Bayes-Adaptive Planning for Data-Efficient Verification of Uncertain Markov Decision Processes

Viraj Brian Wijesuriya(✉) and Alessandro Abate

Department of Computer Science, University of Oxford, Oxford, UK
viraj.wijesuriya@cs.ox.ac.uk

Abstract. This work concerns discrete-time parametric Markov decision processes. These models encompass the uncertainty in the transitions of partially unknown probabilistic systems with input actions, by parameterising some of the entries in the stochastic matrix. Given a property expressed as a PCTL formula, we pursue a data-based verification approach that capitalises on the partial knowledge of the model and on experimental data obtained from the underlying system: after finding the set of parameters corresponding to model instances that satisfy the property, we quantify from data a measure (a confidence) on whether the system satisfies the property. The contribution of this work is a novel Bayes-Adaptive planning algorithm, which synthesises finite-memory strategies from the model allowing Bayes-Optimal selection of actions. Actions are selected for collecting data, with the goal of increasing its information content that is pertinent to the property of interest: this active learning goal aims at increasing the confidence on whether or not the system satisfies the given property.

1 Introduction

Markov Decision Processes (MDPs) [23] have been successfully employed to solve many demanding decision making problems in complex environments. A fully-specified MDP can be leveraged to provide quantitative guarantees for correct behaviour of intricate engineering systems. Formal methods provide mathematically rigorous machinery to obtain such guarantees [3], but their applicability might fall short in the case of incomplete knowledge of the underlying system. Available knowledge of a partially unknown system can be encompassed by a parametric MDP (pMDP) [13], where a set of parameters is used to account for imperfect knowledge.

We are interested in performing data-efficient verification of partially unknown systems with input actions, which can be modelled using pMDPs. Input actions represent nondeterministic choices available for planning. We reason about system properties expressed in probabilistic computational tree logic (PCTL [15]). In this paper, we assume that full data can be gathered from the system to reason about these properties.

D. Parker and V. Wolf (Eds.): QEST 2019, LNCS 11785, pp. 91–108, 2019.
https://doi.org/10.1007/978-3-030-30281-8_6

Our verification approach is both model-based and data-driven [20,21]: on the one hand, we classify the pMDP model into those MDPs satisfying the given property and those that do not; on the other, we augment the knowledge about the pMDP with the information content derived from limited amount of data actively gathered from the system; we finally quantify a confidence on whether the system satisfies the property.

In this work, we perform active learning [1] by seeking optimal strategies to gather data from the system, with the objective of performing verification with the greatest degree of accuracy. The novelty of our contribution is to extend [21], where memoryless strategies were synthesised towards this task, which are highly sub-optimal for maximising the confidence estimate [23]. Here we tackle the requirement of memory dependency by formalising the verification problem as a model-based reinforcement learning task, which is cast into a framework that augments the pMDP with the information acquired through histories of interactions with the system: this results in a model formulation known as Bayes-Adaptive MDP (BAMDP) [8]. We also introduce a new algorithm called *Bayes-Adaptive Temporal Difference Search* for planning with BAMDPs. A reward function, related to the confidence estimate, is introduced over the BAMDP to set up an optimisation task. Optimal strategies help to steer the interaction with the underlying system to ultimately attain the most accurate confidence estimate.

1.1 Related Work

The parameter synthesis problem [13] aims at formulating a range of possible valuations for a set of parameters corresponding to the satisfaction of a property of interest. Recent works [6,13,24] perform synthesis utilising increasingly efficient techniques that scale well on larger state and parameter spaces. Parameter synthesis alone does not answer the question whether the underlying, partly known system satisfies the property. Instead, some information about the parameters also needs to be inferred from the system. We not only perform parameter synthesis but also *parameter inference*, which draws valuations for model parameters from measurement data from the system. When measurement data is readily available (i.e. need not gather from the underlying system as part of the planning process), approaches such as [2] proceed to find *parameter-independent* strategies where the expected reachability probability is optimised.

Depending entirely on data, [5,18] attempt to learn a completely deterministic representation of an MDP model from the system and to subsequently verify properties over the learnt model. Unlike our approach, they do not take into account prior knowledge available to the learner through the incomplete model at hand and the property given, leading to a single model fitting the underlying system. Characterising the transition model of an MDP, [1,25] aim at learning representations from sequences of data using Bayesian learning. The lack of information from a partial model and without the ability to exploit known relationships between parameters themselves, renders these approaches data-inefficient.

In this work we incorporate Bayesian inference [25] when *planning* over pMDPs. Simplicity and analytical behaviour of Bayesian inference has motivated its use herein. Statistical Model Checking (SMC) techniques [26] perform verification over fully-specified models by generating simulation data, or by gathering data from the underlying system if no model is specified. While we do not solve the same problem as SMC techniques do, our work strikes a balance between the two mentioned alternatives, allowing for a substantial reduction in data gathered owing to the knowledge encompassed in the partially known model (i.e., the pMDP). Notice that SMC techniques for MDPs solve nondeterminism via memoryless strategies [16], or employ history-dependent strategies from only a subset of possible strategies [17]. On the other hand, in this work, we have the ability to construct memory-efficient strategies that are focused on the specific objective of asserting whether or not the system satisfies the property.

Computing confidence estimates in formal verification is seen as a meaningful approach in the presence of uncertainty in models. Cognate to this work, [21] utilises ideas from *experiment design* to calculate memoryless schedulers for pMDPs to ultimately compute a confidence estimate for the satisfaction of properties. Similarly, [20] computes confidence estimates for parametric Markov chains (PMCs) and [12] performs data-driven verification over *linear, time-invariant dynamical systems* encompassing measurement uncertainty.

2 Rudiments

2.1 Markov Decision Processes

Let $\mathbb{P}(H)$ denote a probability measure over an *event* H while $\mathbb{E}[X]$ denote the expectation for any given random variable X. We use $P(\cdot)$ to denote a discrete probability distribution over a finite set S where $P\colon S \to [0,1]$ and $\sum_{s\in S} P(s) = 1$.

We consider a discrete-time Markov decision process (MDP), represented as a tuple $\mathcal{M} = (S, A, \mathcal{T}, \iota)$, where S is the finite set of states, A is the finite set of actions. $\mathcal{T}\colon S \times A \times S \to [0,1]$ is a transition probability function such that $\forall s \in S$ and $\forall a \in A$: $\sum_{s'\in S} \mathcal{T}(s,a,s') \in \{0,1\}$. $\iota \subseteq S$ denotes the set of initial states. Any action that belongs to the set $A(s) = \{a \in A \mid \sum_{s'\in S} \mathcal{T}(s,a,s') = 1\}$ is said to be *enabled* at state s.

Consider an underlying data-generating system that allows to observe and collect finite traces of data in the form of sequences of visited states and chosen actions. We take into account the case where an MDP model that exactly represents the system is unknown but is assumed belonging to a class of uncertain models comprised in a parametric Markov decision process (pMDP).

A pMDP is a tuple $\mathcal{M}_p = (S, A, \mathcal{T}_p, \iota, \Theta)$, where the previous definition of an MDP is lifted, ceteris paribus, to include a transition function $\mathcal{T}_p$ with a target set specified using parameters found in an n-dimensional vector θ. All possible *valuations* of θ are held in $\Theta \subseteq [0,1]^n$, with $n \in \mathbb{N}_{>0}$. For any $\theta \in \Theta$, we enforce that $\forall s \in S$ and $\forall a \in A(s)$: $\sum_{s'\in S} \mathcal{T}_p(s,a,s') = 1$. Hence, each valuation of θ induces a single MDP $\mathcal{M}(\theta)$ with a transition function that can

be represented using a *stochastic matrix*. Whilst the probabilities of all non-parameterised transitions of $\mathcal{M}_p$ are assumed to be known, we allow (unknown) probabilities of parameterised transitions to be linearly related, as in [21].

2.2 Strategies

A *strategy* (a.k.a. *policy* or *scheduler*) designates an agent's behaviour within its environment. For an MDP, a strategy is a distribution over actions at a particular state. A *deterministic* strategy selects a single action at a particular state and a *deterministic memoryless* (a.k.a. *stationary*) strategy $\pi\colon S \to A$ where $\forall s \in S$: $\pi(s) \in A(s)$, always selects the same action per state, regardless of any available *memory* of previously selected actions and/or visited states, hence allowing for a time-independent choice.

In this work, we are compelled to introduce the notion of memory (a.k.a. *history*) with respect to strategies, since memoryless strategies fail to be adequate with optimality in the choice of actions [2,23]. We call a memory $\mathbf{m}$, a sequence of states and actions, namely $\mathbf{m} = s_0 a_0 s_1 a_1 s_2 a_2 \ldots$, where $s_i \in S$, $a_i \in A(s_i)$ and $\mathcal{T}(s_i, a_i, s_{i+1}) > 0$. A memory $\mathbf{m}_t = s_0 a_0 s_1 a_1 s_2 a_2 \ldots a_{t-1} s_t$ is finite if it covers a finite-time horizon $t \in \mathbb{N}_{>0}$. Let $\mathscr{M}$ represent the set of possible finite memories. A *deterministic finite-memory* strategy $\hat{\pi}\colon S \times \mathbf{M} \to A$ has finitely many modes $\mathbf{M} \subseteq \mathscr{M}$, such that a single action a_t is chosen at time t, namely $a_t = \hat{\pi}(s_t, \mathbf{m}_t)$, $\forall t > 0$ where $\mathbf{m}_t \in \mathbf{M}$ and $a_t \in A(s_t)$. Obviously, s_t is the last state in memory $\mathbf{m}_t$: the redundant emphasis on pairs $(s_t, \mathbf{m}_t)$ is a notational convenience inherited from literature that will be further justified in Sect. 4.1.

2.3 Bayesian Reinforcement Learning (Bayesian RL)

Model-based Bayesian RL for pMDPs relies on an explicit model, which is learnt assuming priors over model parameters and by updating a posterior distribution using Bayesian inference [25] as more data is gathered from the underlying system. Subsequently (and possibly iteratively), the MDP with parameters sampled from the current posterior is employed to find an optimal policy that maximises the long-term expected reward.

We consider a Bayes-Adaptive RL formulation [8], with a model that allows to encode memory as part of the state space. A Bayes-Adaptive MDP is a tuple $\mathcal{M}_{ba} = (\hat{S}, A, \hat{\mathcal{T}}, \hat{\iota}, \mathcal{R})$, where $\hat{S} = S \times \mathbf{M}$ is the state space encompassing memory, A is as defined in Sect. 2.1 for an MDP, the transition function $\hat{\mathcal{T}}(s, \mathbf{m}, a, s', \mathbf{m}')$ designates transitions between *belief states* $(s, \mathbf{m}) \in \hat{S}$ and $(s', \mathbf{m}') \in \hat{S}$ after choosing an action a. Further, $\hat{\iota} \subseteq \hat{S}$ where $(s, \mathbf{m}) \in \hat{\iota}$ if $s \in \iota$. $\mathcal{R}\colon \hat{S} \times A \times \hat{S} \to \mathbb{R}$ is the newly introduced *transition reward*. The transition (s, a, s') plus the *information state* $\mathbf{m}$ affect the next information state $\mathbf{m}'$, thereby preserving the *Markov property* among transitions between belief states.

When an action a is selected in a belief state (s, $\mathbf{m}$), a transition occurs to the successive belief state $(s', \mathbf{m}') \sim \hat{\mathcal{T}}(s, \mathbf{m}, a, \cdot, \cdot)$ and a transition reward r is received. With a slight abuse of notation, the function $\mathcal{R}(s, \mathbf{m}, a) = \mathbb{E}[r \mid s_t =$

$s, \mathbf{m}_t = \mathbf{m}, a_t = a]$ shall denote the expected reward for the pair: belief state $(s, \mathbf{m})$ and action a.

Given a finite-time horizon $T \in \mathbb{N}_{>0}$ and strategy $\hat{\pi}$, the *action-value* function $Q^{\hat{\pi}}(s, \mathbf{m}, a) = \mathbb{E}_{\hat{\pi}}[\Sigma_{j=t}^{T} r_j \mid s_t = s, \mathbf{m}_t = \mathbf{m}, a_t = a]$ is the expected cumulative transition reward up to the horizon T after action a is chosen over belief state $(s, \mathbf{m})$ and thereafter following strategy $\hat{\pi}$. Solving a BAMDP in our context boils down to finding a finite-memory, deterministic strategy $\hat{\pi}^*$ that maps belief states from the augmented state space $\hat{S}$ to actions in A and which maximises an expected cumulative transition reward over a given finite horizon T.

2.4 PCTL Properties

We consider *non-nested* properties expressed in a fragment of probabilistic computational tree logic (PCTL) [15]. For a PCTL formula ϕ interpreted over states of a given MDP $\mathcal{M}$, a formula φ interpreted over the paths [3], $\bowtie \in \{<, \leq, \geq, >\}$ and $b \in [0, 1]$, the probabilistic operator $\phi = \mathbf{P}_{\bowtie b}(\varphi)$ in a state $s \in S$ expresses the probability for paths starting at s that satisfy φ meet the bounds given by $\bowtie b$. We consider path formulae both bounded: $\varphi = \phi_i \, \mathcal{U}^{\leq k} \, \phi_j$ (with a finite-time bound $k \in \mathbb{N}_{>0}$) and unbounded time: $\varphi = \phi_i \, \mathcal{U} \, \phi_j$. Denote by $\mathbb{P}^{\pi}(\varphi \mid s)$ the probability of satisfying φ along the paths of MDP $\mathcal{M}$ that start from $s \in S$ and follow a given strategy π. The satisfaction of formula $\mathbf{P}_{\bowtie b}(\phi_i \, \mathcal{U}^{\leq k} \, \phi_j)$ over $\mathcal{M}$ is thus given by:

$$\mathcal{M} \models \mathbf{P}_{\bowtie b}(\phi_i \, \mathcal{U}^{\leq k} \, \phi_j) \iff \forall s \in \iota \colon \underset{\pi \in Str_{\mathcal{M}}}{\mathfrak{A}} \mathbb{P}^{\pi}(\phi_i \, \mathcal{U}^{\leq k} \, \phi_j \mid s) \bowtie b,$$

where $\mathbb{P}^{\pi}$ is the measure over the events corresponding to the formula ϕ for the MDP under strategy π and $Str_{\mathcal{M}}$ is the set of all strategies of $\mathcal{M}$ and $\mathfrak{A} \in \{\inf, \sup\}$ with following choices for $\bowtie$: inf if $\geq$ or $>$ and sup if $<$ or $\leq$. The satisfaction of $\mathbf{P}_{\bowtie b}(\phi_i \, \mathcal{U} \, \phi_j)$ over $\mathcal{M}$ can be derived similarly. Considering a pMDP $\mathcal{M}_p$, we let Θ_ϕ denote the set of valuations of θ for which the formula ϕ is satisfied: $\theta \in \Theta_\phi \Leftrightarrow \mathcal{M}(\theta) \models \phi$ and we call Θ_ϕ the *feasible set*.

3 Verification and Learning

We present here our integrated verification and learning approach [20].

3.1 Parameter Synthesis

The aim of parameter synthesis is to classify induced MDP models of the corresponding pMDP, between those that satisfy a given property ϕ and those that do not. This is achieved by producing an output that maps regions corresponding to parameter valuations to truth values [13]. Regions that map to "true" are considered belonging to the feasible set Θ_ϕ, while those that maps to "false" are guaranteed to not contain valuations that satisfy the property ϕ. This step handles actions pessimistically, namely models are considered to verify property ϕ

regardless of the action selection. This is because we plan to fully utilise actions for learning at a later step. Evidently, a different trade-off on actions could be struck, which we delegate to future work. We employ the probabilistic model checker Storm [7] to perform synthesis because it supports the PCTL properties of interest.

3.2 Bayesian Inference

Bayesian inference allows to determine a posterior probability distribution over the likely values of the parameters $\theta \in \Theta$, based on data gathered from the underlying system and to update this probability distribution as more data is collected [25]. It is also possible to incorporate any subjective beliefs about the parameters using a prior distribution $P(\theta)$.

Denote by $\mathscr{D}$ the set of *finite traces* gathered from the system so far, comprising a count $C^a_{s,s'}$ of how many times a particular transition (s, a, s') has been observed. We limit parameterisation of pMDP transitions to two cases of interest: (a) each transition of the pMDP is parameterised either with a single parameter (e.g. θ_i or $1-\theta_i$) or with a constant $k \in (0, 1]$; (b) the pMDP includes transitions whose transition probabilities are expressed as affine functions of the form $g^{s'}_{s,a}(\theta) = k_0 + k_1\theta_1 + \ldots + k_n\theta_n$. For any instance of (b), [21] suggests two transformations that produce a pMDP that contains only transition probabilities of the form given in (a). Therefore, without loss of generality and for the purpose of succinctness, we assume pMDPs with parameterisation corresponding to the form in (a) herein. For transitions having identical parameterisations, their transition counts can be grouped together using *parameter tying* [22].

Denote by $C(\theta_i)$, the number of times transitions parameterised by θ_i have been observed in $\mathscr{D}$: $C(\theta_i) = \sum C^a_{s,s'}$ for $\mathcal{T}_p(s, a, s') = \theta_i$. Similarly, we define $C'(\theta_i)$ to count transitions parameterised by $1 - \theta_i$, i.e., transitions not parameterised by θ_i given that there exists a transition parameterised by θ_i under the same action $a \in A(s)$. We collect both $C(\theta_i)$ and $C'(\theta_i)$ in $\bar{C}(\theta_i)$ for brevity.

Assuming a prior distribution $P(\theta_i)$ over each component parameter $\theta_i \in \theta$, the posterior distribution over θ_i can be expressed using Bayes' rule as:

$$P(\theta_i \mid \mathscr{D}) \propto P(\theta_i)\theta_i^{C(\theta_i)}(1-\theta_i)^{C'(\theta_i)}.$$

The counts $C^a_{s,s'}$ follow a multinomial distribution [25]. Selecting a Beta distribution: $\text{Beta}(\theta_i; (\alpha_{\theta_i}, \beta_{\theta_i}))$ as a conjugate prior, the posterior distribution has a closed-form expression, allowing it to be updated by adding respective parameter counts to the hyper-parameter pair $(\alpha_{\theta_i}, \beta_{\theta_i})$ [25]:

$$P(\theta_i \mid \mathscr{D}) = \text{Beta}(\theta_i; \bar{C}(\theta_i) + (\alpha_{\theta_i}, \beta_{\theta_i})).$$

Note that hyper-parameters α_{θ_i} and β_{θ_i} are the parameters of the (Beta) prior distribution over θ_i. We denote by $U_i(\mathbf{m})$ the update on hyper-parameter counts $\bar{C}(\theta_i)+(\alpha_{\theta_i}, \beta_{\theta_i})$, corresponding to an information state $\mathbf{m}$. Marginals $P(\theta_j \mid \mathscr{D})$ can be combined to form the posterior for the entire vector $\theta \in \Theta$ under the

assumption that each component parameter θ_j is independent over those independent state-action pairs in the pMDP. Thus, the posterior $P(\theta \mid \mathscr{D})$ is given by: $P(\theta \mid \mathscr{D}) = \prod_{\theta_j \in \theta} P(\theta_j \mid \mathscr{D})$. Whenever an analytical update is impossible, we can resort to *sampling* techniques to obtain posterior realisations [21].

3.3 Confidence Computation

Given a specification ϕ and the posterior distribution $P(\theta \mid \mathscr{D})$ for $\theta \in \Theta$ obtained from a system of interest S, the confidence on whether $\mathsf{S} \models \phi$, can be quantified according to [20]:

$$\mathscr{C} = \mathbb{P}(\mathsf{S} \models \phi \mid \mathscr{D}) = \int_{\Theta_\phi} P(\theta \mid \mathscr{D}) d\theta = \int_{\Theta_\phi} \prod_{\theta_j \in \theta} P(\theta_j \mid \mathscr{D}) d\theta. \tag{1}$$

This quantity in general can be computed using Monte Carlo integration [20].

3.4 Overview of the Approach

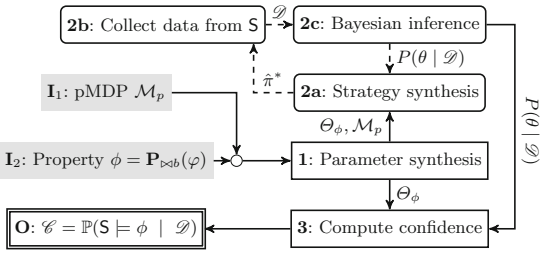

Fig. 1. Verification and Learning. $\mathbf{I}_1$, $\mathbf{I}_2$ are inputs and output $\mathbf{O}$ is the confidence estimate $\mathscr{C}$.

The different phases of our approach are shown in Fig. 1. We assume that the available parametric model $\mathcal{M}_p$ best represents the underlying system together with its uncertain dynamics. We first perform parameter synthesis over the given parametric model to find the feasible set of parameter values Θ_ϕ that satisfy the specification at hand. We then collect data from the system and employ Bayesian inference to update the posterior distribution over the likely values of the parameters with respect to the gathered data. Finally, we output a quantification of the confidence that the underlying system satisfies the specification over the data gathered so far.

In this work, the feasible set and the current distribution over the parameter space are propagated through a BAMDP model and used to plan (viz. synthesise a strategy) for gathering valuable data from the underlying system. We synthesise strategies to sequentially collect data to optimise the measure in Eq. (1) which will be further elaborated in Sect. 4.2.

4 Active Learning

We introduce a new technique for model-based Bayesian RL to synthesise a finite-memory strategy to further explore the system from data.

4.1 Bayes-Adaptive Model

In order to collect maximally useful data from the underlying system, we take into account the importance of (both past and expected) information to decrease the uncertainty associated with model parameters with respect to property satisfaction. Our confidence quantification (cf. Eq. (1)) is a proxy for this uncertainty: if the property is satisfied over the underlying system, one would expect the confidence to be as high as possible and, conversely, as low possibly if the property is not satisfied (essentially, in either case, one ideally wants to be away from the value 0.5).

We lift the BAMDP model described in Sect. 2 to support uncertainty described by parameterised transitions assuming a Beta-Binomial representation for the posterior distribution (as in Sect. 3.2). Using this uncertainty representation, we encompass the information state $\mathbf{m}$ by a joint probability distribution $\mathfrak{i}_{\mathbf{m}}$ over the hyper-parameters, which are collectively denoted by the pair $(\boldsymbol{\alpha}, \boldsymbol{\beta}) = \{U_j(\mathbf{m}) \mid \forall \theta_j \in \theta\}$, namely, $\boldsymbol{\alpha} = \langle C(\theta_1) + \alpha_{\theta_1}, C(\theta_2) + \alpha_{\theta_2}, \ldots, C(\theta_n) + \alpha_{\theta_n} \rangle$ and $\boldsymbol{\beta} = \langle C'(\theta_1) + \beta_{\theta_1}, C'(\theta_2) + \beta_{\theta_2}, \ldots, C'(\theta_n) + \beta_{\theta_n} \rangle$ where $n = |\theta|$. The hyper-parameters for θ_j are thus denoted by $\mathfrak{i}_{\mathbf{m},j}$. The distribution $\mathfrak{i}_{\mathbf{m}}$ acts as a statistic for $\mathbf{m}$ that summarises all information accumulated so far. We furthermore adapt the pair $(s, \mathbf{m})$ to $(s, \mathfrak{i}_{\mathbf{m}})$ as a belief state of the BAMDP, essentially lifting memories $\mathbf{M}$ to the hyper-parameters $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ of the (Beta) posterior distributions, $\widehat{\mathbf{M}}$. This lifting preserves the Markovian property of transition function $\hat{\mathcal{T}}$. For the remainder of the paper, we employ $\mathfrak{i}_{\mathbf{m}}$ and $\widehat{\mathbf{M}}$ to reason about intended concepts over $\mathbf{m}$ and $\mathbf{M}$, respectively.

We formulate BAMDPs in this work to transform the uncertainty in parameters $\theta \in \Theta$ of a pMDP into certainty about belief states of a BAMDP. A prior $P(\mathcal{T}_p)$ on the transition function $\mathcal{T}_p$ of a pMDP corresponds to a prior distribution $P(\theta)$ over parameters θ. Accordingly, we can define a posterior belief $b(\mathcal{T}_p)$ over the transition function given data $\mathscr{D}$, so that $b(\mathcal{T}_p) = P(\theta \mid \mathscr{D})$. For a memory $\mathbf{m}$, this belief can be quantified as $b(\mathcal{T}_p) = P(\mathcal{T}_p \mid \mathfrak{i}_{\mathbf{m}}) \propto P(\mathfrak{i}_{\mathbf{m}} \mid \mathcal{T}_p) P(\mathfrak{i}_{\mathbf{m}})$. The transition dynamics for the BAMDP can be formulated as:

$$\hat{\mathcal{T}}(s, \mathfrak{i}_{\mathbf{m}}, a, s', \mathfrak{i}_{\mathbf{m}'}) = \int_{\mathcal{T}_p} \mathcal{T}_p(s, a, s') P(\mathcal{T}_p \mid \mathfrak{i}_{\mathbf{m}}) \mathrm{d}\mathcal{T}_p, \tag{2}$$

where $\mathbf{m}'$ is the updated memory after the transition (s, a, s') is witnessed. The RHS of Eq. 2 is the expectation of $\mathcal{T}_p(s, a, s')$, which corresponds to the expectation of the posterior $P(\theta \mid \mathscr{D})$, hence $\hat{\mathcal{T}}$ can be expressed as:

$$\hat{\mathcal{T}}(s, \mathfrak{i}_{\mathbf{m}}, a, s', \mathfrak{i}_{\mathbf{m}'}) = \begin{cases} \mathbb{1}_{\mathfrak{i}_{\mathbf{m}as'}}(\mathfrak{i}_{\mathbf{m}'}) \frac{\alpha_{\theta_j}}{\alpha_{\theta_j} + \beta_{\theta_j}} & \text{if } \mathcal{T}_p(s, a, s') = \theta_j, \\ \mathbb{1}_{\mathfrak{i}_{\mathbf{m}as'}}(\mathfrak{i}_{\mathbf{m}'}) \frac{\beta_{\theta_j}}{\alpha_{\theta_j} + \beta_{\theta_j}} & \text{if } \mathcal{T}_p(s, a, s') \neq \theta_j \text{ and} \\ & \exists s_k \in S \colon \mathcal{T}_p(s, a, s_k) = \theta_j, \\ \mathcal{T}_p(s, a, s') & \text{otherwise,} \end{cases} \tag{3}$$

where $\mathbb{1}_b(a)$ is 1 if a equals b, else 0.

4.2 Synthesis of Bayes-Adaptive Strategies

The reward function $\mathcal{R}$ is used in the BAMDP to designate expected confidence updates resulting from transitions and thereby to guide the learning process. We work with finite-horizon problems. Positive rewards defined at the horizon (and zero elsewhere) require the learner to consider complete trajectories spanning the horizon in order to accumulate non-zero rewards. This can be computationally expensive should the horizon be large. Therefore, we focus on obtaining immediate rewards at each step of the learning process.

We define belief-dependent rewards [19] based on the difference between the confidence estimate at the given time step and that at the successive time step. As a learning process is designed to maximise rewards [28], here it is focussed on maximising the information content from the system's data to compute the most accurate confidence estimate possible. In order to achieve this, we need to synthesise strategies that maximise the deviation between *future confidence* and the base case $\mathscr{K} = 0.5$.

Denote by $\mathscr{C}_t$, the confidence estimate at current time step t, where we have set $\mathscr{C}_0 = \mathscr{K}$. After selecting action a_t over the belief state $(s_t, \mathbf{i}_{\mathbf{m}_t})$, the next-step confidence $\mathscr{C}_{t+1}$ can be used to define an immediate confidence gain r_{t+1} as:

$$r_{t+1} = |\mathscr{K} - \mathscr{C}_{t+1}| - |\mathscr{K} - \mathscr{C}_t|.$$

The reward function $\mathcal{R}(s, \mathbf{i}_{\mathbf{m}}, a)$ is thus defined as $\mathcal{R}(s, \mathbf{i}_{\mathbf{m}}, a) = \mathbb{E}[r_{t+1} \mid s_t = s, \mathbf{i}_{\mathbf{m}_t} = \mathbf{i}_{\mathbf{m}}, a_t = a]$, where clearly $\mathcal{R}(s, \mathbf{i}_{\mathbf{m}}, a) = 0$ if there is no associated parameterised transition. An interesting observation about $\mathcal{R}$ is that corresponding rewards might converge to zero in the limit, i.e., go to zero as the agent is left with nothing more to learn.

With respect to a prior distribution $P(\theta)$, the *Bayes-Optimal* policy $\hat{\pi}^*$ that maximises the expected cumulative reward over a finite horizon T is given by

$$\hat{\pi}^*(s, \mathbf{i}_{\mathbf{m}}) = \arg\max_{a \in A(s)} Q^*(s, \mathbf{i}_{\mathbf{m}}, a),$$

where the corresponding Bayes-Optimal *action-value* function is given by

$$Q^*(s, \mathbf{i}_{\mathbf{m}}, a) = \sup_{\hat{\pi}} \mathbb{E}_{\hat{\pi}}[\Sigma_{j=t}^{T} r_j \mid s_t = s, \mathbf{i}_{\mathbf{m}_t} = \mathbf{i}_{\mathbf{m}}, a_t = a, \theta \sim P(\theta)].$$

Note that the Bayes-Optimal policy $\hat{\pi}^*(s, \mathbf{i}_{\mathbf{m}})$ depends on prior beliefs and is consistent with the way in which $\mathscr{C}_{t+1}$ is calculated. According to Eq. (1), this value corresponds to the *expected values* of the parameter counts after transitioning from state s_t by selection of action a_t. Starting from the expected values of transition counts, as described in Sect. 3.2, we can collect the expected parameter counts in $\bar{C}(\theta)$. The expected transition counts correspond to the Binomial distribution over the transitions under a chosen action. The expected values of the transition probabilities can be calculated via the expected values of the parameters. For instance, the expected value of a given parameter θ_j is $\mathbb{E}[\theta_j] = \dfrac{\alpha_{\theta_j}}{\alpha_{\theta_j} + \beta_{\theta_j}}$. The expected transition probability for the transition (s, a, s') is hence $g^{s'}_{s,a}(\mathbb{E}[\theta_j])$.

4.3 Bayes-Adaptive Temporal Difference Search

The learning algorithm we introduce is based on learning from simulated episodes of experiences gathered from the BAMDP. However, a BAMDP can be sizeable, even for a corresponding simple concrete MDP [8]. The information space grows exponentially with the number of state-action pairs in the concrete MDP and the horizon T of exploration. In our setting, T directly relates to the length of the traces drawn from the system and T can be chosen arbitrarily but needs to be large enough to witness several state transitions. However, when the PCTL property imposes a finite-time bound k on satisfaction, T should not exceed k. Even though there exist exact solutions to BAMDPs, for instance, via dynamic programming using *Gittins indices* [11], in most practical cases they are intractable. Let us recall that we denote by $Q(s, \mathrm{i}_{\mathbf{m}}, a)$ the expected cumulative transition reward, when action a is selected at belief state $(s, \mathbf{m})$. It is in practice not possible to store all values of $Q(s, \mathrm{i}_{\mathbf{m}}, a)$ in memory and learning exact values might be too slow. One way of reducing these computational burdens is to observe that distinct memories may yield the same (or a similar) belief [8], hence generalisation of Q values among related paths can be helpful. [10] proposes a Monte Carlo simulation algorithm to estimate Q values with a function approximator, which allows generalisation between states, actions and beliefs. However, such methods require to evaluate the final step of the simulation to update all corresponding Q values.

In this work, we follow an approach based on temporal difference (TD) learning [28], which can update the estimate of the Q value after every step of a simulation. This is helpful when the time horizon is very long (or non-terminating). Furthermore, the ability to learn step by step helps in estimating Q values with low variance and to plan via subsequent decisions that can be correlated in time. Temporal Difference Search (TD Search) is a simulation-based algorithm that employs value function approximation. Initially used for planning in Computer Go [27], we extend TD search to the context of Bayes-Adaptive models.

A new Bayes-Adaptive temporal difference search algorithm is outlined in Algorithm 1. It gathers episodes of simulated trajectories starting from the current belief state $(s_t, \mathrm{i}_{\mathbf{m}_t})$ according to an ϵ–*greedy* strategy. An ϵ–*greedy* strategy selects an action that maximises the local Q value with a probability equal to $1 - \epsilon$ or outputs a random action with probability ϵ. Rather than exploring the whole BAMDP, our algorithm commits to solving a sub-BAMDP that starts at the current belief state and spans a given time horizon T. Once Algorithm 1 synthesises a strategy (based on the current posterior $P(\theta \mid \mathscr{D})$), we roll it out up to the designated time horizon T and collect data (in the form of traces of length T) from the underlying system. We then update the BAMDP model via Bayesian inference using the collected data: the current posterior distributions of each parameter $\theta_i \in \theta$ is updated using the new data. Next, we synthesise a new strategy to further gather data. We continue in this fashion until we have gathered an arbitrary allowed number of traces from the system. We then output the eventual confidence estimate that asserts whether the system satisfies the property (cf. Fig. 1).

Value Function Approximation. In order to approximate the value function, we lift the action-value function with a weight matrix β of learnable parameters: $Q(s, \mathbf{i_m}, a; \beta)$. The goal of the learning process is to then find β that minimises the mean-squared error between approximate and true Q functions: $\mathbb{E}[(Q(s, \mathbf{i_m}, a; \beta) - Q(s, \mathbf{i_m}, a))^2]$.

Algorithm 1. Bayes-Adaptive Temporal Difference Search

Inputs:
  s_t, $\mathbf{i}_{\mathbf{m}_t}$
Initialize:
  $\beta \leftarrow 0$, $\rho \leftarrow 0$
procedure SEARCH(s_t, $\mathbf{i}_{\mathbf{m}_t}$)
  while time remaining **do** ▷ Start episode
    $s \leftarrow s_t$, $\mathbf{i_m} \leftarrow \mathbf{i}_{\mathbf{m}_t}$, $\bar{t} \leftarrow t$
    $a \leftarrow \hat{\pi}_{\epsilon\text{–greedy}}(s, \mathbf{i_m}; Q)$
    $\xi \leftarrow 0$, $\xi' \leftarrow 0$
    while $\bar{t} < T$ **do**
      $s' \sim \hat{\mathcal{T}}(s, \mathbf{i_m}, a, \cdot, \mathbf{i_{ma}}\cdot)$
      $R \leftarrow \mathcal{R}(s, \mathbf{i_m}, a)$
      $a' \leftarrow \hat{\pi}_{\epsilon\text{–greedy}}(s', \mathbf{i}_{\mathbf{m}as'}; Q)$
      $\delta \leftarrow R + Q(s', \mathbf{i}_{\mathbf{m}as'}, a'; \beta) - Q(s, \mathbf{i_m}, a; \beta)$
      $\xi = \lambda\xi + \mathbf{y}(\mathbf{i_m}) \otimes \mathbf{x}(s, a)$
      $\xi' = \lambda\xi + \mathbf{y}(\mathbf{i}_{\mathbf{m}as'}) \otimes \mathbf{x}(s', a')$
      $\beta \leftarrow \beta + \alpha\delta\xi - \alpha\xi'(\xi^\top\rho)$
      $\rho \leftarrow \rho + \omega(\delta J - \xi^\top\rho)\xi$
      $s \leftarrow s'$, $a \leftarrow a'$, $\bar{t} \leftarrow \bar{t} + 1$
    end while
  end while
  return $\arg\max_{a_t} Q(s_t, \mathbf{i}_{\mathbf{m}_t}, a_t; \beta)$
end procedure

We use the backward view of SARSA(λ) [28], with ϵ–*greedy* strategy improvement, to help learn parameters β at each step of simulation sequences. The parameter λ designates up to how far in time should *bootstrapping* occur (bootstrapping refers to updating an estimated value with one or more estimated values of the same kind). To implement the backward view, it is required to maintain an eligibility trace $\xi^a_{s,\mathbf{i_m}}$ [1] for each tuple $(s, \mathbf{i_m}, a)$. An *eligibility trace* [28] temporarily remembers the occurrence of an activity, for instance, visiting a state and choosing an action. The trace signals the learning process that the credit associated to the activity is eligible for change. The trace assigns credit to eligible activities based on combing assignments from two common heuristics: frequency heuristic (where credit is assigned to most frequent activities) and recency heuristic (where credits is assigned to most recent activities).

A BAMDP entails a convex action-value function [8] as a function of the information state: this function becomes piecewise linear if the horizon is finite and if the state-action space is discrete. Therefore, linear value function approximation is appropriate to represent the true convex action-value function for a particular state-action pair. Since the transition rewards we receive correspond to a gain in confidence, the values are bounded within $[0, 1]$. As such, the introduced function approximation is truncated using the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ [27].

Feature Representation. The quality of the value function approximation $Q(s, \mathbf{i_m}, a; \beta)$ greatly depends on employed features: we use the feature triple $(s, \mathbf{i_m}, a)$. A good approximation procedure should generalise well for those memories that lead to similar information states (or beliefs). The feature representation should facilitate likewise representations for such memories. We propose the following representation for $Q(s, a, \mathbf{i_m}; \beta)$ [10]:

$$Q(s, \mathbf{i_m}, a; \beta) = \mathbf{y}(\mathbf{i_m})^\top \beta \mathbf{x}(s, a).$$

[1] Note that in Algorithm 1, we actually maintain an eligibility trace matrix ξ.

This form encodes the feature triple into the Q approximation, with vector $\mathbf{x}(s,a)$ concerning state-action pairs and vector $\mathbf{y}(\mathfrak{i}_{\mathbf{m}})$ representing information states.

$\mathbf{x}(s,a)$ indicates which state-action pair is currently involved. Therefore, for a particular pair, this representation of Q is linear as a function of $\mathfrak{i}_{\mathbf{m}}$ which approximates the true convex action-value function. State-action pairs with similar features will be considered to be similar. We associate $\mathbf{x}(s,a)$ with binary features by assigning it a column vector of size $\mathcal{Z}$, with value *one* assigned to the location of the element corresponding to (s,a) and to any entry corresponding to parameter similar state-action pairs of (s,a), while other entries are assigned the value zero[2]. By parameter similar state-action pairs, we mean those pairs with outgoing transitions having identical parameterisation.

The construction of the vector $\mathbf{y}(\mathfrak{i}_{\mathbf{m}})$ requires representing beliefs in a coordinate vector form. However, as beliefs are not finite-dimensional objects, a finite-dimensional approximation is therefore required. [10] proposes a sampling mechanism based on a *sequential importance sampling particle filter*. We construct $\mathbf{y}\colon \widehat{\mathbf{M}} \to \mathbb{R}^{\mathcal{Z}}$ as follows, assuming that $\mathcal{Z}$ is the degree of the finite-dimensional approximation. We initialise $\mathbf{y}(\mathfrak{i}_{\mathbf{m}})$, a column vector with $\mathcal{Z}$ elements, to $\frac{1}{\mathcal{Z}}$ at the beginning of each episode of Algorithm 1. We then modify $\mathbf{y}(\mathfrak{i}_{\mathbf{m}})$ by updating each entry j at the current step, using the probability given by $\hat{T}(s,\mathfrak{i}_{\mathbf{m}},a,s',\mathfrak{i}_{\mathbf{mas'}})$, as:

$$\mathbf{y}_j(\mathfrak{i}_{\mathbf{mas'}}) = \mathbf{y}_j(\mathfrak{i}_{\mathbf{m}})\hat{T}(s,\mathfrak{i}_{\mathbf{m}},a,s',\mathfrak{i}_{\mathbf{mas'}}).$$

Notice how this scheme allows different memories leading to same belief to be mapped as identical representations, i.e., $\mathbf{y}(\mathfrak{i}_{\mathbf{m}'}) = \mathbf{y}(\mathfrak{i}_{\mathbf{m}})$ if $b(\mathbf{m}') = b(\mathbf{m})$. With this construction, it is not required to explicitly update the information of belief states that are not directly traversed in the simulation, since these updates are implicitly reflected in the finite-dimensional representation. The two updates (cf. Algorithm 1, lines: 15 and 16) on eligibility traces capture the joint effect of the introduced feature vectors $\mathbf{x}$ and $\mathbf{y}$ (in the algorithm, $\otimes$ denotes the standard outer product).

Feature vectors $\mathbf{x}(s,a)$ and $\mathbf{y}(\mathfrak{i}_{\mathbf{m}})$ effectively generalise from states, actions, and memories already seen to those unseen. As such, the rolled-out simulations will achieve the generalisation without the need to traverse all possible states of the BAMDP, making the algorithm much more efficient.

We run Algorithm 1 episodically (note that an episode starts from line 6 and ends in line 21) to learn β using simulated traces from the BAMDP model. We roll-out these simulations (cf. from line 10 to 20) up to the horizon T. The propagation of knowledge from one step of the algorithm to the other is fundamental to learning good representations from past experiences. When rolling-out simulations, one can use $\mathbf{y}(\mathfrak{i}_{\mathbf{m}})$ from previous time step $\tilde{t}-1$ for learning in the current step $\tilde{t}$ but $\mathbf{y}(\mathfrak{i}_{\mathbf{m}})$ may degenerate, e.g. leaving one of its entries $\mathbf{y}_j(\mathfrak{i}_{\mathbf{m}}) = 1$ and rest being zero. Given a threshold Y, we simply re-initialise all entries if $\frac{1}{\sum_j \mathbf{y}_j(\mathfrak{i}_{\mathbf{m}_{\tilde{t}}})^2} < Y$ or else, we reuse $\mathbf{y}$ from the previous β update.

[2] This scheme is sometimes called *one-hot encoding*.

Convergence. In the context of our action-value function approximation, convergence means that entries of β reach a fixed point. Linear SARSA (λ) (which is the underlying learning algorithm used in TD search) is sensitive to initial values of β and does not always converge in view of chattering [9]. Convergence guarantees for standard SARSA(λ), requiring a *greedy in the limit with infinite exploration* (GLIE) sequence of strategies and *Robbins-Monro conditions* for the learning rate α [28], are not in general enough to guarantee convergence for linear SARSA (λ), since it is not a true gradient-descent method [29]. Policy gradient methods [28] on the other hand can be used in place if one desires guaranteed convergence. This motivates the use of stochastic gradient descent algorithms. Since we roll-out a complete strategy before the next stage of data gathering, we are essentially performing open loop control together with the current beliefs we possess. Therefore, our planning stage is based on *off-policy learning*: training on outcomes from an ϵ-*greedy* strategy in order to learn the value of another. *Linear TD with gradient correction* (TDC) [29] can be used to force SARSA to follow the true gradient. We adopt TDC to support the form of Q and the presence of matrix β. Based on the *mean-square projected Bellman error (MSPBE)* objective function [29], TDC employs two additional parameters: matrix ρ and scalar ω and updates β and ρ accordingly on each state transition (cf. lines 17–18 of Algorithm 1, where J is a properly-sized *all-ones* matrix).

5 Experiments

5.1 Setup

We evaluate our approach over three case studies. We consider a range of simulated underlying systems (corresponding to different instantiations of the parameters) and compare obtained confidence results against the corresponding ground truths, via mean-squared error (MSE) metric. We compare strategies generated by our synthesis algorithm, Algorithm 1 (denoted *BA strategy*) against other strategies: a strategy synthesised in [21] (denoted *Synth strategy*), a given probabilistic memoryless strategy (denoted *RS strategy*), and a strategy that randomly select actions at each state (denoted *No strategy*).

First case study involves the pMDP given in [21], endowed with 6 states, 12 transitions, and 2 parameters and the PCTL property $\mathbf{P}_{\geq 0.5}(\mathsf{true}\ \mathcal{U}\ \mathsf{complete})$ ($\mathsf{complete}$ is the label associated to one of the 6 states).

For the second case study (cf. Fig. 2), we extend an MDP model for a smart buildings application [4] to a pMDP with action space $A = \{f_{\text{off}}, f_{\text{on}}\}$ and parameter vector $\theta = \{\theta_1, \theta_2\}$. Actions correspond to the on/off state of a fan inside a room. We verify the satisfaction of the PCTL property $\mathbf{P}_{\geq 0.35} \neg(\mathsf{true}\ \mathcal{U}^{\leq 20}\ (\mathsf{E}, \mathsf{O}))$. Beyond the comparison between strategies described above, we use traces generated by our algorithm and by other strategies with a Bayesian Statistical Model Checking (Bayesian SMC) [30] implementation. SMC collects trajectories from the system, checks whether trajectories satisfy a given property and subsequently uses statistical methods such as *hypothesis testing* to determine whether the system satisfies the property.

Thirdly, we carry out experiments with a well-known pMDP benchmark from [14], the *Randomised Consensus Protocol*. This case study allows to showcase the efficiency and effectiveness of our approach over large MDPs. We consider an instance of the problem with 4112 states and 7692 transitions, where we fix the number of processes $N = 2$ (i.e. two parameters) and the protocol constant $K = 32$ to check the PCTL property $\mathbf{P}_{\geq 0.25}(\mathsf{true}\ \mathcal{U}\ (\mathsf{finished}\ \&\ \mathsf{allCoinsEqualToOne}))$.

Like [21], for convenience of presentation, we have selected all simulated underlying systems such that a single parameter θ_* is responsible for the satisfaction/falsification of the property: $\theta_* = (\theta_1 = \theta_2)$. Intervals over θ_* (namely, feasible sets) corresponding to the system verifying the corresponding property are: $I_1 = [0.369, 0.75]$ (for case 1), $I_2 = [0.0, 0.16]$ (for case 2) and $I_3 = [0.2, 0.5]$ (for case 3), respectively.

Our approach has been implemented in C++. We consider non-informative priors for all parameters involved i.e., uniform ones ($\alpha = \beta = 1$). Over different values of θ_*, data from the simulated underlying system is collected as traces containing state-action pairs visited over time. We gather confidence estimates through n number of runs. If $\mathscr{C}_j$ is the confidence estimate at the j-th run of case study i, then MSE is computed as $= \frac{1}{n}\Sigma_{j=1}^{n}(\mathbb{1}_{I_i}(\theta_*) - \mathscr{C}_j)^2$. For the experiments, we set algorithm parameters (cf. Algorithm 1) as follows. λ is set to 0.8, ω to 0.9, α to $\frac{1}{(\bar{t}+1)^{0.65}}$ and ϵ to $\frac{1}{\bar{t}+1}$. We train β on 1000 episodes for all case studies and set $\mathcal{Z} = 50$ for the large model in case study 3.

5.2 Results

The MSE outcomes for each case study are summarised in Figs. 3a, b and c. The horizontal axis (system parameter) represents values of θ_* for the simulated underlying systems. The intervals I_i above allow to separate systems that satisfy the property from those that do not, by a clear edge/boundary (e.g., for Fig. 3a this edge is rooted at 0.369). The MSE results have been drawn by experiments carried out under limited amount of data (e.g. for the third case study, we have used 10 traces, each of length 10, namely $(t10, l10)$). These results show that our approach clearly outperforms other strategies.

It is important to note that we incur a comparably higher error at system parameters that are very close to the mentioned edge. This is due to the nature of the confidence computation that we perform. For a point closer to the edge, this could yield a posterior distribution that will have its peak centred at the point with probability mass falling almost equally in both the feasible set and outside it. As more data is gathered, the posterior distribution may grow taller and thinner, but with the slightest shift in its peak, a large proportion of the mass may fall on either side, resulting in an increase of the MSE. This increased sensitivity near the edge soon subsides as we move away from the edge, where the mass can now fall in either part of the interval.

Note that we achieve a significant performance for the large pMDP model in case study 3 (cf. Fig. 3c). For a model of this magnitude, the corresponding

Bayes-Adaptive model is enormous, making it impossible to search/traverse it entirely. The proposed generalisation approach embedded in our search algorithm allows to tackle this otherwise intractable problem. Unfortunately, [21] times out (in 1 h after going out of memory) when attempting synthesis, i.e., explicitly evaluating memoryless strategies does not scale well for large models.

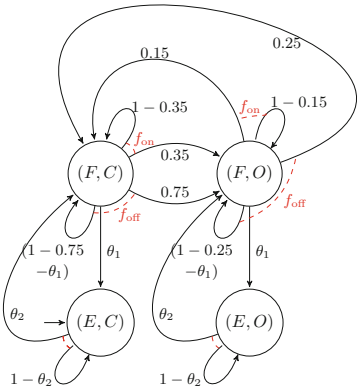

Fig. 2. Extended pMDP model for case study in [4].

Figures 3d and e present results on case study 2. These show that our method is able to gather more useful data than *Synth* and to rapidly converge to the ground truth. Maintaining good performance even at very low amounts of data (e.g. 10 traces or traces of length 5) shows that our approach is robust to the nature of the gathered data. The major reason behind this is that the current strategy constantly looks for parameterised transitions as much as possible: it is over these transitions that confidence gain may happen. This is in stark contrast to techniques like SMC, where the length of horizon of the trace needs to be long enough to either reach a designated state or find counterexamples for the given property.

Figure 3f provides results for an experiment conducted over case study 2, and shows two significant aspects of our approach: first, the information content of the traces that we have generated from our approach, by comparing them against those generated from other strategies; and secondly, the demonstration that SMC can be problematic in situations where one has access to only a limited amount of data. Running traces generated by different strategies (*BA* vs *Synth* vs *No*) through a Bayesian SMC algorithm, demonstrates that our (*BA*) approach converges rapidly to the ground truth, faster than other methods (*Synth* and *No*). This shows that our traces encompass much richer information content to compute better confidence estimates to decide the satisfiability of the property.

SMC provides outcomes that are usually much faster than canonical model checking tools. However, for case study 2, the property we have selected is a negative *bounded-time* property that requires falsification by reaching a specific state (E,O). This is a tricky property to ascertain via SMC, due to the lack of counterexamples with trace lengths much shorter than the formula horizon. On the other hand, such traces processed with our approach (i.e. confidence computation using the posterior distribution) yield much better results than Bayesian SMC. This shows that we can work with much shorter horizons than the formula horizon and are still able to accurately verify properties. Performance at shorter trace lengths is an important performance criterion for large models, like the one in case study 3, where you would need fairly longer trace lengths (e.g. 1000 or more) for SMC to work, whereas our approach is able to verify the property with a couple of orders of magnitudes lower trace lengths (e.g. 10).

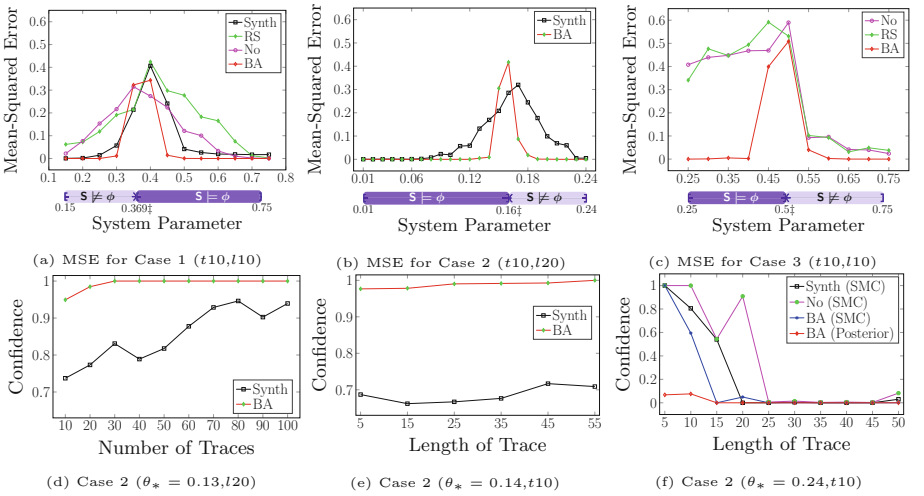

Fig. 3. Mean-squared error (MSE) and average confidence results for the three case studies. The number of generated traces or the length of generated traces is indicated by values prefixed with t or l, respectively. Translucent bars along the x-axes of a, b and c designate the simulated underlying systems that satisfy the property ϕ and those which do not. Respective edges/boundaries are followed by a ‡ symbol.

6 Conclusions and Future Work

We have a data-based efficient verification approach to assert whether a partially unknown probabilistic system satisfies a given property expressed as a logical specification. Our approach takes into account memory in calculating optimal strategies to gather data from the underlying system so as to derive the most accurate confidence estimates possible.

As future work, based on the updated confidence value, one could *tune/repair* the parametric model until a decisive confidence is achieved. For instance, if the output confidence value is 0.5, then there exists an equal chance that the property is either satisfied or not over the system. If this value has been obtained after gathering a substantial amount of data, this may mean that the employed parametric model was not supportive enough to gauge the satisfaction of the property, hence it could be adjusted until a substantial judgement about the satisfiability can be made.

Furthermore, in this work, actions are exclusively selected for learning tasks. Instead, one might choose them in the context of model classification (i.e., parameters selection), in order to steer the system towards property satisfaction.

References

1. Araya-López, M., Buffet, O., Thomas, V., Charpillet, F.: Active learning of MDP models. In: Sanner, S., Hutter, M. (eds.) EWRL 2011. LNCS (LNAI), vol. 7188, pp. 42–53. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29946-9_8
2. Arming, S., Bartocci, E., Chatterjee, K., Katoen, J.-P., Sokolova, A.: Parameter-independent strategies for pMDPs via POMDPs. In: McIver, A., Horvath, A. (eds.) QEST 2018. LNCS, vol. 11024, pp. 53–70. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99154-2_4
3. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge (2008)
4. Cauchi, N., Abate, A.: StocHy: automated verification and synthesis of stochastic processes. In: 25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS) (2019)
5. Chen, Y., Nielsen, T.: Active learning of Markov decision processes for system verification. In: 2012 11th International Conference on Machine Learning and Applications (ICMLA), vol. 2, pp. 289–294, December 2012
6. Cubuktepe, M., et al.: Sequential convex programming for the efficient verification of parametric MDPs. In: Legay, A., Margaria, T. (eds.) TACAS 2017. LNCS, vol. 10206, pp. 133–150. Springer, Heidelberg (2017). https://doi.org/10.1007/978-3-662-54580-5_8
7. Dehnert, C., Junges, S., Katoen, J.-P., Volk, M.: A Storm is Coming: a modern probabilistic model checker. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31
8. Duff, M.O.: Optimal learning: computational procedures for Bayes-Adaptive Markov decision processes. Ph.D. thesis (2002)
9. Gordon, G.J.: Chattering in SARSA(λ) - a CMU learning lab internal report. Technical report (1996)
10. Guez, A., Heess, N., Silver, D., Dayan, P.: Bayes-adaptive simulation-based search with value function approximation. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N.D., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems, Curran Associates, Inc., vol. 27, pp. 451–459 (2014)
11. Guez, A., Silver, D., Dayan, P.: Efficient Bayes-adaptive reinforcement learning using sample-based search. In: Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012, Proceedings of a Meeting Held 3–6 December 2012, Lake Tahoe, Nevada, United States, pp. 1034–1042 (2012)
12. Haesaert, S., Van den Hof, P.M., Abate, A.: Data-driven property verification of grey-box systems by Bayesian experiment design. In: American Control Conference (ACC), 2015, IEEE, pp. 1800–1805 (2015)
13. Hahn, E.M., Han, T., Zhang, L.: Synthesis for PCTL in parametric markov decision processes. In: Bobaru, M., Havelund, K., Holzmann, G.J., Joshi, R. (eds.) NFM 2011. LNCS, vol. 6617, pp. 146–161. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-20398-5_12
14. Hahn, E.M., Hermanns, H., Wachter, B., Zhang, L.: PARAM: a model checker for parametric Markov models. In: Touili, T., Cook, B., Jackson, P. (eds.) CAV 2010. LNCS, vol. 6174, pp. 660–664. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-14295-6_56

15. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects Comput. **6**(5), 512–535 (1994)
16. Henriques, D., Martins, J.G., Zuliani, P., Platzer, A., Clarke, E.M.: Statistical model checking for Markov decision processes. In: Proceedings of the 2012 Ninth International Conference on Quantitative Evaluation of Systems, QEST 2012, IEEE Computer Society, Washington, DC, USA. pp. 84–93 (2012)
17. Legay, A., Sedwards, S., Traonouez, L.-M.: Scalable verification of Markov decision processes. In: Canal, C., Idani, A. (eds.) SEFM 2014. LNCS, vol. 8938, pp. 350–362. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-15201-1_23
18. Mao, H., Chen, Y., Jaeger, M., Nielsen, T.D., Larsen, K.G., Nielsen, B.: Learning Markov decision processes for model checking. In: Proceedings Quantities in Formal Methods, QFM 2012, Paris, France, 28 August 2012, pp. 49–63 (2012)
19. Marom, O., Rosman, B.: Belief reward shaping in reinforcement learning. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018)
20. Polgreen, E., Wijesuriya, V.B., Haesaert, S., Abate, A.: Data-efficient Bayesian verification of parametric Markov chains. In: Agha, G., Van Houdt, B. (eds.) QEST 2016. LNCS, vol. 9826, pp. 35–51. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-43425-4_3
21. Polgreen, E., Wijesuriya, V.B., Haesaert, S., Abate, A.: Automated experiment design for data-efficient verification of parametric Markov decision processes. In: Bertrand, N., Bortolussi, L. (eds.) QEST 2017. LNCS, vol. 10503, pp. 259–274. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66335-7_16
22. Poupart, P., Vlassis, N., Hoey, J., Regan, K.: An analytic solution to discrete Bayesian reinforcement learning. In: Proceedings of the 23rd International Conference on Machine Learning, ICML 2006, ACM, New York, NY, USA, pp. 697–704 (2006)
23. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edn. Wiley, New York (1994)
24. Quatmann, T., Dehnert, C., Jansen, N., Junges, S., Katoen, J.-P.: Parameter synthesis for Markov models: faster than ever. In: Artho, C., Legay, A., Peled, D. (eds.) ATVA 2016. LNCS, vol. 9938, pp. 50–67. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46520-3_4
25. Ross, S., Pineau, J., Chaib-draa, B., Kreitmann, P.: A Bayesian approach for learning and planning in partially observable Markov decision processes. J. Mach. Learn. Res. **12**(May), 1729–1770 (2011)
26. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: Alur, R., Peled, D.A. (eds.) CAV 2004. LNCS, vol. 3114, pp. 202–215. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27813-9_16
27. Silver, D., Sutton, R.S., Müller, M.: Temporal-difference search in Computer Go. Mach. Learn. **87**(2), 183–219 (2012)
28. Sutton, R.S., Barto, A.G.: Introduction to Reinforcement Learning, 1st edn. MIT Press, Cambridge (1998)
29. Sutton, R.S., Maei, H.R., Precup, D., Bhatnagar, S., Silver, D., Szepesvári, C., Wiewiora, E.: Fast gradient-descent methods for temporal-difference learning with linear function approximation. In: Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, ACM, New York, NY, USA, pp. 993–1000 (2009)
30. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to Stateflow/Simulink verification. Form. Methods Syst. Des. **43**(2), 338–367 (2013)