RESEARCH-ARTICLE

Many-objective Self-adaptation under Model Uncertainty

MATTEO CAMILLI, Politecnico di Milano, Milan, MI, Italy

RAFFAELA MIRANDOLA, Karlsruhe Institute of Technology, Karlsruhe, Baden-Wurttemberg, Germany

PATRIZIA SCANDURRA, University of Bergamo, Bergamo, BG, Italy

Published: 12 June 2025
Online AM: 26 February 2025
Accepted: 06 February 2025
Revised: 11 December 2024
Received: 12 June 2024

Many-objective Self-adaptation under Model Uncertainty

MATTEO CAMILLI, Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano, Italy
RAFFAELA MIRANDOLA, Karlsruhe Institute of Technology, Karlsruhe, Germany
PATRIZIA SCANDURRA, Università degli Studi di Bergamo, Dalmine, Italy

The field of uncertainty quantification and mitigation in software-intensive and self-adaptive systems is garnering increased interest, especially with the rise of statistical inference methodologies like Bayesian reasoning. These methods typically address uncertain quality attributes embedded within system models by adjusting model parameters. However, the uncertainty related to selecting a specific system model over plausible alternatives has received limited attention. Our work focuses on self-adaptation, exploring methods to tackle uncertainty in model selection. This includes scenarios where one model is chosen over competing alternatives to encapsulate the system's understanding and anticipate future observations. Our proposed solution augments the conventional feedback loop of self-adaptive systems by combining Bayesian model averaging to mitigate uncertainty and many-objective optimization to take into account multiple, possibly many, dependability requirements at the same time.

We carry out an empirical evaluation to study the effectiveness, cost, and scalability of the proposed approach using two case studies with increasing structural complexity and number of dependability requirements. Results show that our approach based on model averaging is significantly better than model selection in terms of satisfied requirements after adaptation (adaptation success frequency). We also show that our approach can deal with large model spaces ($\sim 10^{19}$) using efficient sampling methods rather than exhaustive model space exploration.

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

Additional Key Words and Phrases: Self-adaptation, Model uncertainty, Bayesian model averaging, Manyobjective search

ACM Reference format:
Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2025. Many-objective Self-adaptation under Model Uncertainty. *ACM Trans. Autonom. Adapt. Syst.* 20, 2, Article 12 (June 2025), 28 pages.
https://doi.org/10.1145/3719349

The work has been partially founded by the topic Engineering Secure Systems of the Helmholtz Association (HGF), by the pilot program Core Informatics at KIT (KiKIT) of HGF, and by the PRIN project SAFEST (Trust assurance of Digital Twins for medical cyber-physical systems), funded by the European Union—Next Generation EU, Mission 4, Component 2, Investment 1.1, CUP F53D23004230006, under the National Recovery and Resilience Plan (NRRP)—Grant Assignment Decree No. 959 adopted on 30 June 2023 by the Italian Ministry of University and Research (MUR).
Authors' Contact Information: Matteo Camilli, Department of Electronics, Information and Bioengineering, Politecnico di Milano, Milano, Italy; e-mail: matteo.camilli@polimi.it; Raffaela Mirandola (corresponding author), Karlsruhe Institute of Technology, Karlsruhe, Germany; e-mail: raffaela.mirandola@kit.edu; Patrizia Scandurra, Università degli Studi di Bergamo, Dalmine, Italy; e-mail: patrizia.scandurra@unibg.it.

ACM 1556-4703/2025/6-ART12
https://doi.org/10.1145/3719349

1 Introduction

In recent years, there has been significant research focus on comprehending, quantifying, and mitigating uncertainty in software-intensive and adaptive systems [1–3]. Mainstream approaches for addressing uncertainty involve augmenting the analysis and planning stages of a feedback control loop architecture for adaptation, like the well-known loop **Monitor-Analyze-Plan-Execute over a Shared Knowledge (MAPE-K)** [4, 5], with model-based inference techniques. These techniques aim to predict future observations and plan actions, coupled with the computation of confidence or credible intervals, which account for uncertainties in parameter estimates of a given model. These approaches operate under the assumption that the "best" model is available or, somehow, has been selected [6, 7]. This assumption can sometimes foster overconfident inferences by neglecting the presence of multiple plausible alternative models describing the dynamics of a target system or, more in general, a phenomenon of interest [8]. This issue is known as *model-selection uncertainty* [9].

In self-adaptive systems, the aforementioned issue typically yields poor predictions that may lead to failures in recognizing the need for an adaptation or unnecessary adaptations affecting dependability attributes [10]. 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 (MS)** or definition step (e.g., [11, 12]). However, according to Kaplan [9], settling on a single model (even the best one) could involve the risk of neglecting possible unforeseen scenarios. A possible solution is the adoption of a **Bayesian Model Averaging (BMA)** [13]. This approach has been adopted in a range of disciplines to incorporate model-selection uncertainty into statistical inference and prediction [9, 13, 14–16], but, to the best of our knowledge, it has never been exploited to enhance analysis and plan activities in self-adaptive systems.

As an example, let us consider a delivery drone, which is an **Unmanned Aerial Vehicle (UAV)** designed to transport and deliver items (e.g., medicines, food, mail) to a target location. An operator notices that the expected delivery time is not respected and starts contemplating whether to wait for the delayed delivery or consider alternative actions. Various *scenario*s may have occurred (i.e., specific situations or set of conditions that may affect the outcome or behavior of the system). Perhaps there is a technical issue with the drone causing a significant delay ($scenario_1$). On the other hand, there might be a temporary weather disturbance, causing a short delay compared to the expectation ($scenario_2$). Alternatively, the delay could be due to an insufficient power level that might have led to a mission failure ($scenario_3$). 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 delay σ, denoted by $p(\sigma|m_i)$. The operator does not know the true scenario m, but what is relevant for the decision to continue waiting or to take action 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 [16]. Adopting BMA means changing the modeling 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.

In this article, we present **Taming Model UNcErtainty II (TUNE-II)**, an approach that extends our previous work [17]. The two main steps of TUNE-II are as follows:

(1) enhancement of the MAPE-K loop with components that consider not only the uncertainty about the parameters given a particular model but also uncertainty about the model structure.

To this end, for each dependability requirement, an ensemble of models is derived and a model averaging scheme is applied to predict possible requirement violations; and

(2) improvement of the adaptation decisions through many-objective metaheuristic optimization leveraging the BMA estimates.

TUNE-II supports the management of multiple dependability requirements simultaneously, it is completely automated and works at runtime along with the managed system in production. To show the applicability and benefits of TUNE-II, we have conducted an empirical evaluation to study its effectiveness, cost, and scalability using two different established case studies from existing literature: a self-adaptive search and rescue robotic system for emergency circumstances [18] and an autonomous team of UAVs that carry out a surveying mission in a hostile environment [19]. Compared to its predecessor [17], the main contributions of TUNE-II can be summarized as follows:

— extended methodology by considering the concurrent satisfaction of multiple (many), possibly independent requirements instead of a single one; and
— redesigned empirical evaluation that includes (i) additional **Research Questions (RQs)**, (ii) additional study subjects, (iii) statistical tests, and (iv) a quantitative comparison of TUNE-II with selected baseline methods.

The remainder of this article is as follows. In Section 2, we present an overview of TUNE-II with an illustrative example in the robotic domain. In Section 3, we introduce background concepts used in the forthcoming sections. The whole TUNE-II approach is described in Section 4. We illustrate our empirical evaluation in Section 5, and we discuss threats to validity, advantages, and shortcomings in Section 6. An overview of related work is presented in Section 7. We then draw our conclusion in Section 8.

2 Preliminaries

In this section, we provide the reader with a preview of TUNE-II, utilizing an illustrative example (i.e., an autonomous robot) that will be revisited in subsequent sections to illustrate major concepts.

2.1 Illustrative Example

We introduce here **Rescue Robot (RR)**. RR is a typical example of a self-adaptive search and rescue robotic system [18] for emergency circumstances such as fire, hurricane, or earthquake. As shown in Figure 1, the managed subsystem consists of sensors (e.g., camera, LiDAR) and actuators (e.g., motors and servo motors for motion and manipulation) embedded in the robot, and key control components to carry out the rescue tasks, such as the navigation as well as the obstacle/human detection via visual perception. In particular, the component *obstacle detector* applies an image processing function (e.g., using a Deep Neural Network for image classification) for detecting and classifying obstacles or human beings. This information is given in input to the *navigator*, which plans the steering angle, the acceleration, and the braking. These actions are implemented by the *controller* to steer or stop the robot safely. Since the environment and resource availability are dynamically changing and uncertain (e.g., heavy rain or presence of smoke, low bandwidth, low estimated battery life), the robotic system includes a *managing layer* embedding a typical MAPE-K feedback loop specifically for changing the robot's configuration and assuring compliance with both functional and dependability requirements at runtime.

The configuration space of the robotic system has multiple discrete and continuous factors. For example, *power* is a discrete configuration dimension for the controller that may switch from *energy saving* to *full power* mode and vice versa, while *cruise speed* is a continuous configuration

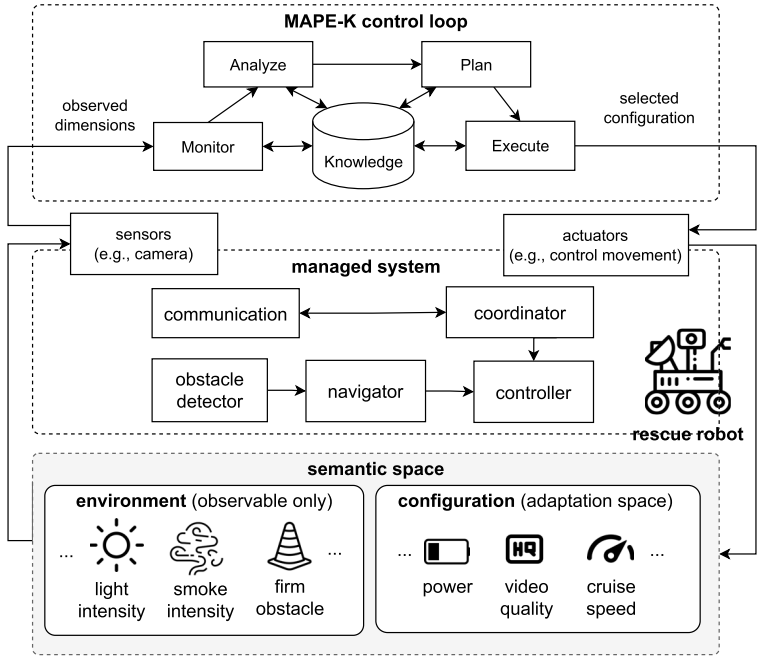


Fig. 1. Main components of the self-adaptive robotic system and its semantic space.

Table 1. Semantic Space of the RR example

Space	Variable	Type	Domain
Configuration	power	integer	[0, 100] %
	cruise speed	continuous	[0, 5] m/s
	bandwidth	continuous	[10, 50] Mbit/s
	quality	categorical	{*low*, *mid*, *high*}
Environment	illuminance	continuous	[40, 120,000] lux
	smoke intensity	categorical	{*none*, *thin*, *thick*}
	obstacle size	continuous	[0, 120] ft^3
	obstacle distance	continuous	[0, 10] m
	firm obstacle	Boolean	true/false

dimension for the navigator. The same considerations apply to the environmental factors, such as *illuminance*, *smoke intensity*, and so on. Hereafter, we refer to this set of all configuration and environment factors as *semantic space*. Table 1 lists all the factors that constitute the semantic space in our illustrative example; they are also roughly shown in Figure 1. Essentially, 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 example, *power* is an integer-valued configuration variable ranging in the percentage interval [0, 100], and *illuminance* is instead a continuous environment variable that ranges from 40 lux (for sunset/sunrise) to 120k lux (for brightest sunlight).

The actual configuration of the RR system's variables affects the ability to satisfy dependability requirements at the system level. For instance, relatively high cruise speed combined with adverse lighting conditions (low illuminance or presence of smoke) may lead to misclassification events from the vision system and, therefore, actuation of potentially unsafe maneuvers leading to violation of the following *safety* requirements:

R_1: "*Violations of the protective distance between the robot and human beings shall occur in less than 5% of the cases*"; and
R_2: "*Contacts between the robot and human beings shall occur in less than 1% of the cases.*"

In addition to safety concerns, the configuration may also impact energy consumption and, therefore, affect other kinds of requirements, such as *performance* (e.g., the controller can decide to operate in energy-saving or full power mode).

In general, the RR has multiple, possibly many, requirements affected by phenomena described by the semantic space. Some requirements may be independent, while others are interrelated. For example, the ability to maintain the protective distance (R_1) directly influences the likelihood of avoiding unwanted contacts (R_2). Predictive models can anticipate satisfaction or violation of requirements. However, the way these phenomena affect possible failures is typically uncertain without enough empirical evidence. Further, it may change over time. Therefore, the pre-selection of specific models without evidence may lead to an over-approximated description of the reality of interest and, therefore, poor predictions ultimately steering adaptation decisions toward wrong system configurations.

2.2 Preview of the Approach

The main elements of TUNE-II are incorporated into a traditional MAPE-K control loop, as shown in Figure 1. Our approach features custom components for knowledge, analysis, and planning, along with a specialized dataflow between these components. It employs a proactive and continual adaptation model [20]. The set of available adaptation alternatives depends on the semantic space of the system. For instance, the RR has two configuration variables (cruise speed and bandwidth) ranging in continuous domains. In this case, the adaptation space is continuous.

TUNE-II uses ensembles of **Generalized Linear Models (GLMs)** [21] as runtime predictive models stored in the shared knowledge, with each ensemble dedicated to a specific system-level dependability requirement. These ensembles are used by the analyze component to predict requirement violations based on the actual value of semantic space variables. We address model uncertainty through averaging, which considers multiple hypotheses having comparable likelihood concurrently rather than applying MS. To manage large model spaces, the analyze component uses efficient sampling methods based on **Markov Chain Monte Carlo (MCMC)** exploration [22] to select the models and estimate posterior probabilities through BMA. The resulting average models summarize ensemble information and are used to predict requirement violations. The predictive BMA analyzer triggers the plan component when a requirement violation is predicted with high probability. This planner employs metaheuristic optimization to find adaptation options that minimize the cost while maximizing the satisfaction likelihood of all requirements. TUNE-II supports handling multiple requirements simultaneously by creating and managing multiple model ensembles.

3 Background

This section recalls the necessary background concepts required to understand technical aspects of TUNE-II. The background includes GLMs, Bayesian inference, BMA, and many-objective optimization.

3.1 GLMs

We consider a class of statistical models that is a natural generalization of ordinary linear regression. GLMs [21] 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 McCullagh and Nelder [21], there exist canonical well-defined link functions. The model associated with Bernoulli and binomial distributions is the logistic regression. In this case, the canonical link function is the *logit* 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.

3.2 BMA

Bayesian inference is a popular framework for statistical inference [23]. The goal is to learn one (or more) uncertain/unknown parameters describing some details of a stochastic phenomenon of interest. To infer the parameters, we observe the phenomenon of interest and we collect observations to compute the conditional density of the quantity of interest given the likelihood of the observations and the prior knowledge.

Let us assume σ is the quantity of interest we need to estimate. BMA takes into account all the possible scenarios m_i that may affect the quantity of interest σ. The BMA estimate is obtained through the following equation:

$$p(\sigma) = \sum_i p(\sigma|m_i) \cdot p(m_i), \tag{4}$$

where $p(\sigma|m_i)$ and $p(m_i)$ are the likelihood function and prior knowledge, respectively, according to model m_i. BMA estimates are adjusted as new observations become available. Observations can be referred to collectively as *data*, and the BMA estimate becomes as follows:

$$p(\sigma|data) = \sum_i p(\sigma|m_i, data) \cdot p(m_i|data). \tag{5}$$

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 into account all possible models. The model-averaged estimate 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.

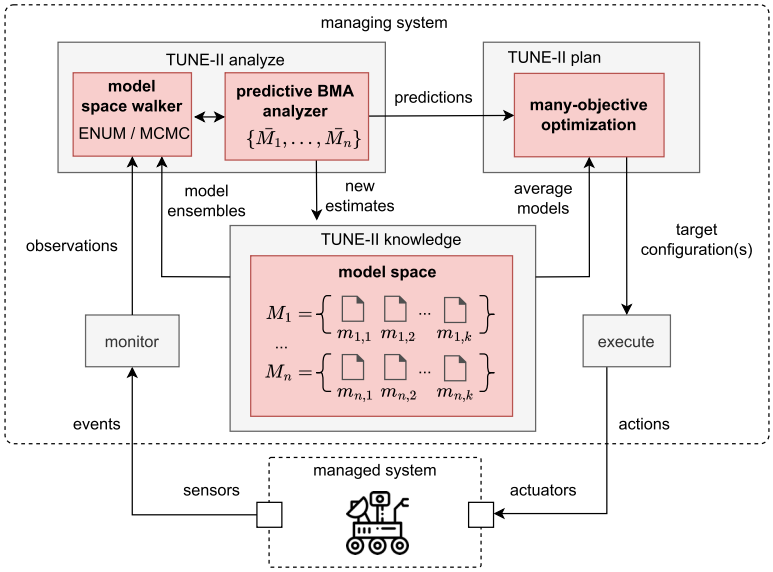


Fig. 2. TUNE-II within a MAPE-K adaptation control loop.

3.3 Many-objective Optimization

Many-objective optimization [24] refers to solving multiple objectives simultaneously using meta-heuristic evolutionary search algorithms. Evolutionary search is inspired by the principles of natural selection and genetics. They operate on a population of potential solutions, applying operators such as selection, crossover, and mutation to evolve the solutions over a given number of generations. The goal is to find a set of optimal or near-optimal solutions that satisfy the given objectives. These algorithms rely on multiple fitness functions (one per objective) that quantify the goodness of candidate solutions. The outcome is a set of solutions satisfying as many objectives as possible. Unlike traditional multi-objective optimization, which typically deals with two or three objectives at most, many-objective algorithms address scenarios with four or more objectives (typically up to 15).

Examples of mainstream algorithms include **Non-dominated Sorting Genetic Algorithm III (NSGA-III)** [25] and MOEA/D [26]. NSGA-III extends its predecessor NSGA-II (multi-objective version) that focuses on both convergence and diversity by balancing the trade-off between reaching the optimal front and maintaining a spread of solutions.

4 The TUNE-II Approach

In this section, we describe the adaptation mechanism of TUNE-II illustrated in Figure 2. As anticipated in Section 4, our approach combines BMA and many-objective optimizations and is integrated into a MAPE-K control loop. In the following, we describe the three main components: TUNE-II knowledge, analyze, and plan.

4.1 TUNE-II Knowledge

The knowledge component in Figure 2 stores and maintains ensembles of GLMs, specifically ensembles of logistic models employed as runtime predictors. More in detail, the knowledge maintains one ensemble for each system-level dependability requirement. Each ensemble contains some logistic models, each anticipating the ability to satisfy a given requirement (i.e., either positive

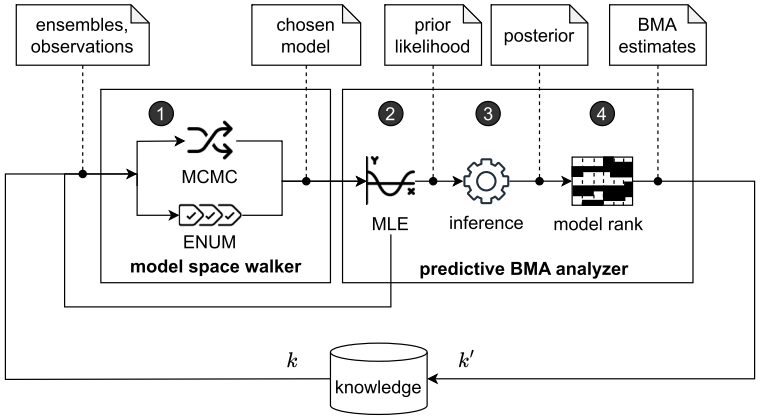


Fig. 3. Zoom into the analyze component of TUNE-II.

or negative outcome) according to Equation (3), given the actual value of the variables in the semantic space.

More precisely, the model space maintained by the knowledge includes the ensembles $\{M_1, ..., M_n\}$ associated with the corresponding set of system-level requirements $\{R_1, ..., R_n\}$. Each ensemble includes $k = 2^{|X|}$ possible models, that is, $M_i = \{m_{i,1}, ..., m_{i,k}\}$, where the model $m_{i,j}$ predicts the satisfaction of R_i by including in the predictor the *jth* subset of the explanatory variables in X.

The components of the MAPE-K loop can retrieve any model $m_{i,j}$ and make predictions based on the observation produced by the monitor. An observation is a vector of values assigned to the explanatory variables X. Given a model $m_{i,j}$, we use the notation $m_{i,j}(X) = Y$ to indicate the prediction for the corresponding requirement, that is, the likelihood of satisfying R_i. If the likelihood is greater than 0.5, the observation X belongs to the *positive* class (requirement satisfied), *negative* class (requirement violated) otherwise.

The knowledge component represents the model space, that is, for each requirement, the space of all possible logistic models that can be built based on the available set of explanatory variables. Intuitively, the size of the model space can quickly explode. For instance, our RR illustrative example has nine variables in the semantic space, thus $2^9 = 512$ models for each requirement. This would lead to a model space with $2.5k$ models with five requirements, $5.1k$ models with 10 requirements, and so on.

Hence, the TUNE-II knowledge does not necessarily store and maintain the whole model space explicitly. Especially in scenarios involving extensive sets of explanatory variables and numerous requirements, explicit enumeration would quickly become not practical.

To overcome this issue, TUNE-II can be configured to sample the model space rather than enumerate all possible models. Namely, our approach relies on MCMC exploration [22]. As further detailed in the following sections, TUNE-II uses MCMC to efficiently generate repeated random draws from the posterior distribution of model parameters.

4.2 TUNE-II Analyze

Figure 3 shows the overall process that analyzes the data and then feeds the knowledge back with new BMA estimates using the two main components: *model space walker*, in charge of steering the exploration of the model space, and *predictive BMA analyzer*, in charge of creating the average models $\{\bar{M}_1, ..., \bar{M}_n\}$ according to the observations.

Table 2. Example of Logistic Regression Results Using Maximum Likelihood Estimation

logit regression results			
#observations	**log-likelihood**	**LL-Null**	**LLR p-value**
225	−213.58	−260.19	2.701×10^{-19}
model parameters $\hat{\beta}$			
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

The *model space walker* receives as input the model ensembles from the knowledge. Each ensemble M_i is associated with the corresponding requirement R_i and each logistic model $m_{i,j}$ in a given ensemble is implicitly represented by the subset of explanatory variables used to make predictions. The component receives also observations, or simply *data* produced by the monitor. The data is a set of pairs, each one composed of values assigned to the explanatory variables X (semantic space) mapping to the corresponding response Y for all requirements. The response for a given requirement R_i is either positive or negative according to monitored phenomena.

The analysis process is driven by the *model space walker* which explores the model space using exhaustive enumeration or stochastic sampling through MCMC (step ❶). Each chosen model $m_{i,j}$ (either sampled or enumerated), along with the data under consideration, is given as input to the *predictive BMA analyzer*.

This latter component uses **Maximum Likelihood Estimations (MLE)** [27] to fit the parameters of the input model $m_{i,j}$ that most likely have generated the data. MLE (step ❷) finds the values of the model parameters $\hat{\beta}$ that maximize the likelihood function over the parameter space:

$$\hat{\beta} = \arg\max_{\beta} p(data \mid \beta, m_{i,j}). \tag{6}$$

Example 1 (MLE Results). Table 2 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 RR 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) 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 $\alpha = 0.005$. 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.

While the parameters of each model are estimated through MLE, the weighting scheme of all models belonging to an ensemble is developed under a Bayesian perspective. Namely, after MLE, the *predictive BMA analyzer* runs a Bayesian inference process (step ❸) to compute the posterior probability used to rank $m_{i,j}$. We leverage the **Bayesian Information Criterion (BIC)**, that

[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).

represents a quantity approximately proportional to the likelihood that the data is produced under the model $m_{i,j}$, as follows:

$$\text{BIC} \approx -2\ ln(p(data \mid m_{i,j})). \tag{7}$$

Notice that calculating the BIC involves the marginal likelihood of the data under the model $m_{i,j}$, that is $p(data \mid m_{i,j})$. This latter quantity is used by the *Bayes factor* to compare two alternative hypotheses $m_{i,j}$ and $m_{i,h}$, as follows:

$$BF[m_{i,j} : m_{i,h}] = \frac{p(data \mid m_{i,j})}{p(data \mid m_{i,h})}. \tag{8}$$

In case $BF[m_{i,j} : m_{i,h}] > 1$, the evidence strongly favors $m_{i,j}$ over $m_{i,h}$. This suggests that the BIC can be used to retrieve the most plausible model having the largest *log* of the marginal likelihood corresponding to the smallest BIC. Notice that picking up only the smallest BIC would lead to a selection process essentially ignoring the presence of model uncertainty. According to available data, other models may have similar BIC indicating comparable levels of support from the data and warranting equal consideration.

To account for the model uncertainty, our approach exploits the posterior probability of all the candidate models in the given ensemble M_i to produce a total order and quantify the extent to which the beliefs expressed by a given model are better or worse than others. The Bayes rule states that the posterior probability of each model $p(m_{i,j}|data)$ can be obtained by updating the prior probability $p(m_{i,j})$ 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, the *inference* step computes the posterior for each model approximating the marginal likelihood based on the BIC defined in Equation (7).

The last step of the *predictive BMA analyzer* is *model rank* (step ❹) in charge of quantifying the existing uncertainty by producing a total order in M_i according to the *log* of the *posterior odd* over the "null" hypothesis, henceforth denoted as model $m_{i,0}$. The posterior odd measures the likelihood of a given model compared to $m_{i,0}$ given the data as follows:

$$PO[m_{i,j} : m_{i,0}] = BF[m_{i,j} : m_{i,0}] \cdot O[m_{i,j}, : m_{i,0}], \tag{9}$$

where $O[m_{i,j} : m_{i,0}]$ represents the *prior odd* defined as the ratio $p(m_{i,j})/p(m_{i,0})$.

Example 2 (Model Uncertainty). Let us consider the RR example. The semantic space includes 9 potential explanatory variables (refer to Table 1), resulting in 512 possible models formed from all variable combinations. Figure 4 illustrates the model uncertainty after gathering 225 observations. The y-axis represents the variables (including the intercept), while the x-axis indicates the rank of the top 20 models based on posterior probability. Each vertical column denotes a distinct logistic model, with black areas representing the exclusion of certain variables from the model. For instance, the top-ranked model includes six variables: the *intercept*, *smoke intensity*, *obstacle size*, *firm obstacle*, *power*, and *cruise speed*. For instance, the first one in the rank includes six variables: the *intercept*, *smoke intensity*, *obstacle size*, *firm obstacle*, *power*, and *cruise speed*. The ranking is determined by sorting the models according to Equation (9). Here, we assume an uninformative prior, meaning each model starts with the same prior probability. Consequently, the prior odds are equal to 1, making the posterior odds equivalent to the Bayes factor in Equation (8). Colors in Figure 4 represent the log of the posterior probability. Models with similar colors have comparable posterior probability, allowing us to identify clusters of models that are equally likely given the current model uncertainty.

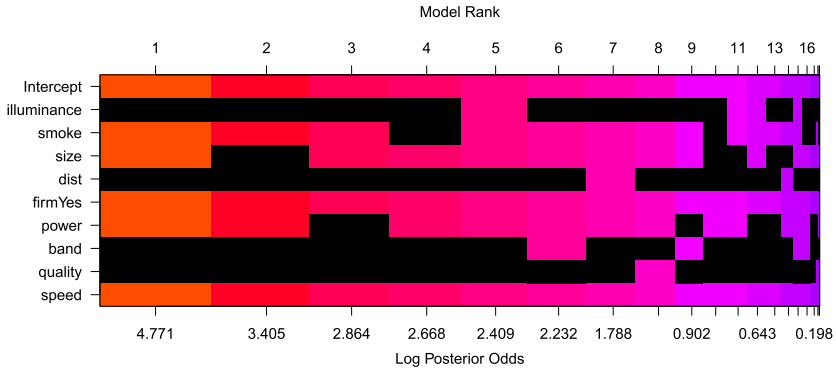


Fig. 4. Model uncertainty after 225 observations for requirement R_1 (see Section 2).

Table 3. 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

The posterior probability of each model is used to calculate BMA estimates according to Equation (5). Given a requirement R_i, the next prediction $\hat{Y}_i^*$ after collecting the data can be calculated as a weighted average of the prediction under each model $m_{i,j}$, with the posterior probability of $m_{i,j}$ being the weight. Models with higher posterior receive higher weights, while models with lower posterior receive lower weights.

Example 3 (BMA Estimates). Consider the model rank in Figure 4. According to the posterior probability of each model $m_{1,i}$, the BMA module computes the estimates reported in Table 3. 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 the violation of R_1 (see the illustrative example in Section 2). The assignment is as follows: { *power* = 52, *cruise speed* = 4.6, *bandwidth* = 28.6, *quality* = *low*, *illuminance* = 11,800, *smoke intensity* = *none*, *obstacle size* = 5.38, *obstacle distance* = 2.58, *firm obstacle* = No}. According to the logistic model in Table 2, 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 3. Using BMA, the prediction $\hat{Y}_1^*$ is 0.47, meaning that the average model is more pessimistic (i.e., probability lower than 0.5).

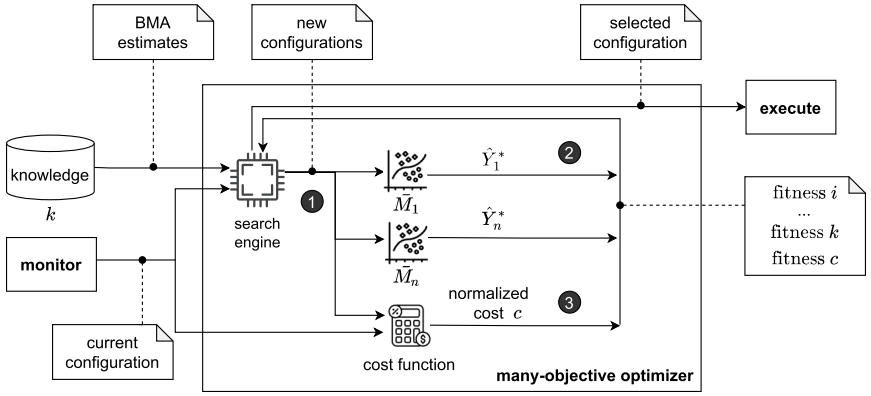


Fig. 5. Zoom into the plan component of TUNE-II.

When the total number of models is relatively small, the *model space walker* can exhaustively enumerate them all to calculate the BMA estimates. In general, we may have a large number of variables, which often leads to long computation time. In this case, the walker uses MCMC based on the *Metropolis-Hastings* algorithm [28]. In the following, we limit the description to a high-level overview and we let the reader refer to [28] 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. Given a requirement R_i, the procedure starts the exploration from an initial model $m_{i,j}$ and proceeds by randomly picking the next model $m_{i,k}$ and by checking whether this model improves the posterior probability computing the posterior odd $PO[m_{i,k} : m_{i,k}]$ based on the Bayes factor and the prior odd, as in Equation (9). If the value is greater than 1, $m_{i,k}$ improves the posterior compared to $m_{i,j}$. In this case, $m_{i,k}$ is included in the final sample. If the posterior odd is less than 1, the final sample still needs some representatives of this model. In this case, the posterior odd reflects the chance that $m_{i,k}$ is included in the final sample. After generating a number of models, the posterior of each model can be estimated by using the relative frequency of occurrence of the model in the sample.

4.3 TUNE-II Plan

Figure 5 illustrates the process of planning and making adaptation decisions based on current operating conditions and the knowledge augmented by the analysis. This plan leverages the *many-objective optimizer*, which uses BMA estimates from the knowledge base and current operating conditions from the monitoring component, represented by value assignments to explanatory variables. The whole plan process is initiated by the analysis component whenever at least one prediction $\hat{Y}_i^*$ indicates a probability of violating the requirement R_i greater than 0.5. In such cases, the *search engine* searches for alternative system configurations that maximize the probability of success and minimize associated. Configurations are defined by value assignments to configuration variables, which are a subset of explanatory variables that the system can control. Collectively, these variables form the adaptation space of the managed system. The *selected* configuration represents the chosen adaptation strategy, which is then executed in the subsequent step of the MAPE-K loop, as depicted in Figure 5. To make adaptation decisions (select new configurations), the *many-objective optimizer* employs a metaheuristic optimizing search [29], reducing the computational cost of evaluating all possible valid configurations.

As shown in Figure 5, the *search engine* generates a set of new values (step ❶) for all the configuration variables (e.g., modified cruise speed). The new values feed the average models

$\bar{M}_1, ..., \bar{M}_n$ which predict the probability of satisfying the corresponding requirements $R_1, ..., R_n$ (step ❷). Both the current and new configurations are input into the *cost calculator*, which computes the cost of transitioning the system from its current state to the newly selected configuration (step ❸). All probability values and the cost represent *fitness* scores that are fed back to the search engine. These scores guide the search towards better configurations in subsequent iterations. This iterative process continues until the given budget (in terms of time or iteration limit) is exhausted.

The *search engine* drives the search process using a many-objective *genetic algorithm* to create new configurations (individuals) using typical *crossover* and *mutation* operators to find better candidates while promoting diversity. The search is driven by the following objectives:

— *maximize* the predicted probability $\hat{Y}_i^*$ for all i; and
— *minimize* the cost of the adaptation.

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

$$cost(C, C') = \sum_i u_i \frac{|C_i - C'_i|}{ub(X_i) - lb(X_i)}, \tag{10}$$

with C and C' vectors representing the current and the new configurations, respectively. The values $ub(X_i)$ and $lb(X_i)$ represent the upper-bound and lower-bound of the corresponding variable X_i, while u_i is the weight for X_i defined by the cost profile. The *cost* function estimates the whole cost based on the weighted magnitude of the relative changes 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*. To avoid biased search, we normalize the range of *cost* in [*0, 1*] using min-max scaling.

Example 4 (*Adaptation*). Consider the BMA estimates from Section 3. In this case, one of the average models indicates a true negative prediction for requirement R_1 (see the illustrative example in Section 2), prompting the initiation of the adaptation process. Starting from the initial operating condition outlined in Section 3, the search engine executes the search process that emits a new configuration, such as {*power* = 52, *cruise speed* = 2.4, *bandwidth* = 28.9, *quality* = *low*}. According to the predictions of the corresponding average models, the new operating condition is expected to satisfy both R_1 and R_2. Let us consider, for example, requirement R_1. The posterior mean and the inclusion probability of *cruise speed* is high. This means that the substantial change in this variable (i.e., the robot slows down) has a strong impact on the prediction. According to the average model, this adaptation decision increases the probability of satisfying R_1 from 0.47 to 0.56.

5 Evaluation

In this section, we report on the empirical evaluation of TUNE-II.[2] We introduce our RQs and the design of the evaluation, present the major results, and finally discuss validity threats.

5.1 RQs

The purpose of the evaluation is to study the extent to which TUNE-II can improve the predictions of the analyze component and the adaptation decisions of the plan component. In particular, we aim to answer the following RQs:

[2]Replication package publicly available at https://zenodo.org/doi/10.5281/zenodo.11581557.

RQ1. What is the prediction accuracy of the analyze component of TUNE-II compared to MS?
RQ2. What is the effectiveness of the adaptation decisions taken by the plan component of TUNE-II compared to MS?
RQ3. What is the cost of calculating the BMA estimates in TUNE-II compared to MS?
RQ4. What is the scalability of the BMA estimates computed by TUNE-II?

5.2 Design of the Evaluation

5.2.1 Evaluation Subjects. To answer the RQs, we designed an experimental campaign to evaluate the cost-effectiveness of TUNE-II using two evaluation subjects. We selected existing benchmarks having nontrivial complexity in terms of size of the semantic space and number of requirements. We simulated multiple runs of the two subjects to sample the semantic space and observe the response, that is, a Boolean vector representing the satisfaction of the requirements.

We run our evaluation subjects using the same simulation platform for cyber-physical systems included in the RUNE framework [30]. The simulator enables the modeling of the stochastic behavior of a target system within a given scenario using state transitions annotated with probabilities that can dynamically adapt based on the semantic space. Formally, this specification is represented as a parametric Markov Decision Process [31], where the actual parameter values depend on configuration and environmental variables. The semantic space is defined as an arbitrary set of variables, each with its own domain. The simulator allows for customization of the effects of changing factors, enabling the simulation of complex, nontrivial requirement violations. Specifically, it uses the current values of variables in the semantic space as inputs to user-defined functions (e.g., non-linear, non-convex) that modify the parameters of the specification. These functions are referred to as mutation operators or simply *mutator*s. In this setting, requirements (formalized using temporal logic) can be automatically verified during a simulation through the RUNE framework. For further details about RUNE and the simulation platform, we refer the reader to Camilli et al. [6, 30].

Notice that TUNE-II is not limited to a specific simulator. The use of RUNE is a technical choice that does not alter the underlying method. In principle, TUNE-II could be integrated with any simulator capable of defining a scenario, specifying the semantic space, and observing the system response as a Boolean vector indicating the satisfaction of the requirements under a given setting (i.e., assignment to semantic space variables).

RR. The study subject RR represents the search and RR system introduced in Section 2. As anticipated above, RR supports emergency circumstances (e.g., fire, hurricane) to carry out rescue tasks in a target area. The robot navigates in the search area avoiding obstacles using vision sensors and LiDAR. Different factors in the semantic space affect the ability of the system to maintain the required level of safety. The semantic space of this subject has nine factors (see Table 1) including system properties (e.g., energy level, robot speed) and environment properties (e.g., lighting conditions, presence of smoke). In the RR system, we considered five probabilistic safety requirements predicting over different scenarios including human detection capability and obstacle avoidance. These requirements are listed in Table 4 using natural language.

UAV. This subject represents an autonomous team of UAVs originally introduced by Moreno et al. [19]. The vehicles carry out a surveying mission in a hostile environment. The objective of the team in the simulated scenario is to reach mission targets on the ground (through downward-looking sensors) and, at the same time, avoid threats (via forward-looking sensors). The distance from the team to existing threats affects the likelihood of detecting them, whereas the distance from the team to the ground affects the likelihood of detecting the targets, but also the likelihood of being damaged by a threat. The UAV subject has 12 requirements derived from the 2 templates T_0 and T_1 listed in Table 6 by replacing the 3 placeholders a_1, a_2, and p_1 with actual values. In our

Table 4. Requirements of the RR Evaluation Subject

Requirement	Scenario	Natural language statement
R_0	human detection	When the robot is moving in the search area, misclassification of human bodies shall occur in less than 2% of the cases.
R_1	human detection	Violations of the protective distance between the robot and human beings shall occur in less than 5% of the cases.
R_2	human detection	Contacts between the robot and human beings shall occur in less than 1% of the cases.
R_3	obstacle avoidance	Whenever an obstacle has been detected, a crash between the robot and the obstacle shall occur in less than 1% of the cases.
R_4	obstacle avoidance	Whenever an obstacle has been detected, it shall be avoided without reaching a critical distance in more than 95% of the cases.

Table 5. Semantic Space of the UAV Evaluation Subject

Variable	Space	Type	Domain
formation	configuration	categorical	{*tight*, *loose*}
flying speed	configuration	continuous	[5, 50] mph
electronic countermeasure	configuration	Boolean	Yes/No
weather	environment	categorical	{*sun*, *clouds*, *rain*, *fog*}
day time	environment	discrete	[0:00 am, 11:59 pm]
threat range	environment	continuous	[0.9, 3.7] km
#threats	environment	discrete	[1, 10]

Table 6. Requirements of the UAV Evaluation Subject

Requirement template	Scenario	Natural language statement
T_0	target detection	When the altitude is in between «a_1» and «a_2» meters, the team of UAVs shall eventually detect the target in more than «p_1» % of the cases.
T_1	threat avoidance	When the altitude is in between «a_1» and «a_2» meters, existing threats shall eventually detect the presence of the UAVs in less than «p_2» % of the cases.

experiments, we defined six different altitude ranges $[a_1, a_2]$ (meters) and corresponding likelihood values p_1 and p_2 (in percentage). The semantic space, illustrated in Table 5, has seven factors in total. It includes again system properties (e.g., flying speed, team formation) and environment properties (e.g., weather conditions, number of threats).

5.2.2 Methods under Comparison. We executed the analyze and the plan component of TUNE-II[3] (based on BMA) multiple times by simulating all evaluation subjects 1,000 times.[4] The analyze component of TUNE-II is currently implemented using `R`. In particular, model averaging is implemented using the packages `BMA` [32] and `BAS` [33]. The plan component is written in Python. Many-objective optimization is currently carried out using the library `pymoo` [34]. Further details on the software and data necessary to run our experiments can be found in our replication package.

We address RQ1 by comparing the predictive ability of BMA estimates and the best models, identified through MS. We carry out MS by comparing common metrics to evaluate the predictive ability in classifying satisfied/unsatisfied requirements, as further detailed in the next

[3]Note that comparing the results of TUNE-II and its predecessor TUNE is inherently unfair, as TUNE does not support multiple requirements. Any meaningful comparison shall be limited to a simplified scenario where system subjects have only one requirement. This scenario represents an edge case in which TUNE-II and TUNE behave in the same way.
[4]We consider each simulation to have one time step (i.e., an entire run from the initial to a final state) that allows us to assess all the requirements.

section. We use model predictions as input to run the plan component of TUNE-II to study and compare the effectiveness of the adaptation decisions. Thus, we answer RQ2 by assessing the effectiveness of TUNE-II compared to decisions guided by MS as well as random adaptation decisions. To answer RQ3, we compute the cost of calculating the BMA estimates by measuring the execution time required to produce BMA estimates and carry out MS. We finally address RQ4 by studying the scalability of BMA varying the sample size and the number of variables in the semantic space.

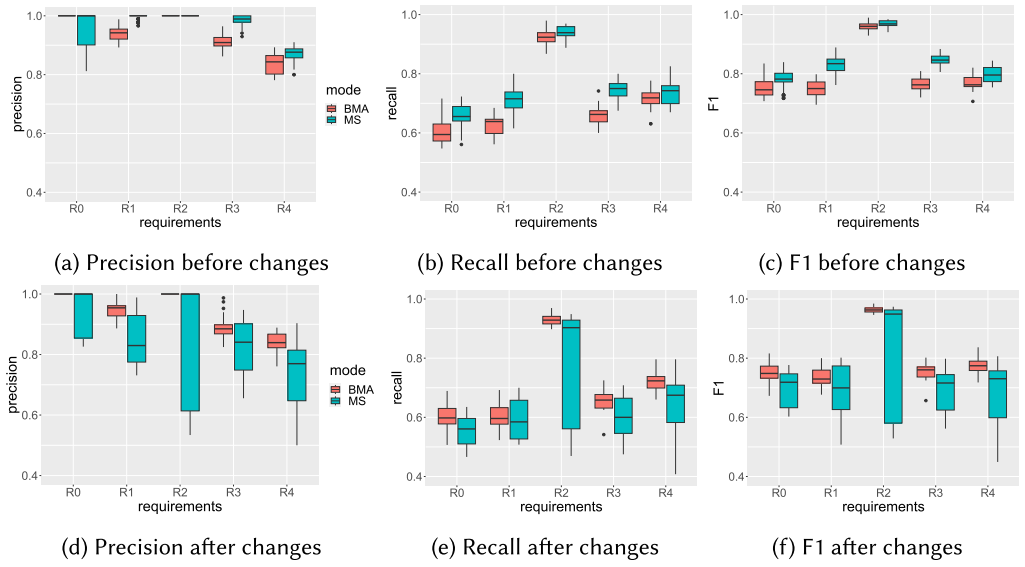
5.2.3 Statistical Tests. To reduce the risk of obtaining results by chance, we account for randomness by running our evaluation subjects multiple times (1,000 simulations for each subject and each method under comparison) to collect a large sample size when comparing the cost-effectiveness of BMA and MS. According to the guideline introduced by Arcuri and Briand [35], we apply Mann-Whitney U tests [36] to evaluate the statistical significance of the results. We interpret the results using a common significance level $\alpha = 0.05$. We also measure Vargha and Delaney's $\hat{A}_{AB}$ [37] to compute the effect size of the difference between A and B. We adopt the following standard classification: effect size $\hat{A}_{AB}$ ($= 1 - \hat{A}_{BA}$) is small, medium, and large when its value is greater than or equal to 0.56, 0.64, and 0.71, respectively.

5.2.4 Testbed. All the experiments have been conducted by using a commodity hardware machine equipped with an Apple M1 Pro chip (6 high-performance cores clocked at 3,228 MHz, and 2 high-efficiency cores clocked at 2,064 MHz), and 16 GB 2,133 MHz LPDDR5 RAM.

5.3 Results

5.3.1 RQ1 (Accuracy of BMA Estimates).
Setup. To address RQ1 we execute the analyze component of TUNE-II (based on BMA) and we compare it against a baseline method relying on MS to select a single (best) model to make predictions. We compare the alternative approaches for each evaluation subject and each requirement under two operational settings: *before changes* and *after changes*.

The setting *before changes* is as follows. For each evaluation subject and each requirement, we build all possible logistic models (512 for each requirement of RR and 128 for each requirement of UAV) fitted using MLE as well as the BMA estimates using the same training and test sets composed of 80% of the observations (800 in total). It is worth noting that the dataset employed in this context is created through simulations that run the evaluation subjects consistently influenced by the semantic space under steady mutators (i.e., mutation functions that do not change over time). We conduct MS by assessing the accuracy of all competing logistic models by comparing the *F1 score* [38] computed using stratified 10-fold cross-validation. After obtaining the best predictive model MS for each requirement, we compare the accuracy of both MS and BMA by computing *precision*, *recall*, and *F1 score* using a test set composed of 20% of the observations (200 in total). In our context, precision indicates the number of times a given requirement is correctly classified as satisfied (i.e., true positives) divided by the total number of positive outcomes in the test set. The recall measures the number of true positives divided by the sum between true positives and false negatives (i.e., requirements wrongly classified as 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).



(a) Precision before changes (b) Recall before changes (c) F1 before changes

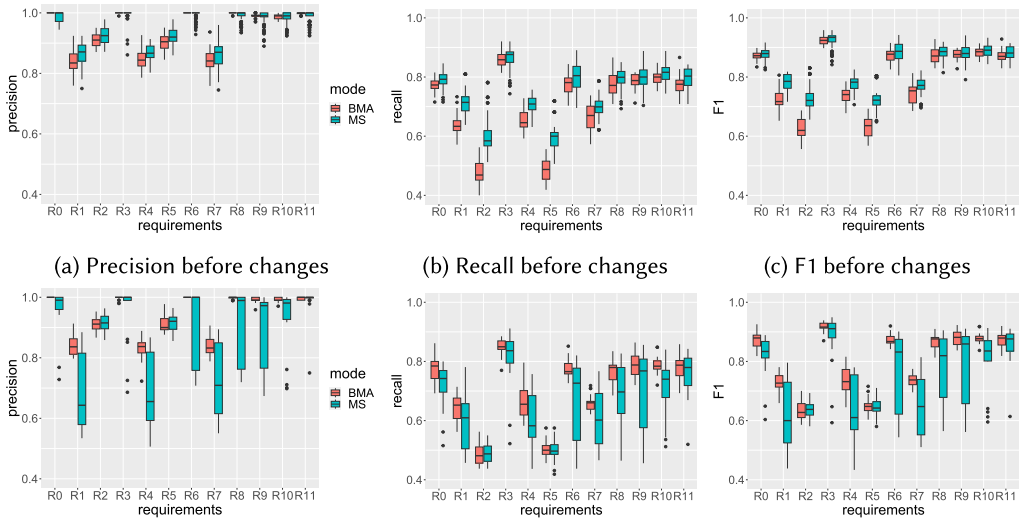
(d) Precision after changes (e) Recall after changes (f) F1 after changes

Fig. 6. Prediction accuracy for RR evaluation subject. (a) Precision before changes. (b) Recall before changes. (c) F1 before changes. (d) Precision after changes. (e) Recall after changes. (f) F1 after changes.

We systematically recreate the setting *after changes* as follows. We simulate again the evaluation subjects (1,000 simulations in total) by injecting random changes in the mutators after 400 simulations (50% of the training set). For each evaluation subject and each requirement, we build the BMA estimates and the model MS (i.e., the best model selected within the setting *before changes*) following the same procedure reported above. Thus, we compare the accuracy of the new models MS and BMA by computing the same metrics: *precision*, *recall*, and *F1 score*.

Results. Figures 6 and 7 show the results collected for all requirements of the evaluation subjects RR and UAV, respectively, under the two alternative settings *before changes* and *after changes*. Tables 7 and 8 list statistical significance results and effect size of the difference between BMA and MS under the experimental setting *before* and *after changes*, respectively. Overall, we can observe that, before changes, all scores (precision, recall, F1) of BMA and MS are very close to each other for all requirements in both RR and UAV. Even though in some cases, MS achieves higher median values, we can observe negligible to small effect size only for both RR and UAV subjects (often with statistically significant results). After changes, we can observe a sharp decrease in the scores associated with MS whereas BMA consistently maintains higher scores with smaller statistical dispersion. Statistical tests in Table 8 confirm we can achieve a significantly large effect size. In RR, the comparison of precision, recall, and F1 score yields a significantly large effect size in 60%, 60%, and 80% of the cases, respectively. Concerning UAV, the comparison of precision, recall, and F1 score yields a significantly large effect size in 66%, 41%, and 50% of the cases, respectively.

Results indicate that MS is in general slightly (not significantly) better than BMA assuming there are no changes after training the predictive models. In case of changes, precision, recall, and F1 scores obtained by using the BMA estimates are consistently and significantly higher than the scores of MS. This means model averaging adopted by TUNE-II is always superior to MS in case the satisfaction of requirements is affected by non-steady functions of the factors in the semantic space.



(a) Precision before changes (b) Recall before changes (c) F1 before changes

(d) Precision after changes (e) Recall after changes (f) F1 after changes

Fig. 7. Prediction accuracy for UAV evaluation subject. (a) Precision before changes. (b) Recall before changes. (c) F1 before changes. (d) Precision after changes. (e) Recall after changes. (f) F1 after changes.

Table 7. Statistical Comparison (BMA vs. MS) of Accuracy Metrics under the Setting *Before Changes*

subject	metric		requirements											
			R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}
RR	precision	p-value	1.10e-2	1.63e-21	NA	7.85e-14	3.43e-05	-	-	-	-	-	-	-
		$\hat{A}_{AB}$	0.62	0.00	0.50	0.01	0.22	-	-	-	-	-	-	-
	recall	p-value	4.27e-7	5.73e-12	8.25e-3	6.54e-12	7.83e-2	-	-	-	-	-	-	-
		A	0.15	0.034	0.32	0.04	0.38	-	-	-	-	-	-	-
	F1 score	p-value	3.03e-5	5.65e-13	8.25e-3	2.02e-13	8.83e-06	-	-	-	-	-	-	-
		$\hat{A}_{AB}$	0.22	0.01	0.32	0.01	0.19	-	-	-	-	-	-	-
UAV	precision	p-value	1.90e-3	3.12e-3	1.10e-2	6.88e-1	8.54e-3	7.50e-3	1.38e-2	3.95e-2	2.60e-2	3.11e-1	8.63e-1	2.03e-1
		$\hat{A}_{AB}$	0.67	0.29	0.33	0.51	0.32	0.32	0.62	0.36	0.62	0.43	0.48	0.57
	recall	p-value	7.25e-3	4.57e-09	2.68e-12	7.37e-2	3.91e-07	2.73e-12	1.18e-2	1.85e-3	6.63e-3	9.56e-2	4.14e-2	3.40e-3
		$\hat{A}_{AB}$	0.32	0.09	0.02	0.37	0.15	0.02	0.32	0.28	0.31	0.38	0.36	0.29
	F1 score	p-value	2.26e-1	4.19e-8	1.11e-12	8.45e-2	4.64e-08	2.10e-12	3.01e-2	5.52e-4	4.13e-2	1.43e-1	1.53e-1	2.49e-2
		$\hat{A}_{AB}$	0.41	0.12	0.01	0.38	0.12	0.01	0.35	0.26	0.36	0.39	0.40	0.34

Gray cells indicate statistically significant results, while boldface denotes large effect size.

RQ1 Summary. Under the setting *before changes*, statistical comparison of the distributions yields negligible to small effect size only. *After changes*, the effect size is significantly large in many cases for both RR and UAV. This means model averaging adopted by TUNE-II is superior to MS in case the satisfaction of requirements is affected by non-steady functions of the factors in the semantic space.

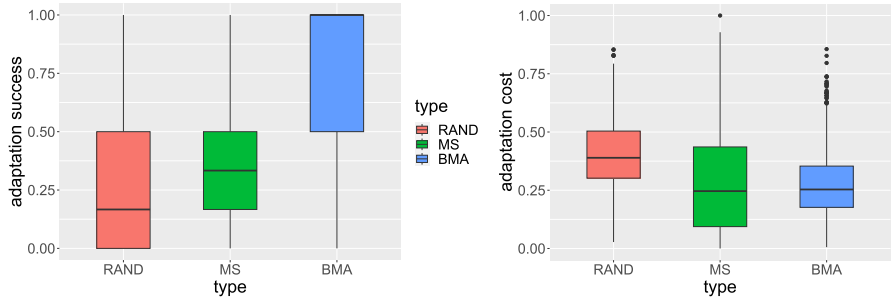
5.3.2 *RQ2 (Effectiveness of Adaptation Decisions).*

Setup. To answer RQ2, we use the setting *after changes* adopted in RQ1 to build BMA estimates and also train the best logistic model MS selected under the setting *before changes* for all requirements and all evaluation subjects. We evaluate the effectiveness of adaptation decisions made by the

Table 8. Statistical Comparison (BMA vs. MS) of Accuracy Metrics under the Setting *After Changes*

subject	metric		requirements											
			R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}
RR	precision	p-value	9.30e-4	1.16e-3	9.57e-3	4.98e-2	4.19e-4	-	-	-	-	-	-	-
		$\hat{A}_{AB}$	**0.72**	**0.80**	0.65	0.68	**0.81**	-	-	-	-	-	-	-
	recall	p-value	9.27e-3	4.16e-1	6.40e-3	2.53e-2	1.72e-3	-	-	-	-	-	-	-
		$\hat{A}_{AB}$	**0.74**	0.57	**0.75**	0.71	**0.79**	-	-	-	-	-	-	-
	F1 score	p-value	4.09e-3	1.23e-1	6.40e-3	5.55e-3	5.30e-4	-	-	-	-	-	-	-
		$\hat{A}_{AB}$	**0.76**	0.64	**0.75**	**0.75**	**0.81**	-	-	-	-	-	-	-
UAV	precision	p-value	4.01e-4	1.28e-3	5.33e-1	2.22e-1	1.86e-3	6.26e-1	9.29e-4	1.09e-2	1.02e-2	3.38e-5	7.61e-3	5.81e-1
		$\hat{A}_{AB}$	**0.75**	**0.79**	0.44	0.59	**0.78**	0.45	**0.72**	**0.73**	**0.72**	**0.88**	**0.74**	0.45
	recall	p-value	6.12e-3	2.13e-2	8.38e-1	1.65e-1	1.65e-2	9.13e-1	3.91e-2	8.18e-2	2.12e-2	8.76e-2	2.86e-3	5.32e-1
		$\hat{A}_{AB}$	**0.75**	**0.72**	0.48	0.63	**0.72**	0.51	0.69	0.66	**0.72**	0.66	**0.78**	0.55
	F1 score	p-value	1.44e-3	1.43e-2	6.74e-1	1.66e-1	6.82e-3	9.03e-1	1.52e-2	4.52e-2	1.27e-2	2.83e-2	2.22e-3	5.15e-1
		$\hat{A}_{AB}$	**0.79**	**0.72**	0.46	0.63	**0.75**	0.51	**0.73**	0.68	**0.73**	0.70	**0.78**	0.56

Gray cells indicate statistically significant results, while boldface denotes large effect size.

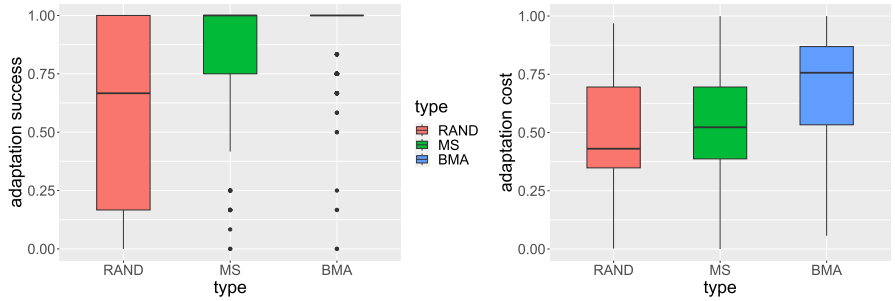


(a) Success frequency of adaptations (b) Normalized cost of adaptations

Fig. 8. Success and cost of the adaptation decisions for RR evaluation subject. (a) Success frequency of adaptations. (b) Normalized cost of adaptations.

planning component of TUNE-II (adopting BMA) against alternative baseline methods, namely random search, and MS. For each evaluation subject, we consider 100 initial configurations where all requirements are unsatisfied. Effectiveness is assessed by measuring the percentage of requirements fulfilled following the execution of adaptation decisions in new simulations of the evaluation subjects. Additionally, we assess the cost of enacting adaptation decisions by computing the total (normalized) relative difference between the initial and the actuated adaptations.

Results. Figures 8 and 9 show the results for the evaluation subjects RR and UAV, respectively. Table 9 lists statistical significance results and effect size of the difference between BMA and RAND, MS and RAND, and then BMA and MS. Concerning the success frequency in RR (Figure 8(a)) and UAV (Figure 9(a)). The RAND baseline is the worst option. MS generally yields a higher frequency compared to RAND with varying effect size values (from small to large). BMA is significantly better than RAND and MS. In both evaluation subjects, the median success frequency of BMA is close to 1.0 (with compact boxes and long whiskers). The difference between BMA and MS is always statistically significant, whereas the effect size ranges from small (in UAV) to large (in RR). Concerning the cost of actuating the adaptation decisions, we can see that in both cases, RR (Figure 8(b)) and UAV (Figure 9(b)), the difference between BMA and the two baselines is not significant. In some cases, we can observe that BMA may lead to higher adaptation costs. Indeed, optimal solutions (maximizing the likelihood of satisfying the requirements) may be far from the starting



(a) Success frequency of adaptations (b) Normalized cost of adaptations

Fig. 9. Success and cost of the adaptation decisions for UAV evaluation subject. (a) Success frequency of adaptations. (b) Normalized cost of adaptations.

Table 9. Statistical Comparison (RAND vs. BMA vs. MS) of Success Frequency and Cost of the Adaptations

Subject	*A*	*B*	Metric	p-value	$\hat{A}_{AB}$
RR	BMA	RAND	success frequency	8.52e-144	**0.82**
			cost	9.25e-74	0.26
	MS	RAND	success frequency	7.18e-06	0.57
			cost	1.86e-29	0.31
	BMA	MS	success frequency	2.60e-73	**0.77**
			cost	0.20	0.52
UAV	BMA	RAND	success frequency	6.61e-80	**0.80**
			cost	4.38e-69	**0.79**
	MS	RAND	success frequency	2.72e-38	**0.74**
			cost	6.37e-08	0.60
	BMA	MS	success frequency	6.56e-06	0.58
			cost	1.86e-19	0.70

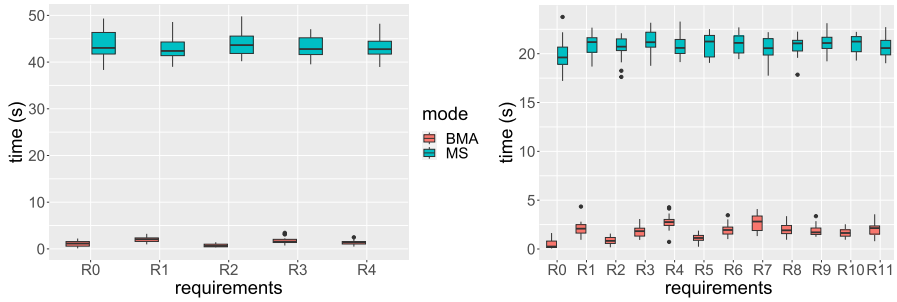
Gray cells indicate statistically significant results, while boldface denotes large effect size.

configurations. This means that model averaging adopted by TUNE-II is superior to both RAND and MS. However, higher effectiveness (in terms of success frequency) does not necessarily yield lower costs.

RQ2 Summary. The effectiveness of BMA in terms of adaptation success frequency is always significantly better than RAND (large effect size) and MS (from small to large effect size) in both evaluation subjects RR and UAV. Concerning the cost of actuating the adaptation decisions, the difference between BMA and the two baselines is not significant. In some cases, such as the UAV subject, BMA yields a slightly (but not significantly) higher cost compared to MS and RAND.

5.3.3 *RQ3 (Cost of BMA Estimates).*

Setup. We address RQ3 by measuring the cost required by the analyze component of TUNE-II to build BMA models and the cost of carrying out MS after computing all possible logistic models available in the model space. We build the models by considering, for each evaluation subject and all requirements, all the simulations executed under the setting *before changes* (1,000 data points) and



(a) Cost for RR subject (b) Cost for UAV subject

Fig. 10. Cost of MS and BMA estimates in terms of execution time. (a) Cost for RR subject. (b) Cost for UAV subject.

Table 10. Statistical Comparison (BMA vs. MS) of Cost Results

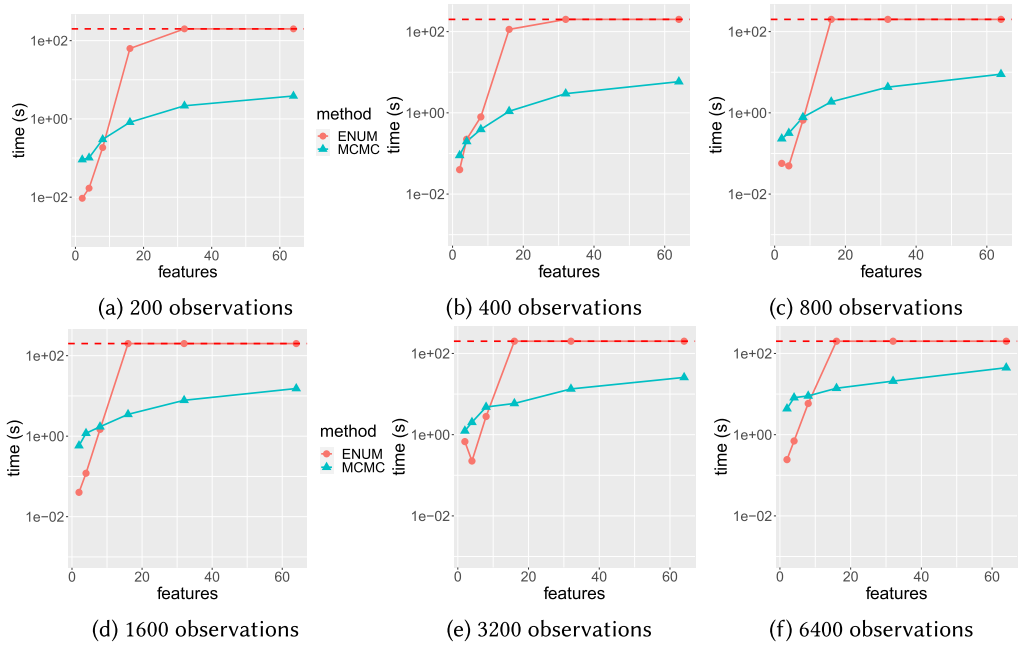
subject	metric		requirements											
			R_0	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}
RR	time	p-value	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11							
		$\hat{A}_{AB}$	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**							
UAV	time	p-value	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11	1.45e-11
		$\hat{A}_{AB}$	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**

Gray cells indicate statistically significant results, while boldface denotes large effect size.

then *after changes* (other 1,000 data points). We repeat the experiment 100 times and we measure the cost as the wall-clock time (seconds).

Results. Figure 10 shows the results collected for the evaluation subject RR (Figure 10(a)) and UAV (Figure 10(b)). We can see that the execution time required by computing BMA estimates is consistently lower for all requirements in both subjects. Concerning the subject RR, the median cost of MS is around 43 seconds for each requirement, while the median cost of BMA is less than 5 seconds for each requirement. Considering the subject UAV, we can observe a smaller gap. For each requirement, the median cost of MS is around 22 seconds, while the median cost of BMA is again less than 5 seconds. Indeed, the subject RR has more features than UAV (nine in the former case, seven in the latter one). This means the model space of RR is larger, and therefore, the cost associated with MS becomes even higher compared to BMA. For UAV, the cost of MS is three times higher than BMA. For RR, the cost of MS is seven times higher than BMA. We can see that the relative difference between the cost of BMA and MS increases by a factor of 1.3 on average when passing from UAV to RR. Table 10 shows the statistical significance and effect size of the difference. Results confirm that the cost of BMA is always significantly lower than MS with effect size equal to 1.0 for all requirements and all subjects.

RQ3 Summary. Results reveal a consistently and significantly lower execution time for BMA estimates across all requirements and subjects. In both RR and UAV, the median cost of MS exceeds 43 seconds and 22 seconds, respectively, while BMA costs remain consistently under 5 seconds, with the relative difference between BMA and MS costs increasing by a factor 1.3 when passing from UAV (7 features) to RR (9 features).



(a) 200 observations (b) 400 observations (c) 800 observations

(d) 1600 observations (e) 3200 observations (f) 6400 observations

Fig. 11. Scalability of BMA considering exhaustive enumeration and MCMC sampling. (a) 200 observations. (b) 400 observations. (c) 800 observations. (d) 1,600 observations. (e) 3,200 observations. (f) 6,400 observations.

5.3.4 *RQ4 (Scalability of BMA Estimates).*

Setup. To address RQ4, we use TUNE-II to compute BMA models multiple times by controlling the number of features in the semantic space (i.e., size of the model space), and the number of observations used to fit the individual logistic models. We change the number of features from 2 to 64 (4 to 1.8×10^{19} models) and the number of observations from 200 to 25,600. We repeated each experiment 100 times and we measured the cost as the wall-clock time (seconds) for each execution with a timeout set at 200 seconds.

Results. Figure 11 shows the execution time of BMA using two different strategies: ENUM and MCMC. The ENUM strategy yields a complete enumeration of all possible logistic models and subsequent averaging as described in Section 4. The strategy MCMC is instead an approximation particularly useful when the number of models in the model space is relatively large. The method makes use of MCMC sampling [22]. Figure 11 shows the cost of the two alternative strategies under different (increasing) numbers of observations used to fit the models: from 200 observations (Figure 11(a)) to 6,400 observations (Figure 11(f)). In all plots, the horizontal dashed line represents the time out (200 seconds) after which the process computing the estimates (either ENUM or MCMC) is killed.

Overall, we can observe that for low-dimensional model spaces (from two to eight variables) the cost is generally lower when using ENUM (exhaustive enumeration), even when increasing the number of observations. In this case, 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. The MCMC strategy yields better scalability. With relatively high-dimensional model spaces (from 16 to 64 features), we can observe that the exhaustive exploration leads to higher costs, exceeding the timeout in almost

all the cases. The MCMC never reaches the timeout even in the worst case, when the model space has an order of magnitude of 10^{19}.

RQ4 Summary. Considering relatively low-dimensional model spaces (up to eight variables), the cost is generally lower when using the ENUM strategy (exhaustive enumeration of all models). MCMC strategy yields better scalability and it never reaches the given timeout (200 seconds). Under high-dimensional model spaces ($\sim 10^{19}$ models), the computation terminates with execution time varying from 3.8 to 147.6 seconds when increasing the observations from 200 to 25,000.

6 Discussion

6.1 Threats to Validity

We mitigated *construct validity* threats by assessing the appropriateness of our claims based on our measurements. We measured the prediction accuracy using common quality metrics in this context: precision, recall, and F1 score. Success frequency in achieving adaptation goals (e.g., satisfaction of system-level requirements) represents a common metric to quantify the effectiveness of adaptation decisions. The execution time is again a natural measure of the cost, especially in critical domains, such as robotics and autonomous vehicles.

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 features in the semantic space and corresponding domains, (steady/unsteady) mutators, and the number of observations. This direct manipulation has been fundamental in assessing cause-effect relations between external factors and the cost-effectiveness 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 systems in our selected evaluation subjects are not indicative of the characteristics of other systems. We limited these threats by adopting existing benchmarks having a nontrivial semantic space commonly adopted by the research community in software engineering for self-adaptive systems. For practical reasons, we conducted the evaluation by simulating our evaluation subjects. The application of our approach to additional evaluation subjects in other domains would increase the generalizability of our results.

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 have been measured by a large number of observations (1,000 observations). Following the guideline introduced by Arcuri and Briand [35], we repeat the experiments multiple times (100 times) assessing both statistical significance and effect size using the Mann–Whitney U tests and the Vargha and Delaney's $\hat{A}_{AB}$ effect size measure.

6.2 Advantages and Shortcomings

According to our experience, TUNE-II can be effectively used to account for, and then mitigate, model uncertainty. The approach offers some advantages as follows.

- *Addressing model uncertainty*: TUNE-II addresses the limitations of traditional MS methods that overly simplify the problem by ignoring model uncertainty and assuming that only one model has any probability of being correct.
- *Enhancing hypothesis testing*: Model averaging, as implemented in TUNE-II, enhances the classical hypothesis testing approach, which tends to accept or reject models entirely (the

"all-or-nothing" mentality). Instead, TUNE-II retains and evaluates all plausible models, ranking them to provide a more nuanced final inference.
— *Robust decision-making*: BMA improves decision-making by making it robust to model misspecification. Relying on a single model assumes 100% confidence in its correctness, which is rarely realistic. In contrast, TUNE-II considers a range of competing models, increasing the likelihood that at least one model will be approximately correct.
— *Flexibility and responsiveness*: TUNE-II improves upon traditional MS methods, which can fail when there are sudden changes in observations that suggest a different best model. By considering a suite of models, TUNE-II remains flexible and responsive to such changes.
— *Dynamic updates*: TUNE-II dynamically updates estimates in the shared knowledge base as new data accumulates. Continually adjusting model weights, prevents model lock-in, ensuring adaptability to changing circumstances.
— *Handling many requirements*: TUNE-II maintains multiple (possibly many) ensembles to take into account several independent dependability requirements at the same time to steer optimal adaptation decisions.
— *Automation and real-time operation*: TUNE-II is fully automated and operates in real-time alongside the managed system in production, eliminating the need for extensive offline (re)training stages.

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

— *Risk of including incorrect models*: Maintaining a set of models inherently carries the risk of assigning nonzero probabilities to some "wrong" models, particularly during the initial phase, or cold start. This can lead to suboptimal performance until sufficient data has been accumulated to better inform the model probabilities.
— *Computational overhead*: The exploration of a broad model space introduces significant computational overhead, especially in high-dimensional model spaces. Techniques like MCMC sampling can be employed to mitigate this drawback by efficiently exploring the model space. However, the computational cost remains higher compared to methods that rely on a single model.

7 Related Work

We organize the related work into two sections. First, we give an overview over the state of the art in uncertainty quantification and mitigation focusing on the most common approaches based on MS. Then, we briefly discuss alternative approaches that are closer to our work. These methods utilize model averaging rather than MS to manage structural uncertainties but are not directly related to self-adaptation.

7.1 Uncertainty Mitigation by MS

Over the last few years, researchers have studied the notion of uncertainty in the context of self-adaptation [2, 8, 18, 39] and proposed mechanisms to explicitly represent and taming uncertainty in different stages of the engineering life-cycle, such as design-time modeling [40], testing [41–43], and runtime verification [6].

In particular, a lot of research has been devoted primarily to the use of stochastic models (such as Markovian models and their different flavors) with parameters that represent uncertainties related to the values of unknown model parameters (e.g., request arrival rates, component failure probabilities, cost or quality of operations). These parameters are updated at runtime by observation and then used by the feedback control loop to reason (e.g., using runtime quantitative verification or,

more scalable, statistical verification for estimating quality values) and make adaptation decisions [44] (e.g., by selecting adaptation configurations that achieve required quality goals). Alternative methods to handle model parameters about uncertainties include: probabilistic methods [45], where parameters are described by probability density functions; fuzzy sets [46], in which uncertain parameters are described with fuzzy boundaries; and interval analysis that is aimed 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 [47, 48]. 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 [47, 49]. 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 [50]) to discard old information [48, 51]. Several approaches have been defined to measure the variation of uncertain input parameters and system output [52]. To quantify the degree of uncertainty as the deviation from initial beliefs, the notion of Bayesian surprise [53] has been introduced to measure the distance from a prior to a posterior and support decision-making when unexpected circumstances possibly require adaptation. This approach quantifies how uncertainties propagate to outputs but does not propose modifications in the analysis/decision-making process itself.

All the aforementioned approaches rely on credible intervals about model parameter estimates, under the overconfident assumption that the best model representing the phenomena of interest has been selected over plausible alternative models. More recent advancements that investigate this type of model uncertainty also exist. For example, the approach introduced by Van Der Donckt et al. [11] applies MS to find 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 goals (threshold goals and optimization goals). Another approach presented by Bencomo and Paucar [12] infers the models at runtime by using a generalization of classical Markov models to deal with situations in which the state of the system is not known in advance.

The selection of a specific model, with its associated distributions and assumptions, may result in overconfident inferences and riskier decision-making for adaptation. In our work, instead, we recognize that the use of models generates new uncertainties being themselves uncertain representations of the reality or system engineer intentions [3]. Using BMA, TUNE-II does not only describe parameter uncertainty through the prior distribution but also model uncertainty obtaining posterior distributions for model parameters and for the model themselves, so adopting a combined model (the average logistic model) with the combined estimation, (hopefully) leading to less risky predictions.

7.2 Uncertainty Mitigation by Model Averaging

While uncertainty quantification in self-adaptive systems and in software engineering, in general, has primarily been on *parametric uncertainties* and on supporting adaptation decisions by selecting one single (best) model of the phenomena of interest (see above), little research has been devoted to *structural uncertainties* that are related to the inability to accurately model real-world phenomena [44]. These structural uncertainties may be revealed in terms of model inadequacy, model bias, model discrepancy, and so on, and are the subject of study in other fields not related to self-adaptive systems and software engineering in general (like statistics, geophysics, hydrogeology, health economics, etc.). In such fields, discrepancy modeling and several model-averaging techniques are commonly adopted to deal with and quantify structural uncertainties [9, 13, 14–16, 54].

In the software engineering field, the BMA method has been used for software reliability modeling. BMA offers the ability to combine the strengths of various individual models, thus enhancing the accuracy of predictions in software reliability assessment. For example, Li et al. [55] propose a BMA-based forecasting method to evaluate software reliability. Through Bayesian inference, this method combines prior information on the unknown quantity with sample information, yielding a posterior distribution from which the unknown value is estimated. This forecasting technique assigns varying weights to different forecasting methods to form a comprehensive combined forecasting model.

8 Conclusion

We present TUNE-II, an approach to mitigate the uncertainty related to the MS process in self-adaptive systems by using multiple model ensembles and BMA. This method improves the predictions made by the analyze component and the plan that adopts many-objective optimizing search to guide the adaptation decisions taking into account multiple (independent) dependability requirements at the same time. Our empirical evaluation shows Model averaging, as implemented by TUNE-II , outperforms traditional MS when dealing with non-steady factors in the semantic space. The adaptation success frequency is significantly better than both random search (large effect size) and MS (ranging from small to large effect size). BMA consistently and significantly reduces execution time for estimates across all requirements and subjects. In relatively low-dimensional model spaces (up to eight variables), exhaustive enumeration of all models generally results in lower costs. However, MCMC sampling offers better scalability and allows us to deal with huge model spaces (up to $\sim 10^{19}$).

We plan to extend our study by considering heterogeneous requirements, such as security, safety, and performance. We also plan to extend our empirical evaluation to other evaluation subjects 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, et al. 2013. Perpetual assurances for self-adaptive systems. In *Software Engineering for Self-Adaptive Systems III. Assurances – International Seminar*, Lecture Notes in Computer Science, Vol. 9640, Springer, 31–63.

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

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

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

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

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

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

[8] Diego Perez-Palacin and Raffaela Mirandola. 2014. Uncertainties in the modeling of self-adaptive systems: A taxonomy and an example of availability evaluation. In *5th ACM/SPEC International Conference on Performance Engineering (ICPE '14)*. ACM, 3–14.

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

[10] M. Jeroen Van Der Donckt, Danny Weyns, M. Usman Iftikhar, and Ritesh Kumar Singh. 2018. Cost-benefit analysis at runtime for self-adaptive systems applied to an internet of things application. In *13th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE '18)*. SciTePress, 478–490.
[11] Jeroen Van Der Donckt, Danny Weyns, Federico Quin, Jonas Van Der Donckt, and Sam Michiels. 2020. Applying deep learning to reduce large adaptation spaces of self-adaptive systems with multiple types of goals. In *IEEE/ACM 15th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '20)*. ACM, 20–30.
[12] Nelly Bencomo and Luis H. Garcia Paucar. 2019. 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)*, 216–226.
[13] Jennifer A. Hoeting, David Madigan, Adrian E. Raftery, and Chris T. Volinsky. 1999. Bayesian model averaging: A tutorial. *Stat. Sci.* 14, 4 (1999), 382–401.
[14] Max Hinne, Quentin Gronau, Don van den Bergh, and Eric-Jan Wagenmakers. 2020. A conceptual introduction to Bayesian model averaging. *Adv. Methods Pract. Psychol. Sci.* 3 (2020), 251524591989865.
[15] B. A. Wintle, Michael Mccarthy, C. Volinsky, and Rod Kavanagh. 2003. The use of Bayesian model averaging to better represent uncertainty in ecological models. *Conserv. Biol.* 17 (2003),1579–1590.
[16] Tiago Fragoso, Wesley Bertoli, and Francisco Louzada. 2018. Bayesian model averaging: A systematic review and conceptual classification. *Int. Stat. Rev.* 86 (2018), 1–28.
[17] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2022. Taming model uncertainty in self-adaptive systems using Bayesian model averaging. *In International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '22)*. Bradley R. Schmerl, Martina Maggio, and Javier Cámara (Eds.), ACM/IEEE, 25–35.
[18] Naeem Esfahani and Sam Malek. 2013. *Uncertainty in Self-Adaptive Software Systems*. Springer, Berlin, 214–238.
[19] Gabriel Moreno, Cody Kinneer, Ashutosh Pandey, and David Garlan. 2019. Dartsim: An exemplar for evaluation and comparison of self-adaptation approaches for smart cyber-physical systems. In *2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*, 181–187. DOI: https://doi.org/10.1109/SEAMS.2019.00031
[20] Danny Weyns. 2020. *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*. John Wiley & Sons.
[21] P. McCullagh and J.A. Nelder. 1983. Generalized Linear Models. *Monographs on Statistics and Applied Probability*. Springer US.
[22] Dani Gamerman and Hedibert F. Lopes. 2006. *Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference*. CRC Press.
[23] James O. Berger. 1985. Statistical Decision Theory and Bayesian Analysis. *Springer Series in Statistics*. Springer.
[24] Hisao Ishibuchi, Noritaka Tsukamoto, and Yusuke Nojima. 2008. Evolutionary many-objective optimization: A short review. *In 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, 2419–2426. DOI: https://doi.org/10.1109/CEC.2008.4631121
[25] Kalyanmoy Deb and Himanshu Jain. 2014. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* 18, 4 (2014), 577–601. DOI: https://doi.org/10.1109/TEVC.2013.2281535
[26] Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* 11, 6 (2007), 712–731. DOI: https://doi.org/10.1109/TEVC.2007.892759
[27] Richard J. Rossi. 2018. *Mathematical Statistics: An Introduction to Likelihood Based Inference*. John Wiley & Sons.
[28] W. K. Hastings. 1970. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* 57, 1 (1970), 97–109.
[29] Christian Blum and Andrea Roli. 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.* 35, 3 (2003), 268–308.
[30] Matteo Camilli, Raffaela Mirandola, and Patrizia Scandurra. 2023. Enforcing resilience in cyber-physical systems via equilibrium verification at runtime. *ACM Trans. Auton. Adapt. Syst.* 18, 3 (2023), 1–32. DOI: https://doi.org/10.1145/3584364
[31] Martin L. Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (1st ed.). John Wiley & Sons, Inc., New York, NY, USA.
[32] Adrian Raftery, Jennifer Hoeting, Chris Volinsky Ian Painter, Ka Yee Yeung, and Hana Sevcikova. 2024. BMA: Bayesian Model Averaging. R Package Version 3.18.19. Retrieved from https://CRAN.R-project.org/package=BMA
[33] Merlise Clyde. 2024. BAS: Bayesian Variable Selection and Model Averaging using Bayesian Adaptive Sampling. R package *version 1.7.5*. Retrieved from https://CRAN.R-project.org/package=BAS
[34] Julian Blank and Kalyanmoy Deb. 2020. Pymoo: Multi-objective optimization in python. *IEEE Access* 8 (2020), 89497–89509. DOI: https://doi.org/10.1109/ACCESS.2020.2990567

[35] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *33rd International Conference on Software Engineering (ICSE '11)*. ACM, 1–10. DOI: https://doi.org/10.1145/1985793.1985795
[36] H. B. Mann and D. R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *Ann. Math. Stat*. 18, 1 (1947), 50–60.
[37] András Vargha and Harold D. Delaney. 2000. A critique and improvement of the "cl" common language effect size statistics of mcgraw and wong. *J. Educ. Behav. Stat*. 25, 2, 101–132. Retrieved from http://www.jstor.org/stable/1165329
[38] Cyril Goutte and Eric Gaussier. 2005. A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In *Advances in Information Retrieval*. David E. Losada and Juan M. Fernández-Luna (Eds.), Springer, Berlin, 345–359.
[39] Andres J. Ramirez, Adam C. Jensen, and Betty H. C. Cheng. 2012. A taxonomy of uncertainty for dynamically adaptive systems. In *7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS '12)*. IEEE Press, 99–108.
[40] Jon Whittle, Pete Sawyer, Nelly Bencomo, Betty H. C. Cheng, and Jean-Michel Bruel. 2009. Relax: Incorporating uncertainty into the specification of self-adaptive systems. In *2009 17th IEEE International Requirements Engineering Conference*, 79–88.
[41] Man Zhang, Shaukat Ali, Tao Yue, Roland Norgren, and Oscar Okariz. 2019. Uncertainty-wise cyber-physical system test modeling. *Softw. Syst. Model.* 18 (2019), 1379–1418.
[42] Matteo Camilli, Angelo Gargantini, and Patrizia Scandurra. 2020. Model-based hypothesis testing of uncertain software systems. *Softw. Test. Verification Reliab*. 30, 2 (2020).
[43] Matteo Camilli, Angelo Gargantini, Patrizia Scandurra, and Catia Trubiani. 2021. Uncertainty-aware exploration in model-based testing. In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, 71–81.
[44] Danny Weyns, Radu Calinescu, Raffaela Mirandola, Kenji Tei, Maribel Acosta, Amel Bennaceur, Nicolas Boltz, Tomas Bures, Javier Camara, Ada Diaconescu, et al. 2023. Towards a research agenda for understanding and managing uncertainty in self-adaptive systems. *SIGSOFT Softw. Eng. Notes* 48. 4 (2023), 20–36. DOI: https://doi.org/10.1145/3617946.3617951.
[45] P. D. Arendt, D. W. Apley, and W. Chen. 2012. Quantification of model uncertainty: Calibration, model discrepancy, and identifiability. *J. Mech. Des*. 134, 10 (2012), 100908.
[46] Pooyan Jamshidi, Amir Sharifloo, Claus Pahl, Hamid Arabnejad, Andreas Metzger, and Giovani Estrada. 2016. Fuzzy self-learning controllers for elasticity management in dynamic cloud architectures. In *2016 12th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA)*, 70–79.
[47] Antonio Filieri, Carlo Ghezzi, and Giordano Tamburrelli. 2012. A formal approach to adaptive software: Continuous assurance of non-functional requirements. *Form. Asp. Comput*. 24, 2 (2012), 163–186.
[48] Radu Calinescu, Yasmin Rafiq, Kenneth Johnson, and Mehmet Emin Bakir. 2014. Adaptive model learning for continual verification of non-functional properties. In *5th ACM/SPEC International Conference on Performance Engineering (ICPE '14)*. ACM, 87–98.
[49] Matteo Camilli, Angelo Gargantini, Patrizia Scandurra, and Carlo Bellettini. 2017. Towards inverse uncertainty quantification in software development (short paper). In *Software Engineering and Formal Methods*. Alessandro Cimatti and Marjan Sirjani (Eds.), Springer International Publishing, Cham, 375–381.
[50] Karl J. Astrom and Bjorn Wittenmark. 1990. *Computer-controlled Systems: Theory and Design* (2nd ed.). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.
[51] Antonio Filieri, Lars Grunske, and Alberto Leva. 2015. Lightweight adaptive filtering for efficient learning and updating of probabilistic models. In *37th International Conference on Software Engineering Volume 1 (ICSE '15)*. IEEE Press, 200–211.
[52] S. H. Lee and Wei Chen. 2009. A comparative study of uncertainty propagation methods for black-box-type problems. *Struct. Optim*. 37, 3 (2009), 239–253
[53] Nelly Bencomo and Amel Belaggoun. 2014. 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*. ACM, 460–463.
[54] Abhishek Singh, Srikanta Mishra, and Greg Ruskauff. 2010. Model averaging techniques for quantifying conceptual model uncertainty. *Groundwater* 48, 5 (2010), 701–715. DOI: https://doi.org/10.1111/j.1745-6584.2009.00642.x
[55] Zhaojun Steven Li, Shun Jia, and Qiumin Yu. 2023. A Bayesian model averaging method for software reliability modeling and assessment. *Qual. Reliab. Eng. Int*. 39, 3 (2023), 958–970. DOI: https://doi.org/10.1002/qre.3273

Received 12 June 2024; revised 11 December 2024; accepted 6 February 2025