Automated Experiment Design for Data-Efficient Verification of Parametric Markov Decision Processes

E. Polgreen[1], V.B. Wijesuriya[1], S. Haesaert[2], and A. Abate[1]

[1] Department of Computer Science, University of Oxford
[2] Department of Electrical Engineering, Eindhoven University of Technology

Abstract. We present a new method for statistical verification of quantitative properties over a partially unknown system with actions, utilising a parameterised model (in this work, a parametric Markov decision process) and data collected from experiments performed on the underlying system. We obtain the confidence that the underlying system satisfies a given property, and show that the method uses data efficiently and thus is robust to the amount of data available. These characteristics are achieved by firstly exploiting parameter synthesis to establish a feasible set of parameters for which the underlying system will satisfy the property; secondly, by actively synthesising experiments to increase amount of information in the collected data that is relevant to the property; and finally propagating this information over the model parameters, obtaining a confidence that reflects our belief whether or not the system parameters lie in the feasible set, thereby solving the verification problem.

1 Introduction

Formal verification relies on full access to accurate models describing the behaviour of systems in order to guarantee their correctness. Such models are often hard to obtain for systems encompassing partially understood behaviours and uncertain events. For a partially unknown system, the unknown model characteristics can be represented via non-determinism in the form of parameters. The resulting parameterised model captures all available knowledge on the underlying system of interest.

We target the verification of a fragment of Probabilistic Computation Tree Logic (PCTL) on partially unknown systems with actions. We develop a new approach that incorporates the available information captured by a parameterised model with the active collection of a limited amount of data from the underlying system. The verification problem is tackled in three phases. In the first phase we use the available parameterised model to synthesise the set of parameters for which the property of interest is satisfied (called the *feasible set*).

In the second phase a series of experiments are designed and executed on the system to update the knowledge available about the parameters of the parameterised model. More precisely, a procedure executes the designed experiments, obtains data from the system and, by means of Bayesian statistics, updates distributions over the likely parameter values of the parameterised model. This updated knowledge is returned to the experiment design module, and the process is repeated until a preset limit on the total amount of collectible data is reached. The design of such experiments is important to attain a reasonable level of confidence in the acceptance or rejection of a property with a limited amount of data.

In the final phase, we combine the output from the parameter synthesis with the updated distributions over the model parameters to quantify the confidence that the system satisfies (or does not satisfy) the property.

This work extends the contributions in [19] by focussing on systems with actions: the presence of (action) non-determinism provides the potential for experiment design, whereby we select actions to improve the accuracy of our confidence value. More precisely we design experiments that maximise the usefulness of the data collected. Intuitively, this means that we want to design experiments to prioritise the collection of data that leads to proving or disproving the satisfaction of the property. In this work, we present the complete approach, and evaluate the contribution of our experiment design procedure . We argue that *automated experiment design* allows us to draw sensible conclusions robustly with a limited amount of data.

Structure of the paper. Section 2 provides the necessary background information for the rest of paper to build upon. Section 3 presents an overview of our algorithm. Subsequent sections detail the different phases of the algorithm: in Section 4 we show how we collect data; Section 5 provides details on the confidence computation; and the key contribution of this work is Section 6, which outlines our experiment design approach.

1.1 Problem Statement

Consider a partially unknown system $\mathbf{S}$, with external non-determinism in the form of actions, and suppose we can gather a limited amount of sample trajectories from this system. Assume the partial knowledge about the system is encompassed within a parameterised model class describing the behaviour of $\mathbf{S}$. We investigate two sub-problems:

- Can we efficiently use this limited amount of data from a system $\mathbf{S}$ to quantify a confidence that the system $\mathbf{S}$ verifies a given PCTL property?
- How should we design an experiment on the system such that the gathered data allows us to verify the property with the greatest degree of accuracy? Let the choice of actions of system $\mathbf{S}$ be something we can control during the experiment, and let there only be a limited amount of available experiment time; can we optimise the sequence of actions to increase the accuracy of the confidence quantification?

1.2 Related Work

We compare our work to two branches of research: Statistical Model Checking (SMC) and research concerned with learning models from system data. We contrast our experiment design method with existing strategy synthesis techniques for *fully known* Markov decision processes (MDPs).

We emphasise that we tackle a different problem than SMC: we target partially unknown systems and gather data from the underlying system; SMC[16] targets fully known models that are too big for conventional verification, and generates large amounts of data from the models themselves. When applied to model-free scenarios [23, 24], SMC generates this data from the underlying system. By using partial model knowledge, we substantially reduce this data requirement. In addition, SMC for systems with non-determinism [4, 12] considers only bounded-time properties, and depends on the ability to generate traces from the model of length greater than the bound. By incorporating parameter synthesis tools, we are able to consider unbounded-time properties and to draw conclusions from much shorter traces.

Research on learning models from system data is broad. [18, 22] use a Bayesian approach to learn full Markov models of completely unknown systems. Our work uses a similar Bayesian method but differs because we include information from the partial model, which allows us to consider known relationships between parameters and thus reduce the amount of data needed for inference. [1, 3] use active learning to discover full MDP models from data, prioritising actions by *variance minimisation* or *KL divergence*. The inclusion of a partial model in our method allows us to instead prioritise gathering data that contributes to the acceptance or rejection of a given property over the system. Although [3] learns the model with the goal of system verification, the authors provide no means of quantifying a confidence that the system satisfies the property, as they do not have a way to assess which transition probabilities have the greatest contribution to the satisfaction of the property.

Considering different model classes, experiment design is also used in system identification [7]. Recent studies [10, 11] have incorporated experiment design to data-driven statistical verification over *dynamical systems* with partly unknown dynamics, controllable inputs and noisy measurements. Similar to our approach, they also compute a confidence estimate on the properties of interest by gathering data through *optimal experiment design*.

Action selection for Markov decision processes, though in our context used for experiment design, is a known problem that in general amounts to synthesising strategies. [15] presents an overview for MDPs with static rewards, and [8] provides solutions for MDPs with non-Markovian rewards. Closer to our approach, [9] synthesises strategies for MDPs online, where an agent learns a state cost only after selecting an action. [13] use inference-based techniques over strategies to pick a strategy that maximises the expected reward for an MDP with arbitrary rewards.

2 Background

We model a fully known system as a Markov decision process [2].

Definition 1 *A discrete-time Markov decision process (MDP)* $\mathbf{M}$ *is a tuple* $(S, \mathrm{Act}, \mathbb{T}, \iota_{init}, \mathrm{AP}, L)$*, where:*

- S *is a finite, non-empty set of states,*
- Act *is a set of actions,*
- $\mathbb{T} : S \times Act \times S \to [0, 1]$ *is the transition probability function, such that* $\forall s \in S$ *and* $\forall \alpha \in \mathrm{Act}$*,* $\sum_{s' \in S} \mathbb{T}(s, \alpha, s') \in \{0, 1\}$*,*
- $\iota_{init} : S \to [0, 1]$ *denotes an initial probability distribution over the states* S*, such that* $\sum_{s \in S} \iota_{init}(s) = 1$*,*
- *The states in* S *are labelled with atomic propositions* $a \in \mathrm{AP}$ *via the labelling function* $L : S \to 2^{\mathrm{AP}}$*.*

An action $\alpha \in Act$ *is enabled in state* s *if and only if* $\sum_{s' \in S} \mathbb{T}(s, \alpha, s') = 1$*. Let* $\mathrm{Act}(s)$ *denote the set of enabled actions in* s*. For any state* $s \in S$*, it is required that* $\mathrm{Act}(s) \neq \varnothing$*. Each state* $s' \in S$ *for which* $\mathbb{T}(s, \alpha, s') > 0$ *is called an* α*-successor of* s*. Those states* s *satisfying the condition* $\iota_{init}(s) > 0$ *are called initial states.*

We assume that the MDP is not known exactly, and instead belongs to the set of MDPs represented by a parametric Markov decision process.

Definition 2 *A discrete-time parametric Markov decision process (pMDP) is a tuple* $\mathbf{M}_\Theta = (S, \mathrm{Act}, \mathbb{T}_\theta, \iota_{init}, \mathrm{AP}, L, \Theta)$*, where* $S, \iota_{init}, \mathrm{Act}, \mathrm{AP}, L$ *are as in Definition 1. The entries in* $\mathbb{T}_\theta$ *are specified in terms of parameters, collected in a parameter vector* $\theta \in \Theta$*, where* Θ *is the set of all possible evaluations of* θ*. Each evaluation gives rise to an induced Markov decision process* $\mathbf{M}(\theta)$*.*

$\forall s \in S, \forall \alpha \in \mathrm{Act}(s), \forall \theta \in \Theta : \sum_{s' \in S} \mathbb{T}_\theta(s, \alpha, s') = 1$, namely any $\theta \in \Theta$ induces an MDP $\mathbf{M}(\theta)$ where the transition function $\mathbb{T}_\theta$ can be represented by a *stochastic matrix*. We also assume a prior distribution on the model parameters (to be used in Bayesian inference). We assume all non-parameterised transition probabilities are known exactly.

As in [19], we consider *linearly parameterised* MPDs, where unknown transition probabilities can be linearly related. More precisely, given $\Theta \subseteq [0, 1]^n$ and parameter vector $\theta = (\theta_1, \ldots, \theta_n) \in \Theta$ with $\theta_i \in [0, 1]$, a pMDP is considered linearly parameterised if all outgoing transition probabilities of state-actions pairs have probability $g_l(\theta)$ or $1 - g_l(\theta)$, where $g_l(\theta) = k_0 + k_1\theta_1 + ... + k_n\theta_n$ with $k_i \in [0, 1]$ and $\sum k_i \leq 1$. This restriction is due to the transformations presented in [19] necessary to perform Bayesian inference over the model parameters. As before, $\forall s \in S, \forall \alpha \in \mathrm{Act}(s), \forall \theta \in \Theta : \sum_{s' \in S} \mathbb{T}_\theta(s, \alpha, s') = 1$.

2.1 Strategies

A strategy for an MDP resolves nondeterminism by choosing an action in each state of the model. In our work experiment design amounts to synthesising a strategy for an MDP, i.e., a sequence of actions, under which we generate data from the system. We focus on *deterministic memoryless* strategies in this paper, i.e., strategies that always pick the same action in any given state, independent of the history of states already visited. Future work will extend to both memory-dependent and randomised strategies.

Definition 3. *A deterministic memoryless strategy for an MDP M is a function $\pi : S \to \mathrm{Act}$ s.t. $\pi(s) \in \mathrm{Act}(s)\ \forall_{s \in \mathbf{S}}$.*

2.2 Properties – Probabilistic Computational Tree Logic

We consider system specifications (aka properties) given in a fragment of Probabilistic Computational Tree Logic (PCTL) [2]. Since we use PRISM [14] for parameter synthesis, we consider *non-nested*, *unbounded-time "until"* properties expressed in PCTL.

Definition 4 *Let a discrete-time MDP be given. Let ϕ be a formula interpreted over states $s \in S$, and φ be a formula interpreted on paths of the MDP. Also, let $\bowtie \in \{<, \leq, \geq, >\}$, $n \in \mathbb{N}$, $p \in [0, 1]$, $c \in AP$. The Syntax of the PCTL fragment we consider is given by:*

$$\phi := \text{true} \mid c \mid \phi \wedge \phi \mid \neg\phi \mid \mathbf{P}_{\bowtie p}(\varphi), \qquad \varphi := \bigcirc\, \phi \mid \phi\ \mathcal{U}\ \phi.$$

Definition 5 *Consider a PCTL formula $\phi := \mathbf{P}_{\bowtie p}(\phi_1\ \mathcal{U}\ \phi_2)$. Let $\mathbb{P}^{\pi}_{\mathbf{M}}(s, \varphi)$ denote the probability associated to the paths of an MDP $\mathbf{M}$ starting from $s \in S$ satisfying the path formula φ under the strategy π. Let $\mathfrak{A}(\mathbf{M})$ denote all deterministic memoryless strategies for $\mathbf{M}$. The satisfaction of the formula ϕ by M is given by:*

$$\mathbf{M} \models \mathbf{P}_{\bowtie p}(\phi_1\ \mathcal{U}\ \phi_2) \iff \forall s \in S, \iota_{init}(s) > 0 : \min_{\pi \in \mathfrak{A}(\mathbf{M})} \mathbb{P}^{\pi}_{\mathbf{M}}(s, \phi_1\ \mathcal{U}\ \phi_2) \bowtie p.$$

We introduce the *feasible set* of parameters, denoted Θ_ϕ, which is the set of parameter evaluations for which the property is satisfied.

Definition 6 *Let $\mathbf{M}(\theta)$ be an induced MDP of the pMDP $\mathbf{M}_\Theta$, indexed by parameter vector $\theta \in \Theta$. Let ϕ be a formula in PCTL. The feasible set Θ_ϕ is defined as: $\theta \in \Theta_\phi \iff \mathbf{M}(\theta) \models \phi$.*

We use $\mathbb{P}(A)$ to denote the probability of an event A, $p(\cdot)$ to represent probability density functions and $\mathbf{P}_{\bowtie p}(\cdot)$ for the probabilistic operator in PCTL.

3 Overview of the Method

Our method is made up of three distinct phases, as shown in Fig. 1.

1. We use a parameter synthesis tool to determine a set of feasible parameters for which the property is satisfied by the system, based on the given parametric Markov decision process, see Section 3.1.
2. (a) We synthesise a strategy for collecting data, based on the feasible set and the prior distribution over the parameters, see Section 6.
   (b) We collect data from the underlying system using the synthesised strategy, see Section 4.
   (c) We use Bayesian inference to infer a distribution over the likely values of the parameters, based on the collected data, and update the respective prior distributions with the new information, see Section 4. If we can sequentially collect more data, loop back to step 2 (a).
3. We compute the confidence that the system satisfies the property, based on the data collected, see Section 5.

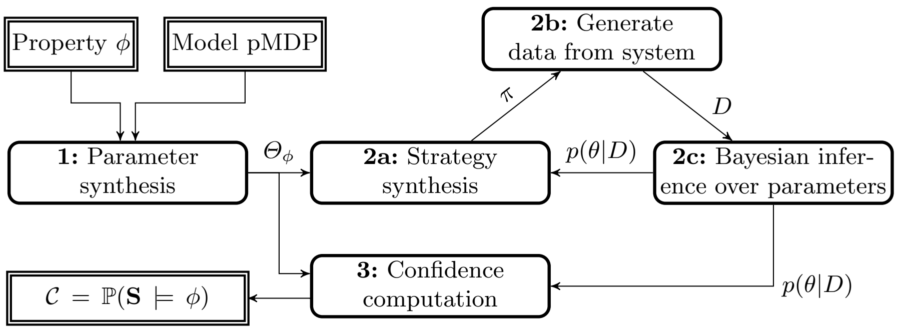

Property ϕ
Model pMDP
2b: Generate data from system
π
D
1: Parameter synthesis
Θ_ϕ
2a: Strategy synthesis
$p(\theta|D)$
2c: Bayesian inference over parameters
3: Confidence computation
$p(\theta|D)$
$\mathcal{C} = \mathbb{P}(\mathbf{S} \models \phi)$

Fig. 1: Overview of the verification procedure.

3.1 Parameter Synthesis

The first phase of the method uses parameter synthesis to find the feasible set of parameters, namely parameter evaluations corresponding to models of the considered pMDP that satisfy the given PCTL property. This step leads to the set of parameters $\Theta_\phi = \{\theta \in \Theta : \mathbf{M}(\theta) \models \phi\}$.

The output of the parameter synthesis procedure is a mapping from hyper-rectangles (which are subsets of parameter evaluations) to truth values, namely "true" if the property is satisfied in the hyper-rectangle and "false" otherwise.

Implementation: We use PRISM [14] for parameter synthesis: the tool computes a rational function of the parameters, which expresses the result obtained from model checking the PCTL property on the parameterised model. Our approach can also make use of Storm [21], which shows potential to be scalable to much larger systems. Storm *lifts* a parametric Markov decision process to a parameter-free Stochastic Game (SG) between two players, and solves the resulting SG via standard value iteration.

4 Bayesian Inference in Parametric Markov Decision Processes

In this work, we collect data from the underlying system and use Bayesian learning to infer a probability distribution over parameters of the pMDP model based on the collected data. Bayesian inference maintains a probability distribution over these parameters and updates the distribution by employing Bayes' rule as more observations are gathered [22]. An initial prior distribution $p(\theta)$ is assumed.

Data. We collect *finite traces* from the underlying system, in the form of a sequence of visited states and actions. We use D to denote a set of finite traces. We split the data into transition counts: D_{s_k,α_1,s_l} denotes the number of times the transition from s_k to s_l under action α_1 appears within the data set D. Each transition count is the outcome of an independent trial in a multinomial distribution[3] with event probabilities given by the transition probabilities.

Assume for now that the transitions are parameterised either with constants or with single parameters of the form θ_i or $1-\theta_i$. We can group transition counts for identically parameterised transitions. We shall denote by D_{θ_j} the transition counts for all transitions with probability given by θ_j.

We wish to obtain posterior distributions for each parameter via *marginal distributions* (which, in this case, are binomial distributions), by applying *parameter-tying* [20] techniques. We thus obtain a number of transition counts for $1-\theta_j$ as the sum of all transitions not parameterised with θ_j, under an action that has a transition parameterised with θ_j, and denote it by $D_{\neg\theta_j}$. Hence D_{θ_j} and $D_{\neg\theta_j}$ are calculated as:

$$D_{\theta_j} = \sum_{s_i \in S, s_l \in S, \alpha_k \in Act} D_{s_i,\alpha_k,s_l} \text{ for } \mathbb{T}(s_i,\alpha_k,s_l) = \theta_j, \text{ and}$$

$$D_{\neg\theta_j} = \sum_{s_i \in S, s_l \in S, \alpha_k \in Act} D_{s_i,\alpha_k,s_l} \text{ for } \mathbb{T}(s_i,\alpha_k,s_l) \neq \theta_j \wedge \exists s_m \in S : \mathbb{T}(s_i,\alpha_k,s_m) = \theta_j.$$

Let $D_{\theta_j,\neg\theta_j}$ denote the pair $(D_{\theta_j}, D_{\neg\theta_j})$. For parameterisations where the transition probabilities are expressed as linear functions of parameters, we obtain

[3] A multinomial distribution is defined by its density function $f(\cdot \mid p, N) \propto \prod_{i=1}^{k} p_i^{n_i}$, for $n_i \in \{0,1,...,N\}$ and such that $\sum_{i=1}^{k} n_i = N$, where $N \in \mathbb{N}$ is a parameter and p is a discrete distribution over k outcomes.

$D_{\theta_j,\neg\theta_j}$ by the same procedure that [19] uses. We expand the markov decision process, introducing new states and new transitions, allowing us to force all transition probabilities to be expressed as constants, or in the form of θ_j or $1-\theta_j$, for any parameter $\theta_j \in \theta$. We can then represent the parameter counts over parameters in the new transitions as multinomial distributions. We omit the detail here and refer the reader to the extended version of this paper, and the original work [19].

Bayesian Inference with Data. Consider a parametric Markov decision process $\mathbf{M}_\Theta = (S, \text{Act}, \mathbb{T}_\theta, \iota_{init}, \text{AP}, L, \Theta)$ with $\Theta \subseteq [0,1]^n$. Suppose that we have obtained D_{θ_j} and $D_{\neg\theta_j}$ for all $\theta_j \in \theta$, and that we have assumed non-informative, uniform prior distributions for all parameters $\theta_j \in \theta$, denoted by $p(\theta_j)$. The posterior density $p(\theta_j \mid D)$ is given by Bayes' rule:

$$p(\theta_j \mid D) = \frac{\mathbb{P}(D \mid \theta_j)p(\theta_j)}{\mathbb{P}(D)} = \frac{p(\theta_j)\theta_j^{D_{\theta_j}}(1-\theta_j)^{D_{\neg\theta_j}}}{\mathbb{P}(D_{\theta_j,\neg\theta_j})}.$$

A standard approach [5, 17, 22] is to consider the prior to be a Dirichlet distribution. The posterior distribution is then updated by adding the event counts to the hyperparameters of the prior. The Dirichlet prior distribution for the pair $(\theta_j, 1-\theta_j)$ is denoted as $\text{Dir}(\theta_j \mid \mu^{\theta_j})$ with hyperparameters $\mu^{\theta_j} = (\mu_1^{\theta_j}, \mu_2^{\theta_j})$. Thus, the updated posterior distribution for the parameter θ_j is given as: $\theta_j \sim p(\theta_j \mid D) = \text{Dir}(\theta_j \mid D_{\theta_j,\neg\theta_j} + \mu^{\theta_j})$.

The posterior distribution for the entire parameter vector θ, given by $p(\theta \mid D)$ is equal to the product of the posterior distributions for all $\theta_i \in \theta$. This holds due to the independence of each θ_i over independent state-action pairs in the pMDP. Note that, if we have a linearly parameterised MDP, we obtain some of the transition counts in the form of multinomial distributions. We hence obtain realisations of the posterior by a sampling procedure from [19] as explained in the extended version of this paper.

5 Computation of Confidence

We determine a confidence, $\mathcal{C}$, for the satisfaction of a PCTL formula ϕ by a system $\mathbf{S}$ of interest. We first presented this procedure in previous work [19], and we need no extension to this due to the external nondeterminism being factored out in the Bayesian inference calculation given in the previous section.

Definition 7. *Given a PCTL formula ϕ that has a binary satisfaction function, i.e., the property is either satisfied or not, and posterior distributions $p(\theta_i \mid D)$ for all $\theta_i \in \theta$, as obtained in the previous section, the confidence in $\boldsymbol{S} \models \phi$ can be quantified by Bayesian inference as*

$$\mathcal{C} = \mathbb{P}(\boldsymbol{S} \models \phi \mid D) = \textstyle\int_{\Theta_\phi} \prod_{\theta_i \in \theta} p(\theta_i \mid D_{\theta_i,\neg\theta_i}) d\theta, \tag{1}$$

The operation shown in Eq. (1) is equivalent to computing the confidence that each parameter is within its feasible set, and then taking the product of all the parameter confidence values. The integral of a Dirichlet distribution is hard to compute using analytical methods, and so we use Monte Carlo integration. This also allows integration with the calculation of the posterior distribution for pMDPs with linear parameterisations, where we have obtained the posterior distribution by means of sampling, as described in [19].

6 Online Experiment Design

The key contribution in this paper is the design of experiments to generate maximally useful data. We describe in the preceding sections how we use a limited amount of data efficiently to obtain a confidence that the system satisfies the property. In this section, we propose a method for selecting the deterministic memoryless strategy that provides the most useful data to input into our confidence computation in Section 5. This allows us to compute the most accurate confidence value for the finite data set of limited size, i.e., the confidence should be high if the underlying system satisfies the property, and low if the underlying system does not satisfy the property.

6.1 Predicted Confidence

We predict the confidence after taking a transition from state s under action α. We define the predicted confidence, $\mathcal{C}^{\text{pred}}_{s,\alpha}$, to be the confidence computed using the *expected parameter counts*, after taking a single transition from s under action α: these are denoted by $\mathbb{E}_{s,\alpha}\left(D_{\theta_i,\neg\theta_i}\right)$ for all $\theta_i \in \theta$. Formally,

$$\mathcal{C}^{\text{pred}}_{s,\alpha} = \int_{\Theta_\phi} \prod_{\theta_i \in \theta} p(\theta_i \mid \mathbb{E}_{s,\alpha}\left(D_{\theta_i,\neg\theta_i}\right)) d\theta,$$

where $p(\theta_i \mid \mathbb{E}_{s,\alpha}\left(D_{\theta_i,\neg\theta_i}\right))$ is the predicted posterior distribution obtained by updating the prior, $Dir(\theta_i \mid \mu^{\theta_i})$, with the expected parameter counts, i.e., $\text{Dir}(\theta_i \mid \mu^{\theta_i} + \mathbb{E}_{s,\alpha}\left(D_{\theta_i,\neg\theta_i}\right))$.

We first compute the *expected transition counts* for the state-action pair, $\mathbb{E}_{s,\alpha}\left(D_{s,\alpha}\right)$, from which we extract the *expected parameter counts* using the method in Section 4. Consider a state s with an action α, and two transitions with probabilities $\mathbb{T}_\theta(s,\alpha,s') = g_l(\theta) = k_0 + k_1\theta_1 + ... + k_n\theta_n$, and $T_\theta(s,\alpha,s) = 1 - g_l(\theta)$. The expected transition counts are given by a multinomial distribution over the outgoing transitions under that action, with event probabilities equal to the *expected transition probabilities*. Note that prior distribution for any parameter $\theta_i \in \theta$ is $Dir(\theta_i \mid \mu^{\theta_i})$. To compute the expected transition probabilities, we require the expected values of the parameters, given by $\mathbb{E}\left(\theta_i\right) = \frac{\mu_1^{\theta_i}}{\mu_1^{\theta_i} + \mu_2^{\theta_i}}$ for all $\theta_i \in \theta$. The expected value of the transition probabilities are then given by evaluating $g_l(\mathbb{E}\left(\theta\right))$ and $1 - g_l(\mathbb{E}\left(\theta\right))$. Hence the expected transition counts

$\mathbb{E}_{s,\alpha}(D_{s,\alpha,s'})$ and $\mathbb{E}_{s,\alpha}(D_{s,\alpha,s})$, are equal to the expected transition probabilities for $\mathbb{T}_\theta(s,\alpha,s')$ and $\mathbb{T}_\theta(s,\alpha,s)$. Consider only the transition parameterised with $\mathbb{T}_\theta(s,\alpha,s') = g_l(\theta)$:

$$\begin{aligned}\mathbb{E}_{s,\alpha}(D_{s,\alpha,s'}) &= \mathbb{E}(\mathbb{T}(s,\alpha,s')) = g_l(\mathbb{E}(\theta)) \\ &= k_0 + k_1\mathbb{E}(\theta_1) + \ldots + k_n\mathbb{E}(\theta_n) = k_0 + \sum_{i=1:n} k_i \frac{\mu_1^{\theta_i}}{\mu_1^{\theta_i} + \mu_2^{\theta_i}}.\end{aligned}$$

We can extract the parameter counts as described in Section 4, to obtain $\mathbb{E}_{s,\alpha}(D_{\theta_i,\neg\theta_i})$.

6.2 Optimisation of Predicted Confidence Gain

The underlying system either satisfies or does not satisfy the given property, so we wish to minimise the difference between our confidence value and the closest among 0 or 1, or to maximise the difference between a confidence of 0.5 and our confidence, i.e., the maximum absolute value of $0.5 - \mathcal{C}$. We can therefore define a predicted confidence gain for a state-action pair (s,α), denoted by $\mathbb{G}_{s,\alpha}$, as the maximisation of this difference, i.e., the biggest step towards either 0 or 1.

$$\mathbb{G}_{s,\alpha} = |0.5 - \mathcal{C}^{\text{pred}}_{s,\alpha}| - |0.5 - \mathcal{C}|$$

For a finite trace of length N, we can calculate the optimal predicted confidence gain for state s and discrete time step t, denoted by x_s^t, as

$$x_s^t = \begin{cases} \max_{\alpha \in Act(s)}(\mathbb{G}_{s,\alpha} + \sum(\mathbb{T}(s,\alpha,s').\, x_{s'}^{t+1})) & \text{if } 0 < t < N \\ 0 & \text{if } t \geq N. \end{cases}$$

It is important to note that the confidence gain is not a static quantity, because $\mathbb{G}_{s,\alpha}$ depends on the distribution over the relevant component parameters of θ at time t.

6.3 Optimal Confidence Gain: Experiment Design via Strategy Synthesis

Due to memory dependency of the confidence gain, computing an optimal strategy is intractable, and cannot be solved via conventional dynamic programming methods [8]. However, we put forward a few alternatives.

Explicitly evaluated memoryless strategies. The conventional way of solving a MDP with non-Markovian rewards is to translate the model into an equivalent MDP with Markovian rewards, whose states result from augmenting those of the original model with extra information capturing enough history to make the reward Markovian. This is in general computationally expensive [8]. Given that we will be performing strategy synthesis repeatedly in our method (i.e., once each time a new batch of data is sequentially gathered), we compromise and use a straightforward selection method to find the best memoryless strategy. This reduces the number of possible strategies and allows us to consider each possible strategy individually. We simplify the calculations in Section 6.1 to compute the expected transition counts for a full trace of length N, and then compute the predicted confidence gain for the entire memoryless strategy. This method works well for small trace lengths, however computing the expected transition counts for a full trace of length N amounts to performing a matrix multiplication N times, so this can be time consuming for large N.

Alternative off-line method. An alternative approach would be to disregard the memory dependency of the confidence gain. This corresponds to an off-line approach: we compute a strategy on the model frozen at the time we start generating traces, assuming that the prior distributions remains unchanged over the trace horizon N. We assign confidence gains to state-action pairs and treat them as static rewards. This allows us to use classical dynamic programming to find the best memoryless strategy, which would require introducing a discount factor on the rewards, to avoid infinite returns inside *strongly-connected components.* This method may be faster for long trace lengths than explicitly evaluating possible strategies, as done previously; however, the selected strategy may not be the best memoryless strategy when the trace lengths are large, and specifically when the prior distributions, which are assumed to remain unchanged, actually change significantly over time as the trace length is being reached.

Comparison. Consider the small pMDP shown in Fig. 2, parameterised with $\theta = (\theta_1, \theta_2)$, and the property $\mathbf{P}_{\leq 0.5}(\text{true}\ \mathcal{U}\ s_1)$. Both parameters have the same prior distributions and both contribute equally to the feasible set. Intuitively, choosing action α_2 or α_3 is better than choosing action α_1, because any trace starting with α_1 only contains one parameterised transition. However, it is also intuitive that choosing α_2 is better than α_3 because any trace starting with α_3 only gives us information about θ_1, whereas traces with α_2 give us information about both parameters.

The dynamic programming approach will pick nondeterministically between action α_3 and α_2 for the first trace, because the reward assigned to (s_0, α_3) is the same as the reward assigned to (s_0, α_2) as the initial priors and the feasible sets are the same. The priors will not be updated until after the full trace is collected. Our strategy synthesis approach calculates the expected updates for these priors, and will thus be able to detect a better strategy, which selects action α_2.

In our experimental evaluation, we use the explicitly evaluated memoryless strategy. Henceforth, the explicitly evaluated memoryless strategy will be referred to as the *synthesised strategy.*

7 Results

We experimentally evaluate the research questions posed in the problem statement: *question 1 – given a limited amount of data, can we use it efficiently to*

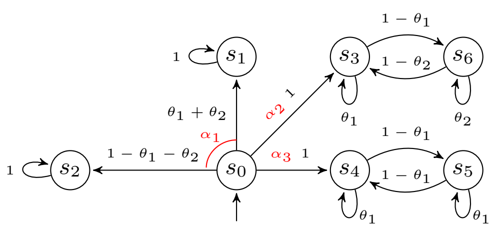


Fig. 2: Example pMDP where offline strategy synthesis may not be optimal

quantify a confidence that our system satisfies a given property? question 2 – can we design experiments that increase the accuracy of this confidence?

Experimental Set-up. Our approach is implemented in C++. We use PRISM [14] for parameter synthesis, and GSL-2.3 [6] for random number generation.

To answer question 2, we evaluate our *synthesised strategy* approach against two alternatives. The first comparison is against a memoryless strategy, randomly selected from the set of all possible memoryless strategies. We term the resultant strategy as *random static strategy.* The second comparison strategy randomly selects actions at each state as data is collected, and therefore we term it as *no strategy.* All three approaches use the same Bayesian inference framework over parameter counts.

We present the analysis of our approach on the simple pMDP model in Fig. 3 and with the PCTL property $\mathbf{P}_{\geq 0.5}(\text{true}\ \mathcal{U}\ \text{complete})$. We also run our approach on models up to 1000 states, but find the scalability depends on the number of actions in the model. We assign non-informative priors to the parameters. Note that in our model, θ_2 does not contribute to the satisfaction of the property, and having validated that this does not affect the confidence results, we set θ_2 equal to θ_1. We simulate a range of underlying systems, corresponding to models $\mathbf{M}(\theta)$ with different values for θ, which allows us to assess the accuracy of our confidence values against a ground truth, G_{true}. For a simulated system modelled by $\mathbf{M}(\theta)$, this is given by:

$$G_{true} = \begin{cases} 0 & \text{if } \theta_1 \notin [0.369, 0.75], \\ 1 & \text{if } \theta_1 \in [0.369, 0.75]. \end{cases} \tag{2}$$

We collect data from the simulated system in the form of a history of state-action pairs visited. We compute the mean squared error (MSE) between the ground truth from Eq. (2) and the confidence estimate, formally, MSE $= \frac{1}{n}\sum_{i=1}^{n}(G_{true} - G_i)^2$, where n is the number of trials and G_i is the output confidence estimate for the i-th run.

Observations and Discussion. The MSE in the confidence from all three strategies, over a range of underlying systems and varying quantities of data (i.e., for different numbers and lengths of traces), are shown in Fig. 4. The

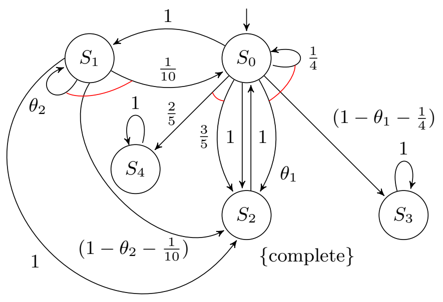
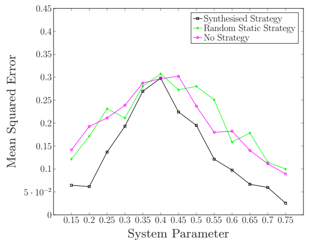
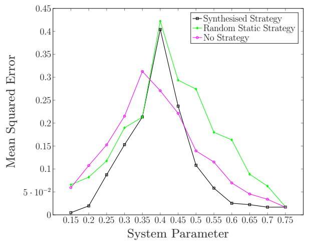
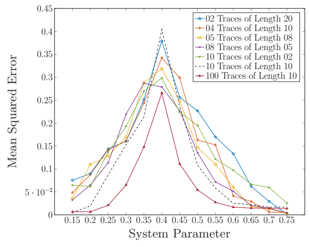


Fig. 3: A simple pMDP for the experimental evaluation.





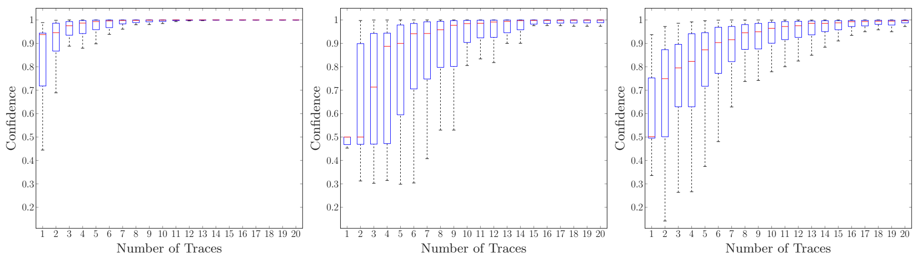


(a) All strategies ($t10$, $l02$) (b) All strategies ($t10$, $l10$) (c) Synthesised strategy

Fig. 4: Errors produced by the confidence computation for the three strategies considered. Plots (a) and (b) show the MSE for each type of strategy and for 10 traces of different trace lengths over different simulated systems. Plot (c) presents the MSE for the *synthesised strategy* over different simulated systems and combinations of number of traces with varying trace lengths. We denote by ($t10$, $l02$) a run with 10 traces, each of length 02.

convergence of the confidence outcome is shown in Fig. 5, with box plots showing the interquartile range (IQR), omitting any outliers, and whiskers extending to the most extreme data points not considered to be outliers.

Accuracy of confidence results. The confidence for all approaches is low around the lower boundary of Θ_ϕ, and the MSE is high, shown in Fig. 4. This is consistent with the goal of the confidence calculation, where one would need to know the exact value of the system parameter θ if its value is near this edge, to be able to decide whether it falls in Θ_ϕ or not, and hence the calculation has a high sensitivity around this boundary This sensitivity increases as the amount of data increases, as seen by comparing the MSE for $\theta_1 = 0.4$ in Fig. 4a, where the trace length is 2, with Fig. 4b when the trace length has increased up to 10. To explore why this is the case, consider that to compute the confidence we integrate the posterior distribution over the feasible set $\Theta_\phi = [0.369, 0.75]$. The posterior distribution for $\theta_i = 0.369$ should have a peak centred at 0.369 and half of the



(a) Synthesised strategy (b) Random static strategy (c) No strategy

Fig. 5: Convergence of confidence outcomes to the ground truth for a run with 20 traces, each of length 10 over a simulated underlying system with both parameters (θ_1, θ_2) set to 0.7.

probability mass falling in the feasible set, leading to $\mathcal{C} = 0.5$. The height and width of the posterior distribution are determined by the amount and spread of data available and for a tall and thin distribution (encompassing a large amount of data), a small change in the position of the peak can move a large percentage of mass of the distribution in or out of the feasible set. This is prominent in Fig. 4b since our approach synthesises a strategy that would yield the highest information gain, i.e., the most useful data. However, as we move away from the edge, increased data effectively places probability mass away from the uncertain regions, thus reducing both variance and MSE. Neither of the other two alternatives has the ability to collect as much useful data and therefore variance is high even at the far ends of the parameter spectrum. The ability of our method to collect more useful data is also illustrated in the convergence graphs shown in Fig. 5, where synthesis approach converges to the ground truth quicker than both comparison strategies.

We conclude that our strategy synthesis does improve the accuracy of the confidence calculation, unless the parameter value falls close to the boundary of Θ_ϕ, and that away from this boundary the confidence converges to the ground truth and we are able to verify the property over S based on the data collected.

Robustness. We run our implementation with varying lengths of traces, where the total number of transitions in the data remains the same, and the results summarised in Fig. 4c show that our approach, on this case study, is relatively insensitive to this variation (compare Fig. 4a with Fig. 4b). Our method depends on the number of parameterised transitions we visit and so depends on the trace length being long enough to visit some parameterised transitions. This is in contrast to Statistical Model Checking techniques, where the accuracy of the approach depends on the trace length being great enough to satisfy the property, e.g., to reach some desired state. In both cases this will vary depending on the structure of the model.

8 Conclusions

In this paper, we have presented an approach for statistical verification of a fragment of unbounded-time PCTL properties on partially unknown systems, by automating the design of smart experiments that maximise the amount of useful data collected from the underlying system. We validate that our approach increases the accuracy of the confidence that the system satisfies the property, compared to selecting data randomly. We are able to achieve meaningful confidence outcomes with comparably limited amounts of available data.

We are pursuing extensions of this framework for much wider class of probabilistic models, in particular continuous time models, with a broad range of applications.

References

1. Araya-López, M., Buffet, O., Thomas, V., Charpillet, F.: Active learning of MDP models. In: EWRL. Lecture Notes in Computer Science, vol. 7188, pp. 42–53. Springer (2011)
2. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
3. Chen, Y., Nielsen, T.D.: Active learning of Markov decision processes for system verification. In: ICMLA (2). pp. 289–294. IEEE (2012)
4. D'Argenio, P., Legay, A., Sedwards, S., Traonouez, L.: Smart sampling for lightweight verification of Markov decision processes. STTT 17(4), 469–484 (2015)
5. Friedman, N., Singer, Y.: Efficient Bayesian parameter estimation in large discrete domains. In: NIPS. pp. 417–423. The MIT Press (1998)
6. Galassi, M., Davies, J., Theiler, J., Gough, B., Jungman, G.: GNU Scientific Library - Reference Manual, Third Edition, for GSL Version 1.12 (3. ed.). Network Theory Ltd (2009)
7. Gevers, M., Bombois, X., Hildebrand, R., Solari, G.: Optimal experiment design for open and closed-loop system identification. Comm. Inform. Syst. 11(3), 197–224 (2011)
8. Gretton, C., Price, D., Thiébaux, S.: Implementation and comparison of solution methods for decision processes with non-Markovian rewards. In: UAI. pp. 289–296. Morgan Kaufmann (2003)
9. Guan, P., Raginsky, M., Willett, R.M.: Online Markov decision processes with Kullback–Leibler control cost. IEEE Transactions on Automatic Control 59(6), 1423–1438 (2014)
10. Haesaert, S., Van den Hof, P.M.J., Abate, A.: Data-driven property verification of grey-box systems by Bayesian experiment design. In: 2015 American Control Conference (ACC). pp. 1800–1805 (July 2015)
11. Haesaert, S., Van den Hof, P.M.J., Abate, A.: Experiment design for formal verification via stochastic optimal control. In: ECC. pp. 427–432. IEEE (2016)
12. Henriques, D., Martins, J., Zuliani, P., Platzer, A., Clarke, E.M.: Statistical model checking for Markov decision processes. In: QEST. pp. 84–93. IEEE Computer Society (2012)
13. Hoffman, M.D., de Freitas, N., Doucet, A., Peters, J.: An expectation maximization algorithm for continuous Markov decision processes with arbitrary reward. In: AISTATS. JMLR Proceedings, vol. 5, pp. 232–239. JMLR.org (2009)

14. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: CAV. Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011)
15. Kwiatkowska, M.Z., Parker, D.: Automated verification and strategy synthesis for probabilistic systems. In: ATVA. Lecture Notes in Computer Science, vol. 8172, pp. 5–22. Springer (2013)
16. Legay, A., Delahaye, B., Bensalem, S.: Statistical model checking: An overview. In: RV. Lecture Notes in Computer Science, vol. 6418, pp. 122–135. Springer (2010)
17. Pasanisi, A., Fu, S., Bousquet, N.: Estimating discrete Markov models from various incomplete data schemes. Computational Statistics & Data Analysis 56(9), 2609–2625 (2012)
18. Peter Eichelsbacher, A.G.: Bayesian inference for Markov chains. Journal of Applied Probability 39(1), 91–99 (2002)
19. Polgreen, E., Wijesuriya, V.B., Haesaert, S., Abate, A.: Data-efficient Bayesian verification of parametric Markov chains. In: QEST. Lecture Notes in Computer Science, vol. 9826, pp. 35–51. Springer (2016)
20. Poupart, P., Vlassis, N.A., Hoey, J., Regan, K.: An analytic solution to discrete Bayesian reinforcement learning. In: ICML. ACM International Conference Proceeding Series, vol. 148, pp. 697–704. ACM (2006)
21. Quatmann, T., Dehnert, C., Jansen, N., Junges, S., Katoen, J.: Parameter synthesis for Markov models: Faster than ever. In: Automated Technology for Verification and Analysis - 14th International Symposium, ATVA 2016, Chiba, Japan, October 17-20, 2016, Proceedings. pp. 50–67 (2016)
22. Ross, S., Pineau, J., Chaib-draa, B., Kreitmann, P.: A Bayesian approach for learning and planning in partially observable Markov decision processes. Journal of Machine Learning Research 12, 1729–1770 (2011)
23. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: CAV. LNCS, vol. 3114, pp. 202–215. Springer (2004)
24. Younes, H.L.S.: Probabilistic verification for black-box systems. In: CAV. Lecture Notes in Computer Science, vol. 3576, pp. 253–265. Springer (2005)