Stochastic Variational Smoothed Model(307,76),(671,119)(307,120),(671,155)Checking
Luca Bortolussi$^1$, Francesca Cairoli$^1$, Ginevra Carbone$^1$, and Paolo Pulcini$^1$
DMG, University of Trieste, Italy

Abstract. Model-checking for parametric stochastic models can be expressed as checking the satisfaction probability of a certain property as a function of the parameters of the model. Smoothed model checking (smMC) [4] leverages Gaussian Processes (GP) to infer the satisfaction function over the entire parameter space from a limited set of observations obtained via simulation. This approach provides accurate reconstructions with statistically sound quantification of the uncertainty. However, it inherits the scalability issues of GP. In this paper, we exploit recent advances in probabilistic machine learning to push this limitation forward, making Bayesian inference of smMC scalable to larger datasets, enabling its application to larger models in terms of the dimension of the parameter set. We propose Stochastic Variational Smoothed Model Checking (SV-smMC), a solution that exploits stochastic variational inference (SVI) to approximate the posterior distribution of the smMC problem. The strength and flexibility of SVI make SV-smMC applicable to two alternative probabilistic models: Gaussian Processes (GP) and Bayesian Neural Networks (BNN). Moreover, SVI makes inference easily parallelizable and it enables GPU acceleration. In this paper, we compare the performances of smMC [4] against those of SV-smMC by looking at the scalability, the computational efficiency and at the accuracy of the reconstructed satisfaction function.

1 Introduction
Parametric verification of logical properties aims at providing meaningful insights into the behaviour of a system, checking whether its evolution satisfies or not a certain requirement, expressed as a temporal logic formula, varying some parameters of the system’s model. Stochastic systems, however, require the use of probabilistic model checking techniques as the satisfaction of a property is itself a stochastic quantity. In this direction, statistical model checking (SMC) uses statistical tools to estimate the satisfaction probability of logical properties from trajectories sampled from the stochastic model and it enriches these estimates with probabilistic bounds of the estimation errors. For instance, if the number of sampled trajectories is sufficiently large, the satisfaction probability, estimated as the average of satisfaction on individual runs, will converge to the true probability. However, if the parameters of the stochastic model vary, the dynamics of the system will also vary. Therefore, SMC has to be performed from
scratch for each set of parameter values, making SMC computationally unfeasible for parametric stochastic models. In this regard, the satisfaction probability of a signal temporal logic (STL) requirement over a parametric stochastic model has been proved to be a smooth function of the parameters of the model [4]. This result enables the use of machine learning techniques to infer an approximation of this function from a limited pool of observations. Observations, computed via SMC for a limited number of parameter values, are noisy and computationally demanding to obtain. This calls for Bayesian approaches, where predictions provide a probabilistic quantification of the predictive uncertainty. In this regard, in [4] the authors present smoothed model checking (smMC), a fully Bayesian solution based on Gaussian Processes (GP). Since the observation process is non-Gaussian, outputs are in fact realizations of a Bernoulli distribution, exact GP inference is unfeasible. The authors thus resort to the Expectation Propagation (EP) algorithm to approximate the posterior inference. Unfortunately, the cost of EP is cubic in the number of observations used to train the GP, making smMC applicable only to models with a low dimensional parameter space that require a limited number of training observations. In [15] this scalability problem is tackled by using a sparse variational approach to GP inference. Sparsification reduces the computational complexity, by performing inference on a limited set of observations, called inducing points. On the other hand, variational inference is used to perform approximate inference of a GP classification (GPC) problem. The variational approach for GPC used in [15] builds on [18], in which the objective function used for optimization does not depend explicitly on the inducing variables, forcing them to be fixed a priori. Scalability is thus improved, but sparsification strongly reduces the reconstruction accuracy. Moreover, in [15] the smMC problem is framed as a GPC problem, meaning that observations come from the satisfaction of a single trajectory. If for a certain parameter we simulate $M$ trajectories, this would result in $M$ different observations. On the contrary, in [4] the observation process was modeled by a Binomial so that the satisfaction of the $M$ simulations is condensed into a single observation. This has a strong effect on the dimension of the training set which is of paramount importance for sake of scalability. Finding an effective solution that makes smMC scale to large datasets remains an open issue. In this paper, we propose a novel approach for scalable smMC, called Stochastic Variational Smoothed Model Checking (SV-smMC) that leverages Stochastic Variational Inference (SVI), a popular solution for making Bayesian inference scalable to large datasets. SVI is extremely flexible, the general idea can be applied either to Gaussian Processes but also to Bayesian Neural Networks. The main advantage of SVI, compared to the VI used for example in [15], is that the gradient-based optimization can be stochastically approximated using mini-batches of data, resulting in the well-known stochastic gradient descent (SGD) algorithms. The use of mini-batches makes inference easily parallelizable, enabling also GPU acceleration. As a result, Bayesian inference can face even very large datasets.

This paper is structured as follows. We start by presenting the background theory in Section 2. In Section 3 the theoretical details of SV-smMC are presented
both for the GP version (Section 3.1) and for the BNN version (Section 3.2). Finally, in Section 4 we compare the performances of SV-smMC against those of smMC on three stochastic models with increasing parametric complexity.

2 Background

2.1 Population Continuous Time Markov Chain

A population model of interacting agents can be modeled as a stochastic system evolving continuously in time over a finite or countable state space $X$. If the system is Markovian, meaning if the memory-less property holds, the population model can be modeled as a population Continuous Time Markov Chain (CTMC) $M$. Given a population with $n$ different species $\{S_1, \ldots, S_n\}$ and $m$ possible reactions $\{R_1, \ldots, R_m\}$, the respective CTMC can be described by:

- a state vector, $X(t) = (X_1(t), \ldots, X_n(t))$, taking values in $X \subseteq \mathbb{N}^n$ and counting the number of agents in each species at time $t$;
- a finite set of reactions $R = (R_1, \ldots, R_m)$ describing how the state of the population changes in time, i.e., the system dynamics. A general reaction $R_i$ is identified by the tuple $(\tau_i, \nu_i)$, where:
  - $\tau_i : S \times \Theta_i \rightarrow \mathbb{R}_{\geq 0}$ is the rate function of reaction $R_i$ that associates to each reaction the rate of an exponential distribution, depending on the global state of the model and on a parameter $\theta_i$, and
  - $\nu_i$ is the update vector, giving the net change of agents due to the reaction, so that the firing of reaction $R_i$ results in a transition of the system from state $X(t)$ to state $X(t) + \nu_i$.

Reaction rules are easily visualised in the chemical reaction style, as

$$R_i : \sum_{j \in \{1, \ldots, n\}} \alpha_{ij} S_j \xrightarrow{\tau_i(X, \theta_i)} \sum_{j \in \{1, \ldots, n\}} \beta_{ij} S_j.$$  

The stoichiometric coefficients $\alpha_i = [\alpha_{i1}, \ldots, \alpha_{in}]$, $\beta_i = [\beta_{i1}, \ldots, \beta_{in}]$ can be arranged so that they form the update vector $\nu_i = \beta_i - \alpha_i$ for reaction $R_i$.

The parameters $\theta = (\theta_1, \ldots, \theta_m)$ have a crucial effect on the dynamics of the system: changes in $\theta$ can lead to qualitatively different dynamics. We stress such crucial dependency by using the notation $M_\theta$. The trajectories of such a CTMC can be seen as samples of random variables $X(t)$ indexed by time $t$ over a state space $X$. A parametric CTMC (pCTMC) is a family $M_\theta$ of CTMCs where the parameters vary in a domain $\Theta$.

2.2 Signal Temporal Logic

Properties of CTMC trajectories are expressed via Signal Temporal Logic (STL) [12]. STL allows the specification of properties of dense-time, real-valued signals, and
the automatic generation of monitors for testing properties on individual trajectories. The rationale of STL is to transform real-valued signals into Boolean ones, using formulae build on the following **STL syntax**:

$$\varphi := \text{true} \mid \mu \mid \neg \varphi \mid \varphi \land \varphi \mid \varphi \text{ U } I,$$

where $I \subseteq T$ is a temporal interval, either bounded, $I = [a, b]$, or unbounded, $I = [a, +\infty)$, for any $0 \leq a < b$. Atomic propositions $\mu$ are (non-linear) inequalities on population variables. From this essential syntax it is easy to define other operators, used to abbreviate the syntax in a STL formula:

- $false := \neg \text{true}$,
- $\varphi \lor \psi := \neg(\neg \varphi \land \neg \psi)$,
- $F_I := \text{true} \text{ U } I \varphi$ and $G_I := \neg F_I \neg \varphi$.

Monitoring the satisfaction of a formula is done recursively leveraging the tree structure of the STL formula. See [12] for the details on the STL Boolean semantics and on Boolean STL monitors.

### 2.3 Smoothed Model Checking

Smoothed Model Checking (smMC), presented in [4], uses Gaussian processes to infer the satisfaction function of pCTMC from a set of observations obtained via statistical model checking.

**Statistical Model Checking (SMC).** Given a CTMC $M_\theta$ with fixed parameters $\theta$, time-bound CTMC trajectories are sampled by standard simulation algorithms, like SSA [8], and monitoring algorithms for STL [12] are used to assess if the formula $\varphi$ is satisfied for each sampled trajectory. This process produces samples from a Bernoulli random variable equal to 1 if and only if $\varphi$ is true. SMC [20,21] then uses standard statistical tools, either frequentist [20] or Bayesian [21], to estimate from these samples the satisfaction probability $Pr(\varphi|M_\theta)$ or to test if $Pr(\varphi|M_\theta) > q$ with a given confidence level.

**Satisfaction Function for pCTMCs.** Building on [4], our interest is to quantify how the satisfaction of STL formulae depends on the unknown parameters of the pCTMC. We define the **satisfaction function** $f_\varphi : \Theta \rightarrow [0, 1]$ associated to $\varphi$ as

$$f_\varphi(\theta) = Pr(\varphi = \text{true}|M_\theta).$$

In order to have an accurate estimation, the satisfaction function $f_\varphi$ over the entire parameter space $\Theta$ by means of SMC would require a prohibitively large number of evaluations. In [4], Theorem 1, it has been shown that $f_\varphi(\theta)$ is a smooth function of the model parameters and thus machine learning techniques can be used to infer this function from a limited set of observations.

**Problem statement.** Given a pCTMC $M_\theta$ and an STL formula $\varphi$, the goal of smMC is to find a statistical estimate of the satisfaction function of (2) from a set of noisy observations of $f_\varphi$ obtained at few parameter values $\theta_1, \theta_2, \ldots$. The task is then to construct a statistical model that, for any value $\theta^* \in \Theta$, computes efficiently an estimate of $f_\varphi(\theta^*)$ together with a credible interval for such a
More precisely, given an input point \( \theta \), our observations are obtained by evaluating a property \( \varphi \) on a single trajectory sampled from the stochastic model \( M_\theta \) via SSA. Thus, given a set of \( N_t \) parameter values, \( \Theta_t = \{ \theta_1, \ldots, \theta_{N_t} \} \), we simulate, for each parameter \( \theta_i \), \( M_t \) trajectories, obtaining \( M_t \) Boolean values \( \ell^j_i \in \{0,1\} \) for \( j = 1, \ldots, M_t \). We condense these Boolean values in a vector \( L_i = [\ell^1_i, \ldots, \ell^{M_t}_i] \). The noisy observations can be summarized as a training set \( D_t = \{(\theta_i, L_i) \mid i = 1, \ldots, N_t \} \).

SmMC uses GP to solve the above inference problem. Since the observation process is non-Gaussian, exact inference is unfeasible and thus Expectation Propagation is used to approximate the posterior distribution. This solution scales as \( O(N_t^3) \), it is thus unfeasible for large datasets. Notice that, if the observation process is modeled by a Bernoulli, as in [15], meaning if Boolean values \( \ell^j_i \) are considered as individual observations instead of condensing them in a vector \( L_i \), inference scales as \( O((N_t M_t)^3) \). In order to mitigate such scalability issue, in [15] the authors use variational inference together with sparsification techniques to make inference feasible for slightly larger datasets. Nonetheless, stochastic variational inference is not applicable and sparsification strongly reduces the reconstruction accuracy, thus scalability remains an open issue.

3 Stochastic Variational Smoothed Model Checking

The goal of Stochastic Variational Smoothed Model Checking (SV-smMC) is to efficiently infer an accurate probabilistic estimate of the unknown satisfaction function \( f_\varphi : \Theta \to [0,1] \). In order to do so, we define a function \( f : \Theta \to [0,1] \) that should behave as similarly as possible to \( f_\varphi \). The main ingredients of a Bayesian approach to the problem stated above are the following:

1. Choose a prior distribution, \( p(f) \), over a suitable function space encapsulating the beliefs about function \( f \) prior to any observations being taken.
2. Determine the functional form of the observation process by defining a suitable likelihood function that effectively models how the observations depend on the uncertain parameter \( \theta \). Our observation process can be modeled by a binomial over \( M_t \) trials with parameter \( f_\varphi(\theta_i) \). Given the nature of our training set, defined in (3), we define the probabilistic likelihood as

\[
p(D_t|f) = \prod_{i=1}^{N_t} \text{Binomial}(L_i \mid M_t, f(\theta_i)).
\]

3. Leverage Bayes’ theorem to define the posterior distribution over functions given the observations

\[
p(f|D_t) = \frac{p(D_t|f)p(f)}{p(D_t)}.
\]

Computing \( p(D_t) = \int p(D_t|f)p(f)df \) is almost always computationally intractable for non-conjugate prior-likelihood distributions. Therefore, we need algorithms to accurately approximate such posterior distribution.
4. Evaluate such posterior at points $\theta^*$, resulting in a predictive distribution $p(f^*|\theta^*,D_t)$, whose statistics are used to obtain the desired estimate of the satisfaction probability together with the respective credible interval.

SV-smMC leverages stochastic variational inference to efficiently compute the approximate posterior distribution $p(f|D_t)$ so that smMC inference scales well to large dataset $D_t$. In particular, SV-smMC proposes two alternative probabilistic models to define distributions over function $f$. The first one is based on Gaussian Processes (GP), whereas the second one is based on Bayesian Neural Networks (BNN).

3.1 Gaussian Processes

Gaussian Processes (GP) define a distribution over real-valued functions of the form $g : \Theta \rightarrow \mathbb{R}$ and such distribution is uniquely identified by its mean and covariance functions, respectively denoted by $\mu(\theta) = \mathbb{E}[g(\theta)]$ and $k_\gamma(\theta,\theta')$. The GP can thus be denoted as $GP(\mu(\theta),k_\gamma(\theta,\theta'))$. This means that the function value at any point $\theta$, $g(\theta)$, is a Gaussian random variable with mean $\mu(\theta)$ and variance $k_\gamma(\theta,\theta)$. Let $g_t$, $\mu_t$ and $K_{N_t}$ denote respectively the latent, the mean and the covariance functions evaluated over the training inputs $\Theta_t$. The outputs of the latent functions $g : \Theta \rightarrow \mathbb{R}$ are mapped into the $[0,1]$ interval by means of a so-called link function $\Phi$, typically the logit or the probit function, so that $f = g \circ \Phi$.

The GP prior over latent functions $g$ is defined as $p(g|\Theta_t) = \mathcal{N}(g | \mu_t, K_{N_t})$. For simplicity, we assume $\mu_t = 0$. In order to do inference over a test input $\theta^*$, with latent variable $g^*$, we have to compute

$$p(f^*|\theta^*,\Theta_t,L_t) = \int \Phi(g^*) p(g^*|\theta^*,\Theta_t,L_t) dg^*, \quad (4)$$

where $L_t$, denotes the set of Boolean tuples corresponding to points in $\Theta_t$. To compute equation (4), we have to marginalize the posterior over the latent Gaussian variables:

$$p(g^*|\theta^*,\Theta_t,L_t) = \int p(g^*|\theta^*,\Theta_t,g_t)p(g_t|\Theta_t,L_t) dg_t, \quad (5)$$

where the posterior $p(g_t|\Theta_t,L_t)$ is not available in closed form since it is the convolution of a Gaussian and a binomial distribution. Hence, we have to rely on approximations in order to compute the posterior $p(g_t|D_t)$ and thus solve (5). Once we obtain a tractable approximation of $p(g^*|\theta^*,\Theta_t,L_t)$, we can easily compute an empirical approximation of $p(f^*|\theta^*,\Theta_t,L_t)$.

**Stochastic Variational Inference.** Here we provide an understanding of how SVI works over GP with non-Gaussian likelihoods. For a more detailed description, we refer the interested reader to [9,10]. Variational approaches to GP start with sparsification. It defines a set of $m \ll N_t$ inducing points $Z = \{z_1, \ldots, z_m\}$
that live in the same space of $\Theta_t$ and, from them, we define a set of inducing variables $u_t = [g(z_i) | z_i \in Z]$. The covariance matrix over inducing points, $K_{mm}$, is less expensive to invert and thus it act as a low-rank approximation of $K_{NtNt}$. Leveraging Jensen inequality and standard variational relations over the log marginal likelihood, we obtain the following bound:

$$\log p(L_t) \geq \mathbb{E}_{q(g_t)}[\log p(L_t | g_t)] - KL[q(u_t) || p(u_t)],$$

where $q(g_t) = \int p(g_t | u_t) q(u_t) du_t$. We choose $q(u_t) := \mathcal{N}(u_t | m, \tilde{S})$, where $\tilde{S}$ is defined using a lower triangular form, $\tilde{S} := SS^T$, in order to maintain positive-definiteness. Then

$$q(g_t) = \mathcal{N}(g_t | Am, K_{NtNt} + A(\tilde{S} - K_{mm}) A^T),$$

where $A = K_{NtNt} K_{mm}^{-1}$. Since our likelihood factors as $p(L_t | g_t) = \prod_{i=1}^{N_t} p(L_i | g_i)$, the lower bound becomes:

$$\log p(L_t) \geq \sum_{i=1}^{N_t} \mathbb{E}_{q(g_i)}[\log p(L_i | g_i)] - KL[q(u_t) || p(u_t)] := \mathcal{L}_{GP}(m, S, Z, \gamma).$$

The SVI algorithm then consists of maximizing $\mathcal{L}_{GP}$ with respect to its parameters using gradient-based stochastic optimization. The gradient of $\mathcal{L}_{GP}$ contains the partial derivatives w.r.t. the SVI hyper-parameters and w.r.t. the hyper-parameter $\gamma$ of the covariance function. Computing the KL divergence in (8) requires only $O(m^3)$ computations. Most of the work will thus be in computing the expected likelihood terms. Given the ease of parallelizing the simple sum over $N_t$, we can optimize $\mathcal{L}_{GP}$ in a distributed or in a stochastic fashion by selecting mini-batches of the data at random.

**Predictive distribution.** The posterior of (5) is now approximated as $p(g_* | L_t) \approx \int p(g_* | u_t) q(u_t) du_t := q(g_*).$ The integral above can be treated similarly to (7), computing its mean and variance takes $O(m^2)$. From the mean and the variance of $q(g_*), we obtain the respective credible interval and use the link function $\Phi$ to map it to a subset of the interval $[0, 1]$, so that we have mean and credible interval of the posterior predictive distribution $p(f_* | \theta_*, D_t)$.

### 3.2 Bayesian Neural Networks

The core idea of Bayesian neural networks (BNNs) is to place a probability distribution over the weights $w$ of a neural network $f_w : \Theta \to [0, 1]$, transforming the latter into a probabilistic model.

The Bayesian learning process starts by defining a prior distribution for $w$ that expresses our initial belief about the weights values. A common choice is to choose a zero-mean Gaussian prior. As we observe data $D_t$, we update this prior to a posterior distribution using Bayes’ rule:

$$p(w \mid D_t) = \frac{p(D_t | w)p(w)}{p(D_t)}.$$
Because of the non-linearity introduced by the neural network function $f_w(\theta)$ and since the likelihood is binomial, the posterior $p(w|D_t)$ is non-Gaussian and it cannot be computed analytically. In order to predict the satisfaction function over an unobserved input $\theta_*$, we marginalize the predictions with respect to the posterior distribution of the parameters, obtaining

$$p(f_*|\theta_*, D_t) = \int f_w(\theta_*) p(w|D_t) dw.$$  

(10)

The latter is called posterior predictive distribution and it can be used to retrieve information about the uncertainty of a specific prediction $f_*$. Unfortunately, the integration is analytically intractable due to the non-linearity of the neural network function [3,11].

**Stochastic Variational Inference.** Given the unknown posterior distribution $p(w|D_t)$, the rationale of SVI is to choose a family of parametric distributions $q_\psi(w)$, over the same latent variables and minimize the KL divergence between these two distributions, $KL[q_\psi(w)||p(w|D_t)]$. The goal is to find the members of the family that are closest to the true posterior $p(w|D_t)$. Since the posterior distribution is not known, a different objective function, called Evidence Lower Bound (ELBO) [6], is introduced. In practice, minimizing the KL divergence is equivalent to maximising the Evidence Lower Bound (ELBO), defined as

$$L_{BNN}(\psi) = E_{q_\psi(w|D_t)} [\log(D_t|w)] - KL [q_\psi(w)||p(w)].$$  

(11)

The first term is the expected log-likelihood of our data with respect to values of $f_w$ sampled from $q_\psi(w|D_t)$, whereas the second term is the KL divergence between the proposal distribution and the prior. The family of distribution $q_\psi$ should be a distribution easy to sample from and such that the KL divergence is easy to compute. A common choice for $q_\psi$ is the Gaussian distribution (where $\psi$ denotes its mean and variance). The KL distribution among two Gaussian distributions has an exact analytical form. Thus, the ELBO of Eq. (11) can be computed and it can be used as the objective function of a maximization problem over $\psi$.

**Empirical approximation of the predictive distribution.** The predictive distribution (10) is a non-linear combination of Gaussian distributions, and thus it is not Gaussian. However, samples can be easily extracted from $q(w; \psi)$, which allows us to obtain an empirical approximation of the predictive distribution. Let $[w_1, \ldots, w_C]$ denote a vector of $C$ realizations of the random variable $w \sim q_\psi(w)$. Each realization $w_i$ induces a deterministic function $f_{w_i}$ that can be evaluated at $\theta_*$, the unobserved input, providing an empirical approximation of $p(f_*|\theta_*, D_t)$. By the strong law of large numbers, the empirical approximation converges to the true distribution as $C \to \infty$ [19]. The sample size $C$ can be chosen, for instance, to ensure a given width of the confidence interval for a statistic of interest [16] or to bound the probability that the empirical distribution differs from the true one by at most some given constant [13].
4 Experiments

4.1 Case studies

In the following, we briefly introduce the case studies that we investigate for understanding the scalability and the accuracy of SV-smMC.

– **Network Epidemics (SIR):** a simple model describing the spread of an epidemics in a population of fixed size. Susceptible nodes, $X_S$, can be infected after contact with an infected node. Infected nodes, $X_I$, can recover after some time, and become immune, $X_R$, from the infection. Here we consider the case of permanent immunisation. The dynamics depends on two parameters $\beta$ and $\gamma$. The chosen STL property describes the termination of epidemics in a time between 100 and 120 time units from the epidemic onset:

$$\varphi = (X_I > 0) \cup_{[100,120]} (X_I = 0).$$

– **Prokaryotic Gene Expression (PrGeEx):** this more complex model captures LacZ, $X_{LacZ}$, protein synthesis in E. coli. The dynamics is governed by 11 parameters $k_1, k_2, \ldots, k_{11}$. We chose a STL property for detecting bursts of gene expression:

$$\varphi = F_{[1600,2100]}(\Delta X_{LacZ} > 0 \land G_{[10,200]}(\Delta X_{LacZ} \leq 0)),$$

where $\Delta X_{LacZ}(t) = X_{LacZ}(t) - X_{LacZ}(t - 1)$. This formula monitors rapid increases in LacZ counts, followed by long periods of lack of protein production.

– **Three-layer Phosphorelay (PhosRelay):** network of three proteins $L_1, L_2, L_3$ involved in a cascade of phosphorylation reactions (changing the state of the protein), in which protein $L_j$, in its phosphorylated form $L_{jp}$, acts as a catalyst of phosphorylation of protein $L(j + 1)$. There is a ligand $B$ triggering the first phosphorylation in the chain. The dynamics depends on 6 parameters $k_p, k_1, k_2, k_3, k_4, k_d$. The chosen STL property models a switch in the most expressed protein between $L_{1p}$ and $L_{3p}$ after time 300

$$\varphi = G_{[0,300]}(L_{1p} - L_{3p} \geq 0) \land F_{[300,600]}(L_{3p} - L_{1p} \geq 0).$$

Details about the dynamics, i.e. about the reactions, the selected initial states and the chosen parametric ranges, are provided in Appendix A.

4.2 Experimental Details

**Dataset generation.** The training set $D_t$ is built as per Eq. (3). The test set, used to validate our results, can be summarized as $D_v = \{(\theta_j, (t_{ij}^1, \ldots, t_{ij}^{M_v})) | j = 1, \ldots, N_v\}$, where $M_v$ is chosen very large, $M_v \gg M_t$, so that we have a good estimate of the true satisfaction probability over each test input. Input data, i.e. the parameter values, are scaled to the interval $[-1, 1]$ to enhance the performances of the inferred models and to avoid sensitivity to different scales.
in the parameter space. In order to deal with scenarios with gradually increasing parametric complexity, we choose, for each case study, different subsets of varying parameters and train a separate model on each of these choices. In other words, we fix some of the parameters and let only the remaining parameters vary. The parameter space considered is thus a subspace of the original one. This allows us to analyze the scalability of the inferred smMC models across the different case studies and with respect to parameter spaces with different dimensions. In particular, in SIR we consider the following configurations: (a) fix $\gamma$ and let $\beta$ vary, (b) fix $\beta$ and let $\gamma$ vary, (c) vary both $\beta$ and $\gamma$; in PrGeEx we consider the following configurations: (d) $k_2$ is the only parameter allowed to vary, (e) we let $k_2$ and $k_7$ vary; in PhosRelay we consider the following configurations: (f) only $k_1$ varies, (g) only $k_p, k_d$ vary, (h) only $k_1, k_2, k_3$ vary, (i) only $k_1, k_2, k_3, k_4$ vary, (l) all six parameters are allowed to vary. The results over the simple one-dimensional scenarios are easy to visualise, thus they represent a good baseline to evaluate the smMC performances. Table 1 contains the dimensions chosen for every training and test dataset.

| Configuration | Dataset Size | Training time |
|---------------|--------------|---------------|
|               | $N_t$ | $M_t$ | $N_v$ | $M_v$ | EP-GP | SVI-GP | SVI-BNN |
| (a) SIR $\beta$ | 500 | 50 | 1 | 1k | 0.05 | 0.11 | 0.50 |
| (b) SIR $\gamma$ | 700 | 50 | 1 | 1k | 0.26 | 0.44 | 0.24 |
| (c) SIR $\beta, \gamma$ | 2500 | 50 | 400 | 1k | 12.22 | 08.12 | 19.15 |
| (d) PrGeEx $k_2$ | 500 | 50 | 400 | 1k | 0.13 | 0.13 | 0.51 |
| (e) PrGeEx $k_2, k_7$ | 2500 | 50 | 400 | 1k | 22.11 | 08.31 | 19.10 |
| (f) PhosRelay $k_1$ | 500 | 50 | 1k | 1k | 0.00 | 0.20 | 0.51 |
| (g) PhosRelay $k_p, k_d$ | 2500 | 50 | 400 | 1k | 06.57 | 08.70 | 19.39 |
| (h) PhosRelay $k_1, k_2, k_3$ | 8k | 20 | 1k | 1k | 36.13 | 26.38 | 66.30 |
| (i) PhosRelay $k_1, \ldots, k_4$ | 10k | 20 | 4096 | 1k | 45.59 | 33.45 | 81.16 |
| (l) PhosRelay $k_p, k_1, \ldots, k_4, k_d$ | 1000k | 20 | 4096 | 1k | – | 286.33 | 92.30 |

Table 1: Size of training ($D_t$) and test ($D_v$) datasets and training time (min) for EP-GP, SVI-GP and SVI-BNN. $N_t$ and $N_v$ denote the number of parameter values, while $M_t$ and $M_v$ denote the number of Bernoulli observations for each parameter value. Underlined values correspond to GPU computations. EP-GP training time for the last configuration is not reported due to memory timeout.

**Experimental settings.** The CTMC dynamics is simulated via StochPy SSA simulator. The Boolean semantics of pcheck library\(^1\) is used to check the satisfaction of a certain formula for a specific trajectory. GPyTorch [7] library is used to train the SVI-GP models and Pyro [2] library is used to train the SVI-BNN models, both built upon PyTorch [14] library. Instead, EP-GP is imple-

\(^1\) https://github.com/simonesilvetti/pcheck
mented in NumPy. The experiments were conducted on a shared virtual machine with 32 cores, Intel(R) Xeon(R) Gold 6140 CPU @ 2.30GHz processors and 264GB of RAM and a NVidia V100 GPU with 16Gb of RAM. The results are fully reproducible. Code is available at: https://github.com/ginevracoal/smoothed-model-checking.

Training and evaluation. We apply Stochastic Variational Inference on both Gaussian Processes (SVI-GPs) and Bayesian Neural Networks (SVI-BNNs) and compare them to the baseline smMC approach, where Gaussian Processes were inferred using Expectation Propagation (EP-GPs). All models (EP-GP, SVI-GP and SVI-BNN) are Bayesian and trained over the training set $D_t$. Once the training phase is over, for each pair $(\theta_j, (\ell_{j1}, \ldots, \ell_{jMv})) \in D_v$ in the test set, we obtain a probabilistic estimate of the satisfaction probability $f(\theta_j)$ (defined in Section 3). We compare such distribution to the satisfaction probability $f_\varphi(\theta_j)$, estimated as the mean over the Bernoulli trials $(\ell_{j1}, \ldots, \ell_{jMv})$. For simplicity, we call the latter SMC satisfaction probability. We stress that SMC estimates provably converge to the true satisfaction probabilities, meaning that the width of confidence intervals converges to zero in the limit of infinite samples, while Bayesian inference quantifies the predictive uncertainty. Consequently, regardless of the number of samples, SMC and Bayesian estimates have different statistical meanings.

Evaluation metrics. To define meaningful measures of performance, let’s clarify the notation. For each point in the test set, $j \in \{1, \ldots, N_v\}$, let $f_\varphi(\theta_j)$ and $\sigma_j$ denote respectively the average and the standard deviation over the $M_v$ Bernoulli trials $(\ell_{j1}, \ldots, \ell_{jMv})$. The inferred models, on the other hand, provide a posterior predictive distribution $p(f_j|\theta_j, D_t)$, let $q_{\alpha}^j$ denote the $\alpha$-th quantile of such distribution. The metrics used to quantify the overall performances of the models over each case study and each configuration are the following:

(i) the mean squared error (MSE) between SMC and the expected satisfaction probabilities, i.e. the average of the squared residuals

$$\text{MSE} = \frac{1}{N_v} \sum_{j=1}^{N_v} (f_\varphi(\theta_j) - \mathbb{E}_{p(f|\theta_j, D_t)}[f(\theta_j)])^2.$$ 

This measure evaluates the quality of reconstruction provided by the mean of the posterior predictive distribution;

(ii) the accuracy over the test set, i.e. the fraction of non empty intersections between SMC confidence intervals and estimated $(1 - \alpha)$ credible intervals:

$$\text{Acc} = \frac{1}{N_v} \cdot \left| \left\{ j \in \{1, \ldots, N_v\} : \left[ -\frac{z\sigma_j}{\sqrt{M_v}}, \frac{z\sigma_j}{\sqrt{M_v}} \right] \cap \left[ q_{\alpha/2}^j, q_{1-\alpha/2}^j \right] \neq \emptyset \right\} \right|.$$ 

\[2\] Our implementation builds on https://github.com/simonesilvetti/pyCheck/blob/master/smoothed/smoothedMC.py
In particular, we set $z = 1.96$ and $\alpha = 0.05$ in order to have the 95% confidence intervals and the 95% credible intervals respectively;

(iii) the average width of the estimated credible intervals

$$\text{Unc} = \frac{1}{N_w} \sum_{j=1}^{N_v} (q_{1-\alpha/2}^j - q_{\alpha/2}^j),$$

which quantifies how informative the predictive uncertainty is and allows us to detect overconservative predictors.

A good predictor should be balanced in terms of low MSE, high test accuracy, i.e. high values for Acc, and narrow credible intervals, i.e. low values for Unc. To account for numerical errors we set the estimated lower bounds $q_{0.025}^j$ to zero whenever $q_{0.025}^j < 10^{-6}$.

**Implementation details.** SVI-GP models are trained for $1k$ epochs with mini-batches of size 100 and a learning rate of $0.01$. The GP prior is computed on a maximum of $1k$ inducing points selected from the training set. SVI-BNNs have a fully connected architecture with 3 layers and Leaky ReLU nonlinear activations. They are trained for $5k$ epochs with mini-batches of size 100 and a learning rate of 0.001. The prior distribution over the weights of the BNNs is a Gaussian $\mathcal{N}(0, 1/m)$ on each weight, where $m$ is the layer width, i.e. the number of neurons per layer. In the most challenging setting, PhosRelay with 6 parameters ($l$), we set the number of epochs to 100 and the batch size to $5k$ for both SVI-GPs and SVI-BNNs. To evaluate SVI-BNNs we take $1k$ samples from the posterior distribution evaluated over test inputs.

### 4.3 Experimental Results

**Computational costs.** The cost of EP-GP inference is dominated by the cost of matrix inversion, which is cubic in the number of points in the training set. The cost of SVI-GP inference is cubic in the number of inducing points, which is chosen to be sufficiently small, and linear in the number of training instances. The cost of SVI-BNN is linear in the number of training points but it also depends on the architectural complexity of the chosen neural network. Variational models are trained by means of SGD, which is a stochastic inference approach. Thus, at least on simple configurations, it is likely to take longer than EP in reaching convergence. The computational advantage becomes significant as the complexity of the case study increases, i.e., when the training set is sufficiently large, moving towards the memory-bound problem of EP, which is typically unfeasible on configurations with a parameter space with dimensions larger than four (like our last testing configuration). Moreover, SVI-GP has an important collateral advantage in that of optimizing the kernel hyperparameters on the fly during the training phase, whereas in EP-GP the hyperparameters search is performed beforehand and it is rather expensive. Table 1 reports training times for EP-GP, SVI-GP and SVI-BNN. We can observe how SVI-GP, trained leveraging GPU
acceleration, is, in general, the most efficient model as its convergence times are comparable to EP’s on simple configurations and they outperform EP on more complex ones. The training times for SVI-BNN are typically larger than EP-GP and SVI-GP for small parameter spaces, becoming more efficient for very large datasets. Moreover, we noticed how it seems more convenient to train SVI-BNN on the CPU alone for small datasets, since using the GPU introduces a significant overhead due to memory transfer.

Evaluation time for EP-GPs and SVI-GPs is negligible, since it is computed from the analytic posterior. Also for SVI-BNNs evaluation time with 1\(k\) posterior samples is negligible.

| Configuration | MSE (×10\(^{-4}\)) | Test accuracy (%) |
|---------------|---------------------|-------------------|
|               | EP-GP | SVI-GP | SVI-BNN | EP-GP | SVI-GP | SVI-BNN |
| (a) SIR \(\beta\) | 1.99 | 1.90 | 2.05 | 100.00 | 98.80 | 100.00 |
| (b) SIR \(\gamma\) | 1.05 | 0.71 | 0.93 | 77.80 | 77.30 | 92.60 |
| (c) SIR \(\beta, \gamma\) | 1.57 | 1.45 | 0.98 | 72.75 | 85.75 | 92.25 |
| (d) PrGeEx \(k_2\) | 28.52 | 33.95 | 30.39 | 93.75 | 94.00 | 97.50 |
| (e) PrGeEx \(k_2, k_7\) | 41.96 | 11.85 | 4.24 | 78.25 | 89.25 | 95.00 |
| (f) PhosRelay \(k_1\) | 5.06 | 3.96 | 3.17 | 99.70 | 99.20 | 100.00 |
| (g) PhosRelay \(k_p, k_d\) | 8.68 | 5.32 | 3.59 | 99.25 | 96.25 | 99.75 |
| (h) PhosRelay \(k_1, k_2, k_3\) | 48.66 | 5.28 | 4.05 | 99.80 | 93.70 | 100.0 |
| (i) PhosRelay \(k_1, \ldots, k_4\) | 113.11 | 5.97 | 7.11 | 99.02 | 93.65 | 99.92 |
| (l) PhosRelay \(k_p, k_1, \ldots, k_4, k_d\) | - | 3.50 | 2.43 | - | 97.02 | 99.80 |

Table 2: Mean Squared Error (×10\(^{-4}\)) and test accuracy (%) for EP-GP, SVI-GP and SVI-BNN. SVI-BNNs are evaluated on 1\(k\) posterior samples. For each case study, we highlight the minimum MSE and the highest accuracy values.
Table 3: Average uncertainty width for EP-GP, SVI-GP and SVI-BNN compared to the average uncertainty width of the test set. SVI-BNNs are evaluated on $1k$ posterior samples. For each case study we highlight the lowest and highest uncertainty values.

![Table 3: Average uncertainty width](image)

Performance evaluation. The evaluation metrics are the mean square error (MSE), the accuracy (Acc) and the width of the uncertainty quantification area (Unc). Results are summarized in Tables 2 (MSE and Acc) and 3 (Unc).

In addition, Figures 1, 5, 6, and 7 show the results over the one-dimensional configurations - (a), (b), (d) and (f) respectively - whose results over the test set are easy to visualise. In particular, we show the mean and the 95% credible intervals of the estimated satisfaction probability $f(\theta_j)$ for EP-GPs, SVI-GPs and SVI-BNNs. Figure 2 and Figures 8 and 9 in Appendix B compare the results of EP-GPs, SVI-GPs and SVI-BNNs over the two-dimensional configurations - (c), (e) and (g). In particular, we compare the SMC satisfaction probability $f_\phi(\theta_j)$ to the average satisfaction probability $E[f(\theta_j)]$ estimated by EP-GPs, SVI-GPs and SVI-BNNs over each input $\theta_j$ of the test set. Figures 3-12 show the width of the credible intervals against the width of SMC confidence intervals. In particular, for one-dimensional configurations (Fig. 3 and Fig. 10 in Appendix B) we show how the width varies w.r.t. parameter values. In all the remaining configurations (Fig. 4 and Fig. 11, 12 in Appendix B), we show the distribution of the width over the test set.

We now compare the numerical results obtained by the variational approaches of SV-smMC to those of the smMC baseline based on EP-GP. Table 2 shows how the MSE of SV-smMC solutions is almost always lower than that of smMC. In particular, SVI-GP outperforms SVI-BNN on low-dimensional configurations, whereas SVI-BNN performs better in high-dimensional scenarios. In addition, the baseline solution presents an MSE that grows proportionally to the complexity and the dimensionality of the underlying configuration. On the contrary, SV-smMC solutions do not reflect such behaviour, as the value of the MSE is almost constant across all the different configurations.
All models reach extremely high accuracies over the test set. SVI-BNN reaches the best performances over all the configurations: the average accuracy is around 97.68% and it is always higher than 92.25%. This result is not surprising, given that SVI-BNN shows low MSEs and slightly over-conservative credible intervals (see Table 3).

About the informativeness of uncertainty estimations, we notice how SVI-BNN tends to produce credible intervals that are always larger than the one of SVI-GP, which, in turn, tends to underestimate the underlying uncertainty. This phenomenon appears in all the different configurations and it is easily observable in Fig. 3-12. On the other hand, the baseline smMC tends to have tight uncertainty estimates on low-dimensional configurations, but it becomes excessively over-conservative in high-dimensional configurations, making the predicted credible intervals almost uninformative.

Discussion. To summarize, we can see how, in general, SV-smMC solutions scale a lot better to high-dimensional problems compared to smMC, both in terms of...
feasibility and in terms of quality of the results. SVI-BNN takes longer training times (for small parameter spaces), reaches the highest accuracy and provides rather conservative predictions. SVI-GP, on the other, is the most efficient with respect to training times in low dimensions, reaches low MSEs and tends to provide overconfident predictions. Finally, we see how EP-GP is competitive only on extremely simple configurations. As the dimensionality increases, so does the error: the MSE increases and the credible intervals becomes excessively broad. Moreover, we soon reach the memory-bound wall that makes EP-GP solution unfeasible on configurations with more than four parameters.

5 Conclusions

This paper presents SV-smMC, an extension of Smoothed Model Checking, based on stochastic variational inference, that scales well to high dimensional parameter spaces and that enables GPU acceleration. In addition, this paper offers a comparison of the performances of stochastic variational inference over two different Bayesian approaches - namely Gaussian processes (SVI-GP) and Bayesian neural networks (SVI-BNN) - against those of the baseline smMC, based on the expectation propagation technique. In particular, our experiments show that the posterior predictive distribution provided by SVI-BNN provides the best overall results in terms of the estimated satisfaction probabilities. On the other hand, thanks to GPU acceleration, SVI-GP is able to achieve competitive performances with a significant speedup in computational time. Furthermore, we show how variational approaches are able to overcome the computational limit of expectation propagation algorithm over large datasets.

SV-smMC can be naturally extended with active learning ideas, following the line of [5,1,17], to solve efficiently parameter synthesis and design tasks.
References

1. Bartocci, E., Bortolussi, L., Nenzi, L., Sanguinetti, G.: System design of stochastic models using robustness of temporal properties. Theor. Comput. Sci. 587, 3–25 (2015). https://doi.org/10.1016/j.tcs.2015.02.046, https://doi.org/10.1016/j.tcs.2015.02.046
2. Bingham, E., Chen, J.P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P.A., Horsfall, P., Goodman, N.D.: Pyro: Deep universal probabilistic programming. J. Mach. Learn. Res. 20, 28:1–28:6 (2019), http://jmlr.org/papers/v20/18-403.html
3. Bishop, C.M.: Pattern recognition and machine learning, Springer (2006)
4. Bortolussi, L., Milios, D., Sanguinetti, G.: Smoothed model checking for uncertain continuous-time markov chains. Information and Computation 247, 235–253 (2016)
5. Bortolussi, L., Silvetti, S.: Bayesian statistical parameter synthesis for linear temporal properties of stochastic models. In: Beyer, D., Huisman, M. (eds.) Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018. Lecture Notes in Computer Science, vol. 10806, pp. 396–413. Springer (2018). https://doi.org/10.1007/978-3-319-89963-3_23, https://doi.org/10.1007/978-3-319-89963-3_23
6. Deodato, G., Ball, C., Zhang, X.: Bayesian neural networks for cellular image classification and uncertainty analysis. bioRxiv p. 824862 (2019)
7. Gardner, J.R., Pleiss, G., Bindel, D., Weinberger, K.Q., Wilson, A.G.: Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. p. 7587–7597. NIPS’18, Curran Associates Inc., Red Hook, NY, USA (2018)
8. Gillespie, D.T.: Exact stochastic simulation of coupled chemical reactions. The journal of physical chemistry 81(25), 2340–2361 (1977)
9. Hensman, J., Fusi, N., Lawrence, N.D.: Gaussian processes for big data. arXiv preprint arXiv:1309.6835 (2013)
10. Hensman, J., Matthews, A., Ghahramani, Z.: Scalable variational gaussian process classification. In: Artificial Intelligence and Statistics. pp. 351–360. PMLR (2015)
11. MacKay, D.J.: A practical bayesian framework for backpropagation networks. Neural computation 4(3), 448–472 (1992)
12. Maler, O., Nickovic, D.: Monitoring temporal properties of continuous signals. In: Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, pp. 152–166. Springer (2004)
13. Massart, P.: The tight constant in the dvoretzky-kiefer-wolfowitz inequality. The annals of Probability pp. 1269–1283 (1990)
14. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems. pp. 8024–8035 (2019)
15. Piho, P., Hillston, J.: Active and sparse methods in smoothed model checking. In: International Conference on Quantitative Evaluation of Systems. pp. 217–234. Springer (2021)
16. Rasch, D., Pilz, J., Verdooren, L., Gebhardt, A.: Optimal experimental design with R. Chapman and Hall/CRC (2011)
17. Silvetti, S., Policriti, A., Bortolussi, L.: An active learning approach to the falsification of black box cyber-physical systems. In: Polikarpova, N., Schneider, S.A. (eds.) Integrated Formal Methods - 13th International Conference, IFM 2017, Turin, Italy, September 20-22, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10510, pp. 3–17. Springer (2017). https://doi.org/10.1007/978-3-319-66845-1_1, https://doi.org/10.1007/978-3-319-66845-1_1

18. Titsias, M.: Variational learning of inducing variables in sparse gaussian processes. In: Artificial intelligence and statistics. pp. 567–574. PMLR (2009)

19. Van der Vaart, A.W.: Asymptotic statistics, vol. 3. Cambridge university press (2000)

20. Younes, H.L., Simmons, R.G.: Statistical probabilistic model checking with a focus on time-bounded properties. Information and Computation 204(9), 1368–1409 (2006)

21. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to simulink/stateflow verification. In: Proceedings of the 13th ACM international conference on Hybrid systems: computation and control. pp. 243–252 (2010)
A Case Studies

- **Network Epidemics (SIR):** spread of an epidemic in a population of fixed size. Here we consider the case of permanent immunisation.

  Reactions:
  \[ R_1 : S + I \rightarrow I + I \text{ with rate function } k_1X_SX_I, \]
  \[ R_2 : I \rightarrow R \text{ with rate function } k_rX_I. \]
  Initial state: \( X_S = 95, X_I = 5, X_R = 0 \) and \( N = 100 \) (\( \Delta t = 0.5 \)). In configuration (a), \( \beta \) varies in the interval \([0.005,0.3]\) and \( \gamma \) varies in \([0.005,0.3]\) and \( \gamma \) varies in \([0.005,0.2]\); whereas in (b), \( \beta = 0.12 \) and \( \gamma \) varies in \([0.005,0.2]\), whereas in (c), \( \beta \) varies in the interval \([0.005,0.3]\) and \( \gamma \) varies in \([0.005,0.2]\).

- **Prokaryotic Gene Expression:** a more complex model that captures LacZ protein synthesis in E. coli.

  Mass-action reactions:
  \[ R_1 : PLac + RNAP \rightarrow PLacRNAP \text{ with rate } k_1, \]
  \[ R_2 : PLacRNAP \rightarrow PLac + RNAP \text{ with rate } k_2, \]
  \[ R_3 : PLacRNAP \rightarrow TrLacZ1 \text{ with rate } k_3, \]
  \[ R_4 : TrLacZ1 \rightarrow RbsLacZ + PLac + TrLacZ2 \text{ with rate } k_4, \]
  \[ R_5 : TrLacZ2 \rightarrow RNAP \text{ with rate } k_5, \]
  \[ R_6 : Ribosome + RbsLacZ \rightarrow RbsRibosome \text{ with rate } k_6, \]
  \[ R_7 : RbsRibosome \rightarrow Ribosome + RbsLacZ \text{ with rate } k_7, \]
  \[ R_8 : RbsRibosome \rightarrow TrRbsLacZ + RbsLacZ \text{ with rate } k_8, \]
  \[ R_9 : TrRbsLacZ \rightarrow LacZ \text{ with rate } k_9, \]
  \[ R_{10} : PLacZ \rightarrow dgrLacZ \text{ with rate } k_{10}, \]
  \[ R_{11} : RbsLacZ \rightarrow dgrRbsLacZ \text{ with rate } k_{11}. \]

  Initial state: \( PLac = 1, RNAP = 35, Ribosome = 350, PLacRNAP = TrLacZ1 = RbsLacZ = TrLacZ2 = RbsRibosome = TrRbsLacZ = LacZ = dgrLacZ = dgrRbsLacZ = 0. \)

  The default parametric values are \( k_1 = 0.17, k_2 = 10, k_3 = k_4 = 1, k_5 = k_9 = 0.015, k_6 = 0.17, k_7 = 0.45, k_8 = 0.4, k_{10} = 0.00000642, k_{11} = 0.3 \). In configuration (d), \( k_2 \) varies in \([10,10000)\), whereas in configuration (c), \( k_2 \) varies in \([10,10000]\) and \( k_7 \) varies in \([0.45,4500]\). In configuration (e), \( k_2 \) varies in \([10,10000]\) and \( k_7 \) varies in \([0.45,4500]\).

- **Three-layer Phosphorelay:** network of three layers \( L_1, L_2, L_3 \). Each layer can be found also in phosphorylate form \( L_1p, L_2p, L_3p \) and there is a ligand \( B \). Initial state: \( L_1p = L_2p = L_3p = L_4p = B = 0, L_1 = L_2 = L_3 = 32 \) and \( N = 5000. \)

  Reactions:
  \[ R_1 : \emptyset \rightarrow B \text{ with rate function } k_p \]
  \[ R_2 : L_1 + B \rightarrow L_1p + B \text{ with rate function } k_1 \cdot L_1 \cdot B/N \]
  \[ R_3 : L_1p + L_2 \rightarrow L_1 + L_2p \text{ with rate function } k_2 \cdot L_1p \cdot L_2/N \]
  \[ R_4 : L_2p + L_3 \rightarrow L_2 + L_3p \text{ with rate function } k_3 \cdot L_2p \cdot L_3/N \]
  \[ R_5 : L_3p \rightarrow L_3 \text{ with rate function } k_4 \cdot L_2p/N \]
  \[ R_6 : B \rightarrow \emptyset \text{ with rate function } k_d \cdot B \]

  The default parametric values are \( k_p = 0.1, k_d = 0.05, k_1 = k_2 = k_3 = 1 \) and \( k_4 = 2 \). In configuration (f), \( k_1 \) varies in \([0.1,2]\); in (g), \( k_p \) varies in \([0.01,0.2]\) and \( k_4 \) varies in \([0.005,0.1]\); in (h), \( k_1, k_2 \) and \( k_3 \) all vary in the interval \([0.1,2]\); in (i), \( k_1, k_2, k_3 \) vary in the interval \([0.1,2]\) and \( k_4 \) varies in \([0.5,5]\); whereas in configuration (l), \( k_1, k_2, k_3, k_4 \) vary as in (i) and \( k_2, k_3 \) varies as in (g).
B Additional Plots

Fig. 5: Satisfaction probability estimated by EP-GPs, SVI-GPs and SVI-BNNs and true satisfaction probability on 30 equispaced points from the test set on configuration (b), with 95% confidence intervals around the mean. SVI-BNNs are evaluated on $1k$ posterior samples.

Fig. 6: Satisfaction probability estimated by EP-GPs, SVI-GPs and SVI-BNNs and true satisfaction probability on 30 equispaced points from the test set on configuration (d), with 95% confidence intervals around the mean. SVI-BNNs are evaluated on $1k$ posterior samples.
Fig. 7: Satisfaction probability estimated by EP-GPs, SVI-GPs and SVI-BNNs and true satisfaction probability on 30 equispaced points from the test set on configuration (f), with 95% confidence intervals around the mean. SVI-BNNs are evaluated on 1k posterior samples.

Fig. 8: True satisfaction probability is compared to the satisfaction probability estimated by EP-GPs, SVI-GPs and SVI-BNNs on the test set for configuration (e). SVI-BNNs are evaluated on 1k posterior samples.

Fig. 9: True satisfaction probability is compared to the satisfaction probability estimated by EP-GPs, SVI-GPs and SVI-BNNs on the test set for configuration (g). SVI-BNNs are evaluated on 1k posterior samples.
Fig. 10: Uncertainty of true (test) and predicted (EP-GP, SVI-GP, SVI-BNN) satisfaction probabilities for models trained on configuration (d) (left) and on configuration (f) (right). SVI-BNNs are evaluated using 1k posterior samples.

Fig. 11: Distribution of uncertainty for test parameters’ tuples of true (test) and predicted (EP-GP, SVI-GP, SVI-BNN) satisfaction probabilities. Models are trained on configurations (e) (left plot) and (g) (right plot). SVI-BNNs are evaluated using 1k posterior samples.

Fig. 12: Distribution of uncertainty for test parameters’ tuples of true (test) and predicted (EP-GP, SVI-GP, SVI-BNN) satisfaction probabilities. Models are trained on configurations (h) (left plot) and (l) (right plot). Notice that EP-GP is unfeasible in the second case study. SVI-BNNs are evaluated using 1k posterior samples.