation subspace possibly satisfying multiple types of goal (threshold goals and optimization goals). A different approach presented in [12] infers the models at runtime exploiting a generalization of classical Markov model to deal with situations in which the state of the system is not known in advance.

To the best of our knowledge, our work introduces the first approach integrating BMA into a MAPE-K control loop to tame model uncertainty at runtime in self-adaptation.

8 CONCLUSIONS

In this paper we presented TUNE, an approach to mitigate the uncertainty related to the model selection process in self-adaptive systems by using Bayesian Model Averaging. This method improves the predictions made by the analyze component as well as the plan that adopts metaheuristic optimizing search to guide the adaptation decisions. Our empirical evaluation shows the cost-effectiveness of TUNE through controlled experiments with an exemplar case study in the robotics domain. According to our experience, BMA is better than model selection. It yields prediction accuracy up to 30% higher and effectiveness of the adaptation decisions 36% higher. The overhead introduced by BMA is negligible with low-dimensional model spaces (up to 8 variables) and it can be alleviated with very high-dimensional spaces using MCMC exploration. Indeed, the approach exhibits good scalability up to 10^9 models and $25k$ observations.

As future work we plan to extend our study by considering heterogeneous requirements, such as threshold and optimization goals. We also plan to extend our empirical evaluation to other case studies in the literature, belonging to different domains.

REFERENCES

[1] Danny Weyns, Nelly Bencomo, Radu Calinescu, Javier Cámara, Carlo Ghezzi, Vincenzo Grassi, Lars Grunske, Paola Inverardi, Jean-Marc Jézéquel, Sam Malek, Raffaela Mirandola, Marco Mori, and Giordano Tamburrelli. Perpetual assurances for self-adaptive systems. In *Software Engineering for Self-Adaptive Systems III. Assurances - International Seminar, Dagstuhl Castle, Germany, December 15-19, 2013, Revised Selected and Invited Papers*, volume 9640 of *Lecture Notes in Computer Science*, pages 31–63. Springer, 2013.

[2] Sara Mahdavi-Hezavehi, Danny Weyns, Paris Avgeriou, Radu Calinescu, Raffaela Mirandola, and Diego Perez-Palacin. Uncertainty in self-adaptive systems: A research community perspective. *ACM Trans. Auton. Adapt. Syst.*, 15(4):10:1–10:36, 2020.

[3] Simona Bernardi, Michalis Famelis, Jean-Marc Jézéquel, Raffaela Mirandola, Diego Perez Palacin, Fiona A. C. Polack, and Catia Trubiani. *Living with Uncertainty in Model-Based Development*, pages 159–185. Springer International Publishing, Cham, 2021. ISBN 978-3-030-81915-6.

[4] Jeffrey O. Kephart and David M. Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.

[5] Danny Weyns, Bradley Schmerl, Vincenzo Grassi, Sam Malek, Raffaela Mirandola, Christian Prehofer, Jochen Wuttke, Jesper Andersson, Holger Giese, and Karl M. Göschka. *On Patterns for Decentralized Control in Self-Adaptive Systems*, pages 76–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-35813-5.

[6] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. Runtime equilibrium verification for resilient cyber-physical systems. In *2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS)*, pages 71–80, 2021.

[7] Radu Calinescu, Carlo Ghezzi, Kenneth Johnson, Mauro Pezzè, Yasmin Rafiq, and Giordano Tamburrelli. Formal verification with confidence intervals to establish quality of service properties of software systems. *IEEE Trans. Reliability*, 65(1): 107–125, 2016.

[8] Diego Perez-Palacin and Raffaela Mirandola. Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ICPE '14, pages 3–14, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2733-6.

[9] David Kaplan. On the quantification of model uncertainty: A bayesian perspective. *Psychometrika*, 86:215–238, 2021.

[10] M. Jeroen Van Der Donckt, Danny Weyns, M. Usman Iftikhar, and Ritesh Kumar Singh. Cost-benefit analysis at runtime for self-adaptive systems applied to an internet of things application. In *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2018, Funchal, Madeira, Portugal, March 23-24, 2018*, pages 478–490. SciTePress, 2018.

[11] Jeroen Van Der Donckt, Danny Weyns, Federico Quin, Jonas Van Der Donckt, and Sam Michiels. Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals. In *Proceedings of the IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '20, page 20–30, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379625.

[12] Nelly Bencomo and Luis H. Garcia Paucar. Ram: Causally-connected and requirements-aware runtime models using bayesian learning. In *2019 ACM/IEEE 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 216–226, 2019.

[13] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: A tutorial. *Statistical Science*, 14(4):382–401, 1999. ISSN 08834237.

[14] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. Bayesian model averaging: a tutorial (with comments by M. Clyde, David Draper and E. I. George, and a rejoinder by the authors. *Statistical Science*, 14(4):382 – 417, 1999.

[15] Max Hinne, Quentin Gronau, Don van den Bergh, and Eric-Jan Wagenmakers. A conceptual introduction to bayesian model averaging. *Advances in Methods and Practices in Psychological Science*, 3:251524591989865, 06 2020.

[16] BA Wintle, Michael Mccarthy, C. VOLINSKY, and Rod Kavanagh. The use of bayesian model averaging to better represent uncertainty in ecological models. *Conservation Biology - CONSERV BIOL*, 17:1579–1590, 12 2003.

[17] Tiago Fragoso, Wesley Bertoli, and Francisco Louzada. Bayesian model averaging: A systematic review and conceptual classification. *International Statistical Review*, 86:1–28, 04 2018.

[18] P. McCullagh and J.A. Nelder. *Generalized Linear Models*. Monographs on Statistics and Applied Probability. Springer US, 1983. ISBN 9780412238505.

[19] James O. Berger. *Statistical Decision Theory and Bayesian Analysis*. Springer Series in Statistics. Springer, 1985. ISBN 9780387960982.

[20] Christian P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer, 2nd edition, May 2007. ISBN 978-0-387-71598-8.

[21] Naeem Esfahani and Sam Malek. *Uncertainty in Self-Adaptive Software Systems*, pages 214–238. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. ISBN 978-3-642-35813-5.

[22] Dani Gamerman and Hedibert F Lopes. *Markov chain Monte Carlo: stochastic simulation for Bayesian inference*. CRC Press, 2006.

[23] Tao Chen, Ke Li, Rami Bahsoon, and Xin Yao. Femosaa: Feature-guided and knee-driven multi-objective optimization for self-adaptive software. *ACM Trans. Softw. Eng. Methodol.*, 27(2), jun 2018. ISSN 1049-331X.

[24] Richard J Rossi. *Mathematical statistics: an introduction to likelihood based inference*. John Wiley & Sons, 2018.

[25] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444.

[26] Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268–308, sep 2003. ISSN 0360-0300.

[27] Kalyanmoy Deb and Kalyanmoy Deb. *Multi-objective Optimization*, pages 403–449. Springer US, Boston, MA, 2014. ISBN 978-1-4614-6940-7.

[28] Cyril Goutte and Eric Gaussier. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In David E. Losada and Juan M. Fernández-Luna, editors, *Advances in Information Retrieval*, pages 345–359, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. ISBN 978-3-540-31865-1.

[29] Andres J. Ramirez, Adam C. Jensen, and Betty H. C. Cheng. A taxonomy of uncertainty for dynamically adaptive systems. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, SEAMS '12, pages 99–108, Piscataway, NJ, USA, 2012. IEEE Press. ISBN 978-1-4673-1787-0.

[30] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty H.C. Cheng, and Jean-Michel Bruel. Relax: Incorporating uncertainty into the specification of self-adaptive systems. In *2009 17th IEEE International Requirements Engineering Conference*, pages 79–88, 2009.

[31] Man Zhang, Shaukat Ali, Tao Yue, Roland Norgren, and Oscar Okariz. Uncertainty-wise cyber-physical system test modeling. *Software & Systems Modeling*, Jul 2017. ISSN 1619-1374.

[32] Matteo Camilli, Angelo Gargantini, and Patrizia Scandurra. Model-based hypothesis testing of uncertain software systems. *Softw. Test. Verification Reliab.*, 30(2), 2020.

[33] Matteo Camilli, Angelo Gargantini, Patrizia Scandurra, and Catia Trubiani. Uncertainty-aware exploration in model-based testing. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, pages 71–81, 2021.

[34] P. D. Arendt, D. W. Apley, and W. Chen. Quantification of model uncertainty: Calibration, model discrepancy, and identifiability. *J. Mech. Des.*, 134(10):100908, 2012.

[35] Pooyan Jamshidi, Amir Sharifloo, Claus Pahl, Hamid Arabnejad, Andreas Metzger, and Giovani Estrada. Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures. In *2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*, pages 70–79, 2016.

[36] Antonio Filieri, Carlo Ghezzi, and Giordano Tamburrelli. A formal approach to adaptive software: Continuous assurance of non-functional requirements. *Form. Asp. Comput.*, 24(2):163–186, March 2012. ISSN 0934-5043.

[37] Radu Calinescu, Yasmin Rafiq, Kenneth Johnson, and Mehmet Emin Bakir. Adaptive model learning for continual verification of non-functional properties. In *Proceedings of the 5th ACM/SPEC International Conference on Performance Engineering*, ICPE '14, pages 87–98, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2733-6.

[38] Matteo Camilli, Angelo Gargantini, Patrizia Scandurra, and Carlo Bellettini. Towards inverse uncertainty quantification in software development (short paper). In Alessandro Cimatti and Marjan Sirjani, editors, *Software Engineering and Formal Methods*, pages 375–381, Cham, 2017. Springer International Publishing. ISBN 978-3-319-66197-1.

[39] Karl J. Astrom and Bjorn Wittenmark. *Computer-controlled Systems: Theory and Design (2Nd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990. ISBN 0-13-168600-3.

[40] Antonio Filieri, Lars Grunske, and Alberto Leva. Lightweight adaptive filtering for efficient learning and updating of probabilistic models. In *Proceedings of the 37th International Conference on Software Engineering - Volume 1*, ICSE '15, pages 200–211, Piscataway, NJ, USA, 2015. IEEE Press. ISBN 978-1-4799-1934-5.

[41] Nelly Bencomo and Amel Belaggoun. A world full of surprises: Bayesian theory of surprise to quantify degrees of uncertainty. In *Companion Proceedings of the 36th International Conference on Software Engineering*, page 460–463. ACM, 2014.