PAC Statistical Model Checking of Mean Payoff in Discrete- and Continuous-Time MDP

Chaitanya Agarwal[1], Shibashis Guha[2(✉)], Jan Křetínský[3], and Pazhamalai Muruganandham[4]

[1] New York University, New York, USA
[2] Tata Institute of Fundamental Research, Mumbai, India
Shibashis@tifr.res.in
[3] Technical University of Munich, Munich, Germany
[4] Chennai Mathematical Institute, Chennai, India

Abstract. Markov decision processes (MDP) and continuous-time MDP (CTMDP) are the fundamental models for non-deterministic systems with probabilistic uncertainty. Mean payoff (a.k.a. long-run average reward) is one of the most classic objectives considered in their context. We provide the first algorithm to compute mean payoff probably approximately correctly in unknown MDP; further, we extend it to unknown CTMDP. We do not require any knowledge of the state space, only a lower bound on the minimum transition probability, which has been advocated in literature. In addition to providing probably approximately correct (PAC) bounds for our algorithm, we also demonstrate its practical nature by running experiments on standard benchmarks.

1 Introduction

Markov decision process (MDP) [7,30,32] is a basic model for systems featuring both probabilistic and non-deterministic behaviour. They come in two flavours: *discrete-time* MDP (often simply MDP) and *continuous-time* MDP (CTMDP). While the evolution of MDP happens in discrete steps, their natural real-time extension CTMDP additionally feature random time delays governed by exponential probability distributions. Their application domain ranges across a wide spectrum, e.g. operations research [10,16], power management and scheduling [31], networked and distributed systems [19,22], or communication protocols [28], to name a few. One of the key aspects of such systems is their performance, often formalized as *mean payoff* (also called long-run average reward), one of the classic and most studied objectives on (CT)MDP [30] with numerous applications [17]. In this context, probabilistic model checking and performance evaluation intersect [5]. While

This work has been partially supported by the DST-SERB project SRG/2021/000466 *Zero-sum and Nonzero-sum Games for Controller Synthesis of Reactive Systems* and by the German Research Foundation (DFG) projects 427755713 (KR 4890/3-1) *Group-By Objectives in Probabilistic Verification (GOPro)* and 383882557 (KR 4890/2-1) *Statistical Unbounded Verification (SUV).*

S. Shoham and Y. Vizel (Eds.): CAV 2022, LNCS 13372, pp. 3–25, 2022.
https://doi.org/10.1007/978-3-031-13188-2_1

the former takes the verification perspective of the worst-case analysis and the latter the perspective of optimization for the best case, they are mathematically dual and thus algorithmically the same.

The range of analysis techniques provided by literature is very rich, encompassing linear programming, policy iteration, or value iteration. However, these are applicable only in the setting where the (CT)MDP is known (*whitebox* setting). In order to handle the *blackbox* setting, where the model is unknown or only partially known, *statistical model checking* (SMC) [37] relaxes the requirement of the hard guarantees on the correctness (claimed precision) of the result. Instead it uses *probably approximately correct* (PAC) analysis, which provides essentially a confidence interval on the result: with probability (confidence) at least $1 - \delta$, the result of the analysis is ε-close to the true value. This kind of analysis may be applicable to those systems for which we do not have exclusive access to their internal functionalities, but we can still observe their behaviour.

In this paper, we provide the *first algorithm with PAC bounds on the mean payoff in blackbox MDP*. We treat both the discrete-time and continuous-time MDP, and the SMC algorithm not only features PAC bounds (returning the result with prescribed precision and confidence), but an anytime algorithm (gradually improving the result and, if terminated prematurely, can return the current approximation with its precision and the required confidence).

The difficulty with blackbox models is that we do not know the exact transition probabilities, not even the number of successors for an action from a state. The algorithm thus must simulate the MDP to obtain any information. The visited states can be augmented to a model of the MDP and statistics used to estimate the transition probabilities. The estimates can be used to compute mean payoff precisely on the model. The results of [12] and [33] then provide a method for estimating the number of times each state-action pair needs to be visited in an MDP to obtain a PAC bound on the expected mean-payoff value of the original MDP. However, notice that this requires that the topology be learnt perfectly, for which we either need some knowledge of the state space or recent development in the spirit of [3]. On the one hand, this simple algorithm thus follows in a straightforward way from the recent results in the literature (although to the best of our knowledge it has not been presented as such yet). On the other hand, the required number of samples using these bounds is *prohibitively large*, and therefore, giving guarantees with such analysis is not feasible at all in practice. In fact, the numbers are astronomic already for Markov chains with a handful of states [13]. We discuss further drawbacks of such a naïve solution in Sect. 3. Our main contribution in this paper is a *practical algorithm*. It takes the most promising actions from every state and uses the *on-demand value iteration* [2], not even requiring an exhaustive exploration of the entire MDP. Using techniques of [3,13], we can show that the partial model captures enough information. Most importantly, instead of using [12,33], the PAC bounds are derived directly from the concrete confidence intervals, reflecting the width of each interval and the topology of the model, in the spirit of the practical SMC for reachability [3].

Our contribution can be summarized as follows:

- We provide the first algorithm with PAC bounds on the mean payoff in blackbox MDP (Sect. 4) and its extension to blackbox CTMDP (Sect. 5).
- We discuss the drawbacks of a possible more straightforward solution and how to overcome them (in Sect. 3 on the conceptual level, before we dive into the technical algorithms in the subsequent sections).
- We evaluate the algorithm on the standard benchmarks of MDP and CTMDP and discuss the effect of heuristics, partial knowledge of the model, and variants of the algorithms (Sect. 6).

Related Work. SMC of unbounded-horizon properties of MDPs was first considered in [23,29] for reachability. [20] gives a model-free algorithm for ω-regular properties, which is convergent but provides no bounds on the current error. Several approaches provide SMC for MDPs and unbounded-horizon properties with *PAC guarantees*. Firstly, the algorithm of [18] requires (1) the mixing time T of the MDP (2) the ability to restart simulations also in non-initial states (3) visiting *all* states sufficiently many times, and thus (4) the knowledge of the size of the state space $|S|$. Secondly, [9], based on delayed Q-learning [34], lifts the assumptions (2) and (3) and instead of (1) requires only (a bound on) the minimum transition probability p_{min}. Thirdly, [3] additionally lifts the assumption (4), keeping only p_{min}, as in this paper. In [13], it is argued that while unbounded-horizon properties cannot be analysed without any information on the system, knowledge of (a lower bound on) the minimum transition probability p_{min} is a relatively light and realistic assumption in many scenarios, compared to the knowledge of the whole topology. In this paper, we thus adopt this assumption.

In contrast to SMC that uses possibly more (re-started) runs of the system, there are *online learning* approaches, where the desired behaviour is learnt for the single run. Model-based learning algorithms for mean payoff have been designed both for minimizing regret [4,36] as well as for PAC online learning [25,26].

Due to lack of space, the proofs and some more experimental results and discussions appear in [1].

2 Preliminaries

A *probability distribution* on a finite set X is a mapping $\rho : X \mapsto [0,1]$, such that $\sum_{x \in X} \rho(x) = 1$. We denote by $\mathcal{D}(X)$ the set of probability distributions on X.

Definition 1. (MDP). *A* Markov decision process *is a tuple of the form* $\mathcal{M} = (\mathsf{S}, s_{init}, \mathsf{Act}, \mathsf{Av}, \mathbb{T}, r)$*, where* S *is a finite set of states,* $s_{init} \in \mathsf{S}$ *is the initial state,* Act *is a finite set of actions,* $\mathsf{Av} : \mathsf{S} \to 2^{\mathsf{Act}}$ *assigns to every state a set of available actions,* $\mathbb{T} : \mathsf{S} \times \mathsf{Act} \to \mathcal{D}(\mathsf{S})$ *is a transition function that given a state* s *and an action* $a \in \mathsf{Av}(s)$ *yields a probability distribution over successor states, and* $r : \mathsf{S} \to \mathbb{R}^{\geq 0}$ *is a reward function, assigning rewards to states.*

For ease of notation, we write $\mathbb{T}(s,a,t)$ instead of $\mathbb{T}(s,a)(t)$. We denote by $\mathsf{Post}(s,a)$, the set of states that can be reached from s through action a. Formally, $\mathsf{Post}(s,a) = \{t \mid \mathbb{T}(s,a,t) > 0\}$.

The choices of actions are resolved by strategies, generally taking history into account and possibly randomizing. However, for mean payoff it is sufficient to consider *positional* strategies of the form $\pi : \mathsf{S} \to \mathsf{Act}$. The semantics of an MDP with an initial state s_{init} is given in terms of each strategy σ inducing a Markov chain $\mathcal{M}^{\sigma}_{s_{\mathsf{init}}}$ with the respective probability space and unique probability measure $\mathbb{P}^{\mathcal{M}^{\sigma}_{s_{\mathsf{init}}}}$, and the expected value $\mathbb{E}^{\mathcal{M}^{\sigma}_{s_{\mathsf{init}}}}[F]$ of a random variable F (see e.g. [6]). We drop $\mathcal{M}^{\sigma}_{s_{\mathsf{init}}}$ when it is clear from the context.

End Components An *end-component* (EC) $M = (T, A)$, with $\emptyset \neq T \subseteq \mathsf{S}$ and $A : T \to 2^{\mathsf{Act}}$ of an MDP $\mathcal{M}$ is a *sub-MDP* of $\mathcal{M}$ such that: for all $s \in T$, we have that $A(s)$ is a subset of the actions available from s; for all $a \in A(s)$, we have $\mathsf{Post}(s, a) \subseteq T$; and, it's underlying graph is strongly connected. A *maximal end-component* (MEC) is an EC that is not included in any other EC. Given an MDP $\mathcal{M}$, the set of its MECs is denoted by $\mathsf{MEC}(\mathcal{M})$. For $\mathsf{MEC}(\mathcal{M}) = \{(T_1, A_1), \ldots, (T_n, A_n)\}$, we define $\mathsf{MEC}_{\mathsf{S}} = \bigcup_{i=1}^{n} T_i$ as the set of all states contained in some MEC.

Definition 2. (continuous-time MDP (CTMDP)). *A* continuous-time Markov decision process *is a tuple of the form* $\mathcal{M} = (\mathsf{S}, s_{\mathit{init}}, \mathsf{Act}, \mathsf{Av}, \mathsf{R}, r)$, *where* S *is a finite set of states,* $s_{\mathit{init}} \in \mathsf{S}$ *is the initial state,* Act *is a finite set of actions,* $\mathsf{Av} : \mathsf{S} \to 2^{\mathsf{Act}}$ *assigns to every state a set of available actions,* $\mathsf{R} : \mathsf{S} \times \mathsf{Act} \times \mathsf{S} \to \mathbb{R}_{\geq 0}$ *is a transition rate matrix that given a state* s *and an action* $a \in \mathsf{Av}(s)$ *defines the set of successors* t *of* s *on action* a *if* $\mathsf{R}(s, a, t) > 0$, *and* $r : \mathsf{S} \to \mathbb{R}_{\geq 0}$ *is a reward rate function, assigning a reward function to a state denoting the reward obtained for spending unit time in* s.

A strategy in a CTMDP decides immediately after entering a state which action needs to be chosen from the current state. For a given state $s \in \mathsf{S}$, and an action $a \in \mathsf{Av}(s)$, we denote by $\lambda(s, a) = \sum_t \mathsf{R}(s, a, t) > 0$ the *exit rate* of a in s. The *residence time* for action a in s is exponentially distributed with mean $\frac{1}{\lambda(s,a)}$. An equivalent way of looking at CTMDP is that in state s, we wait for a time which is exponentially distributed with mean $\lambda(s, a)$, and then with probability $\Delta(s, a, t) = \mathsf{R}(s, a, t)/\lambda(s, a)$, we make a transition to state t. The reward accumulated for spending time t in s is $r(s) \cdot \mathsf{t}$.

Uniformization. A *uniform* CTMDP has a constant exit rate C for all state-action pairs i.e., $\lambda(s, a) = C$ for all states $s \in \mathsf{S}$ and actions $a \in \mathsf{Av}(\mathsf{s})$. The procedure of converting a non-uniform CTMDP into a uniform one is called *uniformization*. Consider a non-uniform CTMDP $\mathcal{M}$. Let $C \in \mathbb{R}_{\geq 0}$ such that $C \geqslant \lambda(s, a)$ for all $s \in \mathsf{S}$ and $a \in \mathsf{Act}$. We can obtain a uniform CTMDP $\mathcal{M}_C$ by assigning the new rates.

$$\mathsf{R}'(s, a, t) = \begin{cases} \mathsf{R}(s, a, t) & \text{if } s \neq t \\ \mathsf{R}(s, a, t) + C - \lambda(s, a) & \text{if } s = t \end{cases} \tag{1}$$

For every action $a \in \mathsf{Av}(s)$ from each state s in the new CTMDP we have a self loop if $\lambda(s, a) < C$. Due to a constant transition rate, the mean interval time between two any two actions is constant.

Mean Payoff. In this work, we consider the (maximum) *mean payoff* (or *long-run average reward*) of an MDP $\mathcal{M}$, which intuitively describes the (maximum) average reward per step we expect to see when simulating the MDP for time going to infinity. Formally, let S_i, A_i, R_i be random variables giving the state visited, action played, and reward obtained in step i, and for CTMDP, T_i the time spent in the state appearing in step i. For MDP, $R_i := r(S_i)$, whereas for CTMDP, $R_i := r(S_i) \cdot T_i$; consequently, for a CTMDP and a strategy π, we have $\mathbb{E}_s^\pi(R_i) = \frac{r(S_i)}{\lambda(S_i,A_i)}$.

Thus given a strategy π, the n-step average reward is

$$v_n^\pi(s) := \mathbb{E}_s^\pi\left(\frac{1}{n}\sum_{i=0}^{n-1} R_i\right) = \frac{1}{n}\sum_{i=0}^{n-1}\frac{r(S_i)}{\lambda(S_i,A_i)}.$$

with the latter equality holding for CTMDP. For both MDP and CTMDP, the *mean payoff* is then

$$v(s) := \max_\pi \liminf_{n\to\infty} v_n^\pi,$$

where the maximum over all strategies can also be without loss of generality restricted to the set of positional strategies Π^{MD}. A well-known alternative characterization we use in this paper is

$$v(s) = \max_{\pi\in\Pi^{\mathsf{MD}}} \sum_{M\in\mathsf{MEC}(\mathcal{M})} \mathbb{P}_s^\pi[\Diamond\Box M]\cdot v_M, \tag{2}$$

where $\Diamond$ and $\Box$ respectively denote the standard LTL operators *eventually* and *always* respectively. Further, $\Diamond\Box M$ denotes the set of paths that eventually remain forever within M and v_M is the unique value achievable in the (CT)MDP restricted to the MEC M. Note that v_M does not depend on the initial state chosen for the restriction.

We consider algorithms that have a limited information about the MDP.

Definition 3. (Blackbox and greybox). *An algorithm inputs an MDP or a CTMDP as* blackbox *if*

- *it knows* s_{init},
- *for a given state,*[1] *an oracle returns its available actions,*
- *given a state* s *and action* a*, it can sample a successor* t *according to* $\mathbb{T}(s,a)$,
- *it knows* $p_{\mathsf{min}} \leqslant \min_{\substack{s\in\mathsf{S},a\in\mathsf{Av}(s)\\ t\in\mathsf{Post}(s,a)}} \mathbb{T}(s,a,t)$*, an under-approximation of the minimum transition probability.*

When input as greybox*, it additionally knows the number* $|\mathsf{Post}(s,a)|$ *of successors for each state* s *and action* a*. Note that the exact probabilities on the transitions in an MDP or the rates in a CTMDP are unknown for both blackbox and greybox learning settings.*

[1] In contrast to practical setups in monitoring, our knowledge of the current state is complete, i.e., the previously visited states can be uniquely identified.

3 Overview of Our Approach

Since no solutions are available in the literature and our solution consists of multiple ingredients, we present it in multiple steps to ease the understanding. First, we describe a more *naïve* solution and pinpoint its drawbacks. Second, we give an *overview* of a more sophisticated solution, eliminating the drawbacks. Third, we fill in its *details* in the subsequent sections. Besides, each of the three points is first discussed on discrete-time MDPs and then on continuous-time MDPs. The reason for this is twofold: the separation of concerns simplifies the presentation; and the algorithm for discrete-time MDP is equally important and deserves a standalone description.

3.1 Naïve Solution

We start by suggesting a conceptually simple solution. We can learn mean payoff MP in an MDP $\mathcal{M}$ as follows:

(i) Via simulating the MDP $\mathcal{M}$, we learn a model $\mathcal{M}'$ of $\mathcal{M}$, i.e., we obtain confidence intervals on the *t*ransition *p*robabilities of $\mathcal{M}$ (of some given width ε_{TP}, called *TP-imprecision*, and confidence $1 - \delta_{TP}$, where δ_{TP} is called *TP-inconfidence*).

(ii) We compute the mean payoff $\widehat{MP}$ on the (imprecise) model $\mathcal{M}'$.

(iii) We compute the *MP-imprecision* $\varepsilon_{MP} = |\widehat{MP} - MP|$ of the *m*ean *p*ayoff from the TP-imprecision by the "robustness" theorem [8] which quantifies how mean payoff can change when the system is perturbed with a given maximum perturbation. Further, we compute the overall *MP-inconfidence* δ_{MP} from the TP-inconfidence δ_{TP}; in particular, we can simply accumulate all the uncertainty and set $\delta_{MP} = |\mathbb{T}| \cdot \delta_{TP}$, where $|\mathbb{T}|$ is the number of transitions. The result is then probably approximately correct, being ε_{MP}-precise with confidence $1 - \delta_{MP}$. (Inversely, from a desired ε_{MP} we can also compute a sufficient ε_{TP} to be used in the first step.)

Learning the model, i.e. the transition probabilities, can be done by observing the simulation runs and collecting, for each state-action pair (s, a), a statistics of which states occur right after playing a in s. The frequency of each successor t among all successors then estimates the transition probability $\mathbb{T}(s, a, t)$. This is the standard task of estimating the generalized Bernoulli variable (a fixed distribution over finitely many options) with confidence intervals. We stop simulating when *each* transition probability has a precise enough confidence interval (with ε_{TP} and δ_{TP} yielded by the robustness theorem from the desired overall precision).[2] The drawbacks are *(D1: uniform importance)* that even transitions with

[2] Several non-trivial questions are dealt with later on: how to resolve the action choices during simulations; when to stop each simulation run and start a new one; additionally, in the black-box setting, when do we know that all successors of each transition have been observed. In particular, the last one is fundamental for the applicability of the robustness theorem. While the literature typically assumes the greybox setting or even richer information, to allow for such an algorithm with PAC bounds, our approach only needs p_{min}.

little to no impact on the mean payoff have to be estimated precisely (with ε_{TP} and δ_{TP}); and *(D2: uniform precision required)* that, even restricting our attention to "important" transitions, it may take a long time before the last one is estimated precisely (while others are already estimated overly precisely).

Subsequently, using standard algorithms the mean payoff $\widehat{MP}$ can be computed precisely by linear programming [30] or precisely enough by value iteration [2]. The respective MP can then be estimated by the robustness theorem [8], which yields for a given maximum perturbation of transition probabilities (in our case, $\varepsilon_{TP}/2$) an upper bound on the respective perturbation of the mean payoff $\varepsilon_{MP}/2$. The drawbacks are *(D3: uniform precision utilized)* that more precise confidence intervals for transitions (obtained due to D2) are not utilized, only the maximum imprecision is taken into account; and *(D4: a-priori bounds)* that the theorem is extremely conservative. Indeed, it reflects neither the topology of the MDP nor how impactful each transition is and thus provides an a-priori bound, extremely loose compared to the possible values of mean payoff that can be actually obtained for concrete values within the confidence intervals. This is practically unusable beyond a handful of states even for Markov chains [13].

For CTMDP $\mathcal{M}$, we additionally need to estimate the rates (see below how). Subsequently, we can uniformize the learnt CTMDP $\mathcal{M}'$. Mean payoff of the uniformized CTMDP is then equal to the mean payoff of its embedded MDP[3]. Hence, we can proceed as before but we also have to compute (i) confidence intervals for the rates from finitely many observations, and (ii) the required precision and confidence of these intervals so that the respective induced error on the mean payoff is not too large. Hence all the drawbacks are inherited and, additionally, also applied to the estimates of the rates. Besides, *(D5: rates)* while imprecisions of rates do not increase MP-imprecision too much, the bound obtained via uniformization and the robustness theorem is very loose. Indeed, imprecise rates are reflected as imprecise self-loops in the uniformization, which themselves do not have much impact on the mean payoff, but can increase the TP-imprecision and thus hugely the MP-imprecision from the robustness theorem.

Finally, note that for both types of MDP, *(D6: not anytime)* this naïve algorithm is not an anytime algorithm[4] since it works with pre-computed ε_{TP} and δ_{TP}. Instead it returns the result with the input precision if given enough time; if not given enough time, it does not return anything (also, if given more time, it does not improve the precision).

3.2 Improved Solution

Now we modify the solution so that the drawbacks are eliminated. The main ideas are (i) to allow for differences in TP-imprecisions (ε_{TP} can vary over

[3] An embedded MDP of a CTMDP is obtained by considering for every state s, actions $a \in \mathsf{Av}(s)$, and transitions $t \in \mathsf{Post}(s,a)$, such that $\mathbb{T}(s,a,t) = \Delta(s,a,t)$, and by disregarding the transition rate matrix.

[4] An anytime algorithm can, at every step, return the current estimate with its imprecision, and this bound converges to 0 in the limit.

transitions) and even deliberately ignore less important transitions and instead improve precision for transitions where more information is helpful the most; (ii) rather than using the a-priori robustness theorem, to utilize the precision of each transition to its maximum; and (iii) to give an anytime algorithm that reflects the current confidence intervals and, upon improving them, can efficiently improve the mean-payoff estimate without recomputing it from scratch. There are several ingredients used in our approach.

Firstly, [2] provides an anytime algorithm for approximating mean payoff in a fully known MDP. The algorithm is a version of value iteration, called *on-demand*, performing improvements (so called Bellman updates) of the mean-payoff estimate in each state. Moreover, the algorithm is simulation-based, performing the updates in the visited states, biasing towards states where a more precise estimate is helpful the most ("on demand"). This matches well our learning setting. However, the approach assumes precise knowledge of the transition probabilities and, even more importantly, heavily relies on the knowledge of MECs. Indeed, it decomposes the mean-payoff computation according to Eq. 2 into computing mean payoff within MECs and optimizing (weighted) reachability of the MECs (with weights being their mean payoffs). When the MECs are unknown, none of these two steps can be executed.

Secondly, [3] provides an efficient way of learning reachability probabilities (in the greybox and blackbox settings). Unfortunately, since it considers TP-inconfidence to be the same for all transitions, causing different TP-imprecisions, the use of robustness theorem in [3] makes the algorithm used there practically unusable in many cases. On a positive note, the work identifies the notion of δ_{TP}-sure EC, which reflects how confident we are, based on the simulations so far, that a set of states is an EC. This notion will be crucial also in our algorithm.

Both approaches are based on "bounded value iteration", which computes at any moment of time both a lower and an upper bound on the value that we are approximating (mean payoff or reachability, respectively). This yields anytime algorithms with known imprecision, the latter—being a learning algorithm on an incompletely known MDP—only with some confidence. Note that the upper bound converges only because ECs are identified and either collapsed (in the former) or deflated [24] (in the latter), meaning their upper bounds are decreased in a particular way to ensure correctness.

Our algorithm on (discrete-time) MDP $\mathcal{M}$ performs, essentially, the following. It simulates $\mathcal{M}$ in a similar way as [3]. With each visit of each state, not only it updates the model (includes this transition and improves the estimate of the outgoing transition probabilities), but also updates the estimate of the mean payoff by a Bellman update. Besides, at every moment of time, the current model yields a hypothesis what the actual MECs of $\mathcal{M}$ are and the respective confidence. While we perform the Bellman updates on all visited states deemed transient, the states deemed to be in MECs are updated separately, like in [2]. However, in contrast to [2], where every MEC is fully known and can thus be collapsed, and in contrast to the "bounded" quotient of [3] (see Appendix A of [1]), we instead introduce a special action stay in each of its states, which simulates *staying* in the (not fully known) MEC and obtaining its mean-payoff estimate via reachability:

Definition 4. (stay-augmented MDP). *Let* $\mathcal{M} = (\mathsf{S}, s_{init}, \mathsf{Act}, \mathsf{Av}, \mathbb{T}, r)$ *be an MDP and* $l, u : \mathsf{MEC}(\mathcal{M}) \to [0,1]$ *be real functions on MECs. We augment the* stay *action to* $\mathcal{M}$ *to obtain* $\mathcal{M}' = (\mathsf{S}', s_{init}, \mathsf{Act}', \mathsf{Av}', \mathbb{T}', r')$*, where*

- $\mathsf{S}' = \mathsf{S} \uplus \{s_+, s_-, s_?\}$,
- $\mathsf{Act}' = \mathsf{Act} \uplus \{\mathsf{stay}\}$,
- $\mathsf{Av}'(s) = \begin{cases} \mathsf{Av}(s) & \text{for } s \in S \setminus \bigcup \mathsf{MEC}(\mathcal{M}) \\ \mathsf{Av}(s) \cup \{\mathsf{stay}\} & \text{for } s \in \bigcup \mathsf{MEC}(\mathcal{M}) \\ \{\mathsf{stay}\} & \text{for } s \in \{s_+, s_-, s_?\} \end{cases}$
- $\mathbb{T}'$ *extends* $\mathbb{T}$ *by* $\mathbb{T}'(s, \mathsf{stay}) = \{s_+ \mapsto l(M), s_- \mapsto 1 - u(M), s_? \mapsto u(M) - l(M)\}$ *on* $s \in M \in \mathsf{MEC}(\mathcal{M})$ *and by* $\mathbb{T}'(s, \mathsf{stay}, s) = 1$ *for* $s \in \{s_+, s_-, s_?\}$.
- r' *extends* r *by* $r'(s_+) = r'(s_?) = r'(s_-) = 0$.[5]

Corollary 1. *If* l, u *are valid lower and upper bounds on the mean-payoff within MECs of* $\mathcal{M}$ *then* $\max_\sigma \mathbb{P}^{M^\sigma}[\Diamond\{s_+\}] \leqslant v(s_{init}) \leqslant max_\sigma \mathbb{P}^{M^\sigma}[\Diamond\{s_+, s_?\}]$[6] *where,* $\max_\sigma \mathbb{P}^{M^\sigma}[\Diamond S]$ *gives the maximum probability of reaching some state in* S *over all strategies.*

This turns the problem into reachability, and thus allows for deflating (defined for reachability in [3]) and an algorithm combining [3] and [2]. The details are explained in the next section. To summarize (D1) and (D2) are eliminated by not requiring uniform TP-imprecisions; (D3) and (D4) are eliminated via updating lower and upper bounds (using deflating) instead of using the robustness theorem.

Concerning CTMDP, in Sect. 5 we develop a confidence interval computation for the rates. Further, we design an algorithm deriving the MP-imprecision resulting from the rate imprecisions, that acts directly on the CTMDP and not on the embedded MDP of the uniformization. This effectively removes (D5).

4 Algorithm for Discrete-Time MDP

Now that we explained the difficulties of a naïve approach, and the concepts from literature together with novel ideas to overcome them, we describe the actual algorithm for the discrete-time setting. Following a general outline of the algorithm, we give detailed explanations behind the components and provide the statistical guarantees the algorithm gives. Detailed pseudocode of the algorithms for this section is provided in Appendix B of [1].

Overall Algorithm and Details. Our version of an on-demand value iteration for mean payoff in black-box MDP is outlined in Algorithm 1. Initially, the input MDP $\mathcal{M}$ is augmented with terminal states ($\{s_+, s_-, s_?\}$) to obtain the

[5] A higher transition probability to s_+ indicates that the MEC has high value, a higher transition probability to $s_?$ indicates high uncertainty in the value of the MEC, while a higher transition probability to s_- indicates that the MEC has low value.

[6] For simplicity of the presentation, we assume the rewards are between 0 and 1, for all states. If they are not, we can always rescale them to [0,1] by dividing them by the maximum reward observed so far and correspondingly adjust $\mathbb{T}(\cdot, \mathsf{stay}, \cdot)$.

stay-augmented MDP $\mathcal{M}'$. We learn a stay-augmented MDP $\mathcal{M}' = (\mathsf{S}', s_{\mathsf{init}}, \mathsf{Act}', \mathsf{Av}', \mathbb{T}', r')$ by collecting samples through several simulation runs (Lines 5-8). Over the course of the algorithm, we identify MECs with δ_{TP} confidence (Line 13) and gradually increase precision on their respective values (Lines 9-11). As stated earlier, these simulations are biased towards actions that lead to MECs potentially having higher rewards. Values for MECs are encoded using the stay action (Line 12) and propagated throughout the model using bounded value iteration (Lines 14-19). In Line 14, we reinitialize the values of the states in the partial model since new MECs may be identified and also existing MECs may change. Finally, we claim that the probability estimates $\mathbb{T}'$ are correct with confidence δ_{MP} and if the bounds on the value are precise enough, we terminate the algorithm. Otherwise, we repeat this overall process with improved bounds (Line 20).

Simulation. The SIMULATE function simulates a run over the input blackbox MDP $\mathcal{M}$ and returns the visited states in order. The simulation of $\mathcal{M}'$ is executed by simulating $\mathcal{M}$ together with a random choice if action stay is taken. Consequently, a simulation starts from s_{init} and ends at one of the terminal states ($\{s_+, s_-, s_?\}$). During simulation, we enhance our estimate of $\mathcal{M}'$ by visiting new states, exploring new actions and improving our estimate of $\mathbb{T}'$ with more samples. When states are visited for the first time, actions are chosen at random, and subsequently, actions with a higher potential reward are chosen. If a simulation is stuck in a loop, we check for the presence of an MEC with δ_{TP} confidence. If a δ_{TP}-sure MEC is found, we add a stay action with $l, u = 0, 1$, otherwise we keep simulating until the required confidence is achieved. After that, we take the action with the highest upper bound that is leaving the MEC to continue the simulation. We do several such simulations to build a large enough model before doing value iteration in the next steps.

Estimating Transition Probabilities. [3] gives an analysis to estimate bounds on transition probabilities for reachability objective in MDPs. For completeness, we briefly restate it here. Given an MP-inconfidence δ_{MP}, we distribute the inconfidence over all individual transitions as

$$\delta_{TP} := \frac{\delta_{MP} \cdot p_{\mathsf{min}}}{|\{a | \mathsf{s} \in \mathsf{S}' \wedge a \in \mathsf{Av}'(s)\}|},$$

where $\frac{1}{p_{\mathsf{min}}}$ gives an upper bound on the maximum number of possible successors for an available action from a state[7]. The Hoeffding's inequality gives us a bound on the number of times an action a needs to be sampled from state s, denoted $\#(s, a)$, to achieve a TP-imprecision $\varepsilon_{TP} \geqslant \sqrt{\frac{\ln \delta_{TP}}{-2\#(s,a)}}$ on $\mathbb{T}(s, a, t)$, such that

$$\widehat{\mathbb{T}}(s, a, t) := \max(0, \frac{\#(s, a, t)}{\#(s, a)} - \varepsilon_{TP})$$

[7] Knowing additionally $\max_{s \in \mathsf{S}, a \in \mathsf{Av}(s)} |\mathsf{Post}(s, a)|$ gives slightly smaller TP-imprecision. See Appendix G.4 in [1].

Algorithm 1. Mean-payoff learning for black-box MDP

Input: MDP $\mathcal{M}$, imprecision $\varepsilon_{MP} > 0$, MP-inconfidence $\delta_{MP} > 0$, lower bound $p_{\min}$ on transition probabilities in $\mathcal{M}$
Parameters: revisit threshold $k \geq 2$, episode length $n \geq 1$
Output: upon termination ε_{MP}-precise estimate of the maximum mean payoff for $\mathcal{M}$ with confidence $1 - \delta_{MP}$, i.e. $(\varepsilon_{MP}, 1 - \delta_{MP})$-PAC estimate

procedure ON_DEMAND_BVI
  //Initialization
  Set $\mathsf{L}(s_+) = \mathsf{U}(s_+) = \mathsf{U}(s_?) = 1$, $\mathsf{L}(s_-) = \mathsf{U}(s_-) = \mathsf{L}(s_?) = 0$ ▷ Augmentation
  $\mathsf{S}' = \emptyset$ ▷ States of learnt model
  repeat
    //Get n simulation runs and update MP of MECs where they end up
    for n times **do**
      $w \leftarrow \mathsf{SIMULATE}(k)$ ▷ Path taken by the simulation
      $\mathsf{S}' \leftarrow \mathsf{S}' \cup w$ ▷ Add states to the model
      $\delta_{TP} \leftarrow \frac{\delta_{MP} \cdot p_{\min}}{|\{a \mid s \in \mathsf{S}' \wedge a \in \mathsf{Av}'(s)\}|}$ ▷ Split inconfidence among all transitions
      if last state of w is s_+ or $s_?$ **then** ▷ Probably entered a good MEC M
        $M \leftarrow$ MEC from which we entered the last state of w
        $\mathsf{UPDATE\_MEC\_VALUE}(M)$ ▷ Increase precision using more VI
        Update $\mathbb{T}'(s, \mathsf{stay})$ according to Definition 4 for all $s \in M$
    //Identify δ_{TP}-sure MECs and propagate their MP by VI for reachability
    ProbableMECs $\leftarrow \mathsf{FIND\_MECS}$ ▷ δ_{TP}-sure MECs
    $\mathsf{INITIALIZE\_VI\_BOUNDS}$ ▷ Reinitialize L, U for all states
    repeat
      $\mathsf{UPDATE}(\mathsf{S}')$ ▷ One Bellman update per state
      for $T \in$ *ProbableMECs* **do**
        $\mathsf{DEFLATE}(T)$ ▷ Ensure safe but converging U
    until L and U close to their respective fixpoints
  until $\mathsf{U}(s_{\mathsf{init}})$ - $\mathsf{L}(s_{\mathsf{init}}) < \frac{2\varepsilon_{MP}}{r_{\max}}$ ▷ ε_{MP} is the absolute error; we use "$< \frac{2\varepsilon_{MP}}{r_{\max}}$" for relative difference between upper and lower values, where $r_{\max} = \max_{s \in \mathsf{S}'} r(s)$.

where, $\#(s, a, t)$ is the number of times t is sampled when action a is chosen in s.

Updating mean-payoff values Using $\widehat{\mathbb{T}}(s, a, t)$, we compute estimates of the upper and lower bounds of the values corresponding to every action from a state visited in the partial model that is constructed so far. We use the following *modified* Bellman Eq. [3]:

$$\widehat{\mathsf{L}}(s,a) := \sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t) \cdot \mathsf{L}(t)$$

$$\widehat{\mathsf{U}}(s,a) := \sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t) \cdot \mathsf{U}(t) + \Big(1 - \sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t)\Big),$$

where $\mathsf{L}(t) = \max_{a \in \mathsf{Av}(t)} \widehat{\mathsf{L}}(t,a)$ and $\mathsf{U}(t) = \max_{a \in \mathsf{Av}(t)} \widehat{\mathsf{U}}(t,a)$ are bounds on the value of from a state, $v(s)$. When a state is discovered for the first time during the

simulation, and is added to the partial model, we initialize $\mathsf{L}(s)$, and $\mathsf{U}(s)$ to 0, and 1, respectively. Note that $\sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t) < 1$. We attribute the remaining probability to unseen successors and assume their value to be 0 (1) to safely under-(over-)approximate the lower (upper) bounds. We call these *blackbox Bellman update* equations, since it assumes that all the successors of a state-action pair may not have been visited.

Estimating Values of End-Components. End-components are identified with an inconfidence of δ_{TP}. As observed in [13], assuming an action has been sampled n times, the probability of missing a transition for that action is at most $(1 - p_{\mathsf{min}})^n$. Thus, for identifying (T, A) as a δ_{TP}*-sure* MEC, every action in A that is available from a state $s \in T$ needs to be sampled at least $\frac{\ln \delta_{TP}}{\ln(1-p_{\mathsf{min}})}$ times.

Once a δ_{TP}*-sure* MEC M is identified, we estimate its upper (v^u_M) and lower (v^l_M) bounds using value iteration.[8] While running value iteration, we assume, with a small inconfidence, that there are no unseen outgoing transitions. So we use the following modified Bellman update equations inside the MEC where we under-(over-)approximate the lower(upper) bound to a much lesser degree.

$$\widehat{\mathsf{L}}(s,a) := \sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t) \cdot \mathsf{L}(t) + \min_{t:\#(s,a,t)>0} \mathsf{L}(t) \cdot (1 - \sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t))$$

$$\widehat{\mathsf{U}}(s,a) := \sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t) \cdot \mathsf{U}(t) + \max_{t:\#(s,a,t)>0} \mathsf{U}(t) \cdot (1 - \sum_{t:\#(s,a,t)>0} \widehat{\mathbb{T}}(s,a,t))$$

Following the assumption, we call these *greybox* (See Definition 3) Bellman update equations. The value iteration algorithm further gives us bounds on v^u_M and v^l_M. We say that the upper estimate of v^u_M ($\widehat{v}^u_M$) and the lower estimate of v^l_M ($\widehat{v}^l_M$) are the overall upper and lower bounds of the mean-payoff value of M, respectively. To converge the overall bounds, we need value iteration to return more precise estimates of v^l_M and v^u_M, and we need to sample the actions inside M many times to reduce the difference between v^l_M and v^u_M. We call this procedure, UPDATE_MEC_VALUE.

Now, some MECs may have very low values or may not be reachable from s_{init} with high probability. In such cases, no optimal strategy may visit these MECs, and it might not be efficient to obtain very precise mean-payoff values for every MEC that is identified in an MDP. We follow the on-demand heuristic [2] where we progressively increase the precision on mean-payoff values as an MEC seems more likely to be a part of an optimal strategy. The stay action on MECs helps in guiding simulation towards those MECs that have a higher lower bound of the mean-payoff value. In particular, whenever the simulation ends up in s_+ or $s_?$, we run UPDATE_MEC_VALUE with higher precision on the MEC that led to these states. If the simulation ends up in these states through a particular MEC more often, it indicates that the MEC is likely to be a part of an optimal strategy, and it would be worth increasing the precision on its mean-payoff value.

[8] Note that one requires the ECs to be aperiodic for the VI to converge. [30] suggests a way that deals with this.

Deflate Operation. Unlike in the case of computation of mean payoff for whitebox models [3] where a MEC is collapsed following the computation of its value, for blackbox learning, once a set of states is identified as a δ_{TP}-*sure* MEC, we cannot collapse them. This is because collapsing would prevent a proper future analysis of those states, which is undesirable in a blackbox setting. However, this leads to other problems. To illustrate this, we consider an MDP that only has a single MEC M and one outgoing action from every individual state. Recall from Eq. 2 that we compute the mean-payoff by reducing it to a reachability problem. Once the mean-payoff for the MEC, and the probabilities corresponding to stay action in Line 12 are computed, to compute the reachability probability, the upper and lower bounds of all states in the MECs are initialized to 1 and 0 respectively. Now suppose that the sum of probabilities to s_+ and $s_?$ be p denoting the upper bound on the value of the mean-payoff to be $p \cdot r_{\max}$. Clearly, the upper bound on the reachability value of this MDP is p. Now, when we do BVI to calculate this value, from every state in M, there would be *at least* two action choices, one that stays inside the MEC, and one that corresponds to the stay action. Initially, all states, except the terminal states, would have upper and lower values set to 0 and 1, respectively. Thus, among the two action choices, one would have upper value p, while the other would have upper value 1, and hence, the Bellman update assigns the upper value of the state to 1. As one can see, this would go on, and convergence wouldn't happen, and hence the true mean-payoff value will not be propagated to the initial state of the MDP. To avoid this, we need the deflate operation which lowers the upper reachability value to the best outgoing action, i.e. in this case, the stay action with value p.

Statistical Guarantees. The following theorem shows that the mean-payoff value learnt by Algorithm 1 is PAC on an input blackbox MDP.

Theorem 1. *Algorithm 1 has the property that when it stops, it returns an interval for the mean-payoff value of the MDP that is PAC for the given MP-inconfidence δ_{MP} and the MP-imprecision ε_{MP}.*

Anytime Algorithm. As a direct consequence, we obtain an anytime algorithm from Algorithm 1 by (1) dropping the termination test on Line 20, i.e. replacing it with **until false**, and (2) upon query (or termination) by the user, we output $(\mathsf{U}(s_{\mathsf{init}}) + \mathsf{L}(s_{\mathsf{init}}))/2$ as the estimate and, additionally, we output $(\mathsf{U}(s_{\mathsf{init}}) - \mathsf{L}(s_{\mathsf{init}}))/2$ as the current imprecision.

Using Greybox Update Equations During Blackbox Learning. We also consider the variant where we use greybox update equations to estimate the mean-payoff values. However, assuming we keep the TP-imprecision unchanged, the overall TP-inconfidence now has to include the probability of missing some successor of a state s for an action a[9]. Given a number of samples $\#(s,a)$, the probability that we miss a particular successor is at most $(1 - p_{\mathsf{min}})^{\#(s,a)}$, and hence the

[9] Assuming $\#(s,a)$ to be as small as 200, and $p_{\mathsf{min}} = 0.05$, the probability of missing a transition is $3.5 \cdot 10^{-5}$.

overall TP-inconfidence corresponding to using greybox equations for blackbox learning increases to $\delta_{TP} + (1 - p_{\mathsf{min}})^{\#(s,a)}$.

We also note that the use of greybox update equations on estimating the transition probabilities also gives us a PAC guarantee but with an increased MP-Inconfidence resulting from an increased TP-inconfidence.

5 Algorithm for Continuous-Time MDP

In this section, we describe an algorithm to learn blackbox CTMDP models for mean-payoff objective while respecting the PAC guarantees. As in the case of MDPs, we reduce the mean-payoff problem to a reachability problem. We follow the same overall framework as in MDPs, where we compute the probability to reach the end-components under an optimal strategy, and we compute their respective mean-payoff values. Computing reachability probabilities in a CTMDP is the same as computing reachability probabilities in the underlying embedded MDP. Similar to estimating $\mathbb{T}(s, a, t)$ in Sect. 4 for MDPs, we estimate $\Delta(s, a, t)$[10] for CTMDPs, and follow the simulation-based procedure in Algorithm 1 to compute reachability probabilities. However, unlike MECs in MDPs, where the mean-payoff value depends solely on the transition probabilities, the mean-payoff value in a CTMDP also depends on the rates $\lambda(s, a)$ for $s \in T$ and $a \in A(s)$ for an MEC $M = (T, A)$. Thus to compute the value of an MEC, we also estimate the rates of the state-action pairs. Once we get the estimates of the rates, we uniformize the CTMDP to obtain a uniform CTMDP that can be treated as an MDP by disregarding the rates while preserving the mean-payoff value [30]. Detailed pseudocode of the algorithms for this section are provided in Appendix F of [1].

Estimating Rates. Recall that for an action a, the time spent in s is exponentially distributed with a parameter $\lambda(s, a)$, and $\frac{1}{\lambda(s,a)}$ is the mean of this distribution. During the simulation of a CTMDP, for every state s reached and action a chosen from s, we construct a sequence $\tau_{s,a}$ of the time difference between the entry and the corresponding exit from s when action a is chosen. Then, the average over the sequence $\tau_{s,a}$ gives us an estimate $\frac{1}{\widehat{\lambda}(s,a)}$ of $\frac{1}{\lambda(s,a)}$ (Abbreviated to $\frac{1}{\lambda}$ from now on when (s, a) is clear from the context.).

Assuming a multiplicative error α_R on our estimates of $\frac{1}{\lambda}$, the lemma below uses Chernoff bounds[11] to give the number of samples that need to be collected from an exponential distribution so that the estimated mean $\frac{1}{\widehat{\lambda}}$ is at most α_R-fraction away from the actual mean $\frac{1}{\lambda}$ with probability at least $1 - \delta_R$, where $\alpha_R, \delta_R \in (0, 1)$. Further by Cramer's theorem [15], it follows that this is the tightest possible bound for the number of samples collected.

[10] Recall that an estimate of $\Delta(s, a, t)$ is the ratio between $\#(s, a, t)$ and $\#(s, a)$, and is the probability with which we go to state t from s when action a is chosen from s.

[11] Since λ is not bounded, we cannot use Hoeffding's inequality as in the case of estimating the transition probabilities.

Lemma 1. *Let $X_1, \ldots, X_n$ be exponentially distributed i.i.d. random variables with mean $\frac{1}{\lambda}$. Then we have that*

$$\mathbb{P}\Big[\Big|\frac{1}{\widehat{\lambda}} - \frac{1}{\lambda} \geqslant \frac{1}{\lambda} \cdot \alpha_R)\Big|\Big] \leqslant \inf_{-\lambda<t<0}\Big(\frac{\lambda}{\lambda+t}\Big)^n \cdot e^{\frac{tn}{\lambda}(1+\alpha_R)} + \inf_{t>0}\Big(\frac{\lambda}{\lambda+t}\Big)^n \cdot e^{\frac{tn}{\lambda}(1-\alpha_R)},$$

where $\frac{1}{n}\sum_{i=1}^{n} X_i = \frac{1}{\widehat{\lambda}}$.

Assuming the right-side of the inequality is at most δ_R, we have that $\lambda \in [\hat{\lambda}(1-\alpha_R), \hat{\lambda}(1+\alpha_R)]$, or $\widehat{\lambda} \in [\frac{\lambda}{1+\alpha_R}, \frac{\lambda}{1-\alpha_R}]$ with probability at least $1-\delta_R$. Table 1 shows the number of samples required for various values of α_R and δ_R[12].

Table 1. Lookup table for number of samples based on α_R and δ_R

$\alpha_R \setminus \delta_R$	10%	5%	0.01%	0.00001%
3%	7000	9000	23000	60000
5%	2500	3100	8000	13400

Given a maximum multiplicative error α_R on the mean of the exponential distributions of the state-action pairs in a CTMDP, we say that the rate λ is known α_R*-precisely* if $\widehat{\lambda} \in [\frac{\lambda}{1+\alpha_R}, \frac{\lambda}{1-\alpha_R}]$. We now quantify the bounds on the estimated mean-payoff value. Let $\mathcal{M}$ be a CTMDP, $v_{\mathcal{M}}$ be its actual mean-payoff value, and let $\widehat{v}_{\mathcal{M}}$ denote its mean-payoff when the rates of the state-action pairs are known α_R-precisely. Then we have the following.

Lemma 2. *Given a CTMDP $\mathcal{M}$ with rates known α_R-precisely, with transition probabilities known precisely, and with maximum reward per unit time over all states r_{max}, we have $v_{\mathcal{M}}(\frac{1-\alpha_R}{1+\alpha_R}) \leq \widehat{v}_{\mathcal{M}} \leq v_{\mathcal{M}}(\frac{1+\alpha_R}{1-\alpha_R})$ and $|\widehat{v}_{\mathcal{M}} - v_{\mathcal{M}}| \leq r_{max}\frac{2\alpha_R}{1-\alpha_R}$.*

Estimating Mean-Payoff Values of MECs. Using our bounds on the rates of the transitions, we now compute bounds on the mean-payoff values of MECs in CTMDPs. We first show that the mean payoff is maximized or minimized at the boundaries of the estimates of the rates. Intuitively, to maximise the mean-payoff value, for a state s_i with a high reward, we would like to maximise the time spent in s_i or equivalently, minimise the rate $\lambda(s_i, a)$ for every outgoing action a from s_i. We do the opposite when we want to find a lower bound on the mean-payoff value in the MEC. Consider an MEC M having states $T = \{s_1, ..., s_m\}$. Assume that λ_i is the rate of an action a from state s_i, such that a positional mean-payoff maximizing strategy σ chooses a from s_i. Then, the expected mean-payoff value of M is given by,

[12] In Appendix E of [1], we show the computation of the number of samples for one of the entries.

$$v_M = \frac{\sum\limits_{s_i \in T} \frac{r(s_i)\pi_i}{\lambda_i}}{\sum\limits_{s_i \in T} \frac{\pi_i}{\lambda_i}}, \tag{3}$$

where π_i denotes the expected fraction of total time spent in s_i under σ.

Now, we have estimates $\frac{1}{\widehat{\lambda}_i}$ of $\frac{1}{\lambda}$, such that, $\lambda_i \in \left[\widehat{\lambda}_i\left(1-\alpha_R\right), \widehat{\lambda}_i\left(1+\alpha_R\right)\right]$ with high probability. Let $\lambda_i^l = \widehat{\lambda}_i\left(1-\alpha_R\right)$ and $\lambda_i^u = \widehat{\lambda}_i\left(1+\alpha_R\right)$.

Proposition 1. *In Eq. 3, the maximum and the minimum values of v_M occur at the boundaries of the estimates of λ_i for each $1 \leqslant i \leqslant m$.*

In particular, v_M is maximized when,

$$\lambda_i = \begin{cases} \lambda_i^l, & \text{if } r(s_i) \geq v_M \\ \lambda_i^u, & \text{otherwise} \end{cases} \tag{4}$$

Once we fix the rates for each of the states in M, we uniformize M to obtain a uniform CTMDP M_C which is an MEC and can be treated as an MDP for computing its mean-payoff value [30]. Let for a state-action pair, the rate be $\lambda(s,a)$, and the uniformization constant be C. For a successor t from s under action a such that $t \neq s$, we have $\Delta(s,a,t) = \frac{\#(s,a,t)}{\#(s,a)} \cdot \frac{\lambda(s,a)}{C}$, and $\Delta(s,a,s) = 1 - \sum\limits_{t \neq s} \Delta(s,a,t)$. Finally, value iteration on M_C with appropriate confidence width gives us the lower and the upper estimates of the mean-payoff value of the MEC M.

We now describe an iterative procedure to identify those states of the MEC for which the upper bound on the estimates of the rates are assigned, and those states for which the lower bound on the estimates of the rates are assigned in order to maximize or minimize the mean-payoff value of the MEC. Assume w.l.o.g. that the states $s_1, \ldots, s_m$ are sorted in decreasing order of their rewards $r(s_i)$. In iteration j, we set $\lambda_i = \lambda_i^l$ for $1 \leqslant i \leqslant j$, and we set $\lambda_i = \lambda_i^u$ for the remaining states and recompute v_M. The maximum value of v_M across all iterations gives the upper bound on v_M. Similarly we can find the lower bound on v_M. Overall, value iteration is done $2|T|$ times[13].

Overall Algorithm. As stated in the beginning of this section, an algorithm for computing the mean payoff in blackbox CTMDP models largely follows the same overall framework as stated in Sect. 4. By sampling the actions, we obtain estimates of the rates and the transition probabilities. The reachability probabilities

[13] In our experiments, we use a heuristic to estimate v_M that provides good approximate bounds and is more efficient. We first compute an initial estimate of $\widehat{v}_M$ using our current estimates, $\widehat{\lambda}$. We then compute the upper bound by assigning the rates as in Eq. 4 where v_M is replaced with $\widehat{v}_M$. Similarly, the lower bound can also be found. A detailed pseudocode of this algorithm is described in Algorithm 18 of [1].

to the MECs of the CTMDP are estimated using the estimates of the transition probabilities while the mean-payoff values of MECs are estimated using uniformization as decribed above. The confidence widths on the transition probabilities in a uniformized MEC are assigned based on the number of samples $\#(s,a)$ for a state-action pair (s,a).

Statistical Guarantees. Let δ_{TP} and δ_R be the TP-inconfidence and the inconfidence on individual transition rates, respectively. Further, let δ_{MP1} and δ_{MP2} be the overall inconfidence on the transition probabilities and transition rates, respectively. Then, $\delta_{TP} := \frac{\delta_{MP1} \cdot p_{\mathsf{min}}}{|\{a|\, \mathsf{s} \in \widehat{S} \wedge a \in \mathsf{Av}(s)\}|}$, and $\delta_R := \frac{\delta_{MP2}}{|\{a|\, \mathsf{s} \in \widehat{S} \wedge a \in \mathsf{Av}(s)\}|}$. Thus, we have that the overall inconfidence on the mean-payoff value, $\delta_{MP} = \delta_{MP1} + \delta_{MP2}$. Thus, to achieve a given inconfidence on the mean-payoff value, we fix δ_{TP} and δ_R, and adjust the imprecisions ε_{TP} and α_R accordingly.[14]

As in the case of MDPs, our learning algorithm for blackbox CTMDP models is an anytime algorithm that is PAC for the given MP-inconfidence δ_{MP}.

6 Experimental Results

We implemented our algorithms as an extension of Prism [27] and tested it on 15 MDP benchmarks and 10 CTMDP benchmarks. Several of these benchmarks were selected from the Quantitative Verification Benchmark Set [21][15]. The results for MDP and CTMDP blackbox learning are shown in Table 2 and Table 3 respectively. Here, we scale the upper and lower bounds to 1 and 0, and show the average values taken over 10 experiments. The experiments were run on a desktop machine with an Intel *i*5 3.2 GHz quad core processor and 16 GB RAM. The MP-imprecision ε_{MP} is set to 10^{-2}, revisitThreshold k is set to 6, MP-inconfidence δ_{MP} is set to 0.1 and n is set to 10000. We further use a timeout of 30 minutes. In the case of a timeout, the reported upper and lower bounds on the mean payoff still correspond to the input MP-inconfidence δ_{MP}, although the MP-imprecision may not be the desired one.

Blackbox Learning for MDPs. We see that in Table 2 for blackbox learning, 9 out of 15 benchmarks converge well, such that the precision is within 0.1. In fact, for many of these 9 benchmarks, a precision of 0.1 is achieved much before the timeout (TO). In Fig. 1a and Fig. 1b, we show this for zeroconf and pacman. zeroconf has a large transient part and a lot of easily reachable single state

[14] See Appendix G of [1] for a more detailed calculation of the number of samples required to make transition probabilities and the rates precise.

[15] The CTMDP benchmarks are available as Markov automata models that were converted to CTMDP models using a tool developed in the thesis [11].

Table 2. Results on MDP benchmarks.

Benchmarks	Number of states[a]	Value	Blackbox				Blackbox with greybox update equations			
			States explored	Lower bound	Upper bound	Time (s)	States explored	Lower bound	Upper bound	Time (s)
virus	809	0	809	0.0	0.5319	TO	809	0.0	0.008	273.01
cs_nfail	184	0.333	184	0.3275	0.3618	TO	184	0.332	0.337	126.77
investor	6688	0.95	6284	0.8458	0.9559	TO	5835	0.945	0.954	620.23
zeroconf	3001911	TO	487	0.923	1.0	TO	360	0.990	1.0	116.04
sensors	189	0.333	189	0.3299	0.3513	TO	189	0.332	0.336	64.64
consensus	272	0.1083	272	0.093	0.1605	TO	272	0.103	0.113	190.32
ij10	1023	1	1023	0.3626	1.0	TO	1023	0.999	1.0	26.822
ij3	7	1	7	0.990	1.0	15.92	7	0.999	1.0	0.7127
pacman	498	0.5511	496	0.5356	0.5754	TO	496	0.5477	0.5577	215.36
wlan	2954	1	2954	0.6577	1.0	TO	2935	1.0	1.0	16.924
blackjack	3829	0	3829	0.0	0.3014	TO	3829	0.0	0.006	91.503
counter	8	0.5	8	0.4998	0.5	30.37	8	0.4999	0.5	15.215
recycling	5	0.727	5	0.726	0.727	1.309	5	0.726	0.727	0.927
busyRing	1912	1	1733	0.706	1.0	TO	1542	0.999	1.0	34.86
busyRingMC	2592	1	2574	0.969	1.0	TO	2507	0.999	1.0	114.50

[a] The number of states and the values are computed using the probabilistic model-checker Storm [14]
The number of states and the true mean-payoff values are computed by first uniformizing the CTMDP, and then using Storm on the underlying MDP.

MECs. Since it has a true value of 1, the upper and the lower values converge after exploring only a few MECs. Our algorithm only needed to explore a very small percentage of the states to attain the input precision. cs_nfail has many significant MECs, and the learning algorithm needs to explore each of these MECs, while in sensor there is a relatively large MEC of around 30 states, and the simulation inside this MEC takes considerable amount of time.

virus consists of a single large MEC of more than 800 states, and its true value is 0. As we simulate the MEC more and more, the TP-imprecision on the transition probabilities decreases and the upper bound on the mean-payoff reduces over time. ij10 contains one MEC with 10 states in it. The value converges faster and reaches a value of 1, during blackbox learning. This model has relatively high number of actions, more than 5, for many of its states outside the MEC. This leads to a higher TP-imprecision. Further, due to the conservative nature of the blackbox update equations, the upper and the lower values converge very slowly.

consensus, ij10, ij3, pacman, wlan were used in [3] for learning policies for reachability objectives. The target states in these benchmarks are sink states with self loops, and we add a reward of 1 on these target states so that the rechability probability becomes the same as the mean payoff. The mean-payoff

Table 3. Results on CTMDP benchmarks

Benchmarks	Number of states	Value	Blackbox				Blackbox with greybox update equations			
			States explored	lower bound	upper bound	Time (s)	States explored	lower bound	upper bound	Time (s)
DynamicPM	816	1.0	816	0.436	1.0	TO	816	0.998	1.0	37.68
ErlangStages	508	1.0	508	0.962	1.0	TO	508	0.999	1.0	8.118
PollingSystem1	16	0.922	16	0.811	0.937	TO	16	0.816	0.937	TO
PollingSystem2	348	0.999	348	0.637	0.999	TO	348	0.998	0.999	21.893
PollingSystem3	1002	0.999	1002	0.232	1.0	TO	1002	0.99	1.0	864.05
QueuingSystem	266	0.8783	266	0.703	0.906	TO	266	0.865	0.886	TO
SJS1	17	1.0	17	0.999	1.0	133.96	17	0.997	1.0	1.05
SJS2	7393	0.999	7341	0.02	1.0	TO	7268	0.936	1.0	TO
SJS3	433	1.0	433	0.919	1.0	TO	432	0.999	0.999	5.3814
toy	12	1.0	12	0.99	1.0	5.6	12	0.999	1.0	1.112

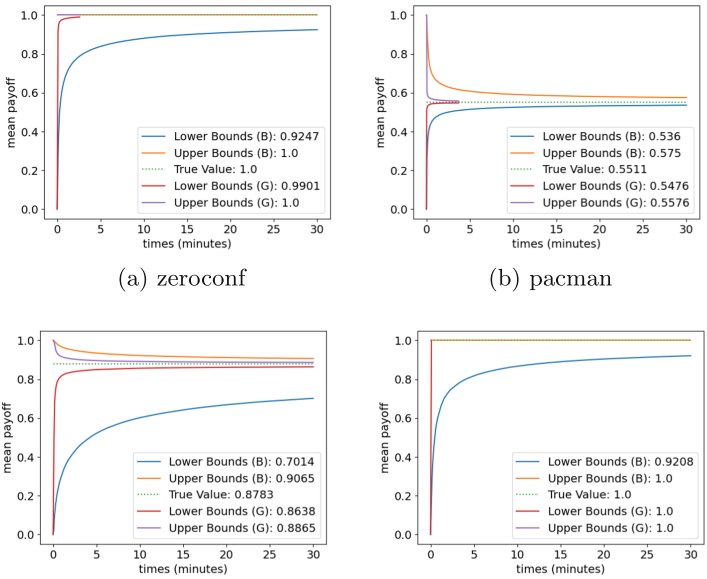
results we observe are similar to the bounds reported for reachability probability in [3], and our experiments also take similar time as reported in [3].

The blackjack model [35] is similar to zeroconf model. It has 3829 states and 2116 MECs. It has a large transient part and a lot of single state MECs. However, unlike zeroconf all of the MECs have a value of 0. Thus, simulation takes more time as the TP-imprecision reduces slowly.

Blackbox Learning with Greybox Update Equations. We show the results of these experiments in the right side of Table 2. As observed, convergence is much faster here for all the benchmarks. All our benchmarks converged correctly within a few seconds to a few minutes. Hence for a small degradation in MP-inconfidence use of greybox update equations works well in practice. We show the effect on MP-inconfidence in more detail in Table 8 in Appendix G of [1].

Blackbox Learning for CTMDPs. In Table 3 we show the results for CTMDP benchmarks. The number of states in these benchmarks vary from as low as 12 to more than 7000. All the models used here have a lot of small end-components. We observe that the upper and the lower values take more time to converge as the size of the model grows. Figure 1c and Fig. 1d show the convergence of lower and upper bounds for QueuingSystem and SJS3. As in the case of MDPs, using *greybox update equations* speeds up the learning process significantly.

Greybox Learning. Recall from Definition 3 that in greybox learning, for every state-action pair, we know the number of successors of the state for the given action. As expected, their convergence is much faster than that for blackbox learning, but the convergence is comparable to the case where we do blackbox learning with greybox update equations. The details of the greybox learning experiments can be found in Appendix G of [1].



(a) zeroconf (b) pacman

(c) QueuingSystem (d) SJS3

Fig. 1. Convergence of lower and upper bounds for blackbox update equations and greybox update equations.

7 Conclusion

We presented the first PAC SMC algorithm for computing mean payoff in unknown MDPs and CTMDPs, where the only information needed is a lower bound on minimum transition probability, as advocated in [13]. In contrast to a naive algorithm, which follows in a quite straightforward way from the literature, our algorithm is practically applicable, overcoming the astronomic number of simulation steps required. To this end, in particular, the inconfidence had to be distributed in non-uniformly over the transitions and then imprecision propagated by value iteration with precision guarantees. In future, we would like to thoroughly analyse how well weakening the PAC bounds can be traded for a yet faster convergence. On the practical side, applying importance sampling and importance splitting could further improve the efficiency.

Acknowledgements. The second author would like to thank Subhajit Goswami for insightful discussions on learning transition rate matrix in a CTMDP and for pointing to useful references.

References

1. Agarwal, C., Guha, S., Pazhamalai, M., Křetínský, J.: Pac statistical model checking of mean payoff in discrete- and continuous-time mdp (2022). CoRR, abs/2206.01465

2. Ashok, P., Chatterjee, K., Daca, P., Křetínský, J., Meggendorfer, T.: Value iteration for long-run average reward in markov decision processes. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 201–221. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_10
3. Ashok, P., Křetínský, J., Weininger, M.: PAC statistical model checking for markov decision processes and stochastic games. In: Dillig, I., Tasiran, S. (eds.) CAV 2019. LNCS, vol. 11561, pp. 497–519. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-25540-4_29
4. Auer, P., Ortner, R.: Logarithmic online regret bounds for undiscounted reinforcement learning. In: NIPS, pp. 49–56. MIT Press (2006)
5. Baier, C., Haverkort, B.R., Hermanns, H., Katoen, J.-P.: Performance evaluation and model checking join forces. Commun. ACM **53**(9), 76–85 (2010)
6. Baier, C., Katoen, J-P.: Principles of Model Checking. MIT Press (2008)
7. Bertsekas, D.P.: Dynamic Programming and Optimal Control, vol. II. Athena Scientific (1995)
8. Brázdil, T., Brožek, V., Chatterjee, K., Forejt, V., Kučera, A.: Two views on multiple mean-payoff objectives in Markov decision processes. LMCS **10**(1), 1–29 (2014)
9. Brázdil, T., et al.: Verification of markov decision processes using learning algorithms. In: Cassez, F., Raskin, J.-F. (eds.) ATVA 2014. LNCS, vol. 8837, pp. 98–114. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11936-6_8
10. Bruno, J.L., Downey, P.J., Frederickson, G.N.: Sequencing tasks with exponential service times to minimize the expected flow time or makespan. J. ACM **28**(1), 100–113 (1981)
11. Butkova, Y.: Towards efficient analysis of Markov automata. PhD thesis, Saarland University, Saarbrücken, Germany (2020)
12. Chatterjee, K.: Robustness of structurally equivalent concurrent parity games. In: FOSSACS, pp. 270–285 (2012)
13. Daca, P., Henzinger, T.A., Křetínský, J., Petrov, T.: Faster statistical model checking for unbounded temporal properties. In: Chechik, M., Raskin, J.-F. (eds.) TACAS 2016. LNCS, vol. 9636, pp. 112–129. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49674-9_7
14. 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
15. Dembo, A., Zeitouni, O.: Large deviations techniques and applications. Springer, Cham (2010). https://doi.org/10.1007/978-3-642-03311-7
16. Feinberg, E.A.: Continuous time discounted jump markov decision processes: a discrete-event approach. Math. Oper. Res. **29**(3), 492–524 (2004)
17. Feinberg, E.A., Shwartz, A.: Handbook of Markov decision processes: methods and applications, volume 40. Springer Science & Business Media, New York (2012). https://doi.org/10.1007/978-1-4615-0805-2
18. Fu, J., Topcu, U.: Probably approximately correct MDP learning and control with temporal logic constraints. Science and Systems, In Robotics (2014)
19. Ghemawat, S., Gobioff, H., Leung, S.: The google file system. In: SOSP (2003)
20. Hahn, E.M., Perez, M., Schewe, S., Somenzi, F., Trivedi, A., Wojtczak, D.: Omega-regular objectives in model-free reinforcement learning. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 395–412. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_27

21. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The quantitative verification benchmark set. In: Vojnar, T., Zhang, L. (eds.) TACAS 2019. LNCS, vol. 11427, pp. 344–350. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-17462-0_20
22. Haverkort, B.R., Hermanns, H., Katoen, J-P.: On the use of model checking techniques for dependability evaluation. In: SRDS 2000 (2000)
23. Henriques, D., Martins, J.G., Zuliani, P., Platzer, A., Clarke, E.M.: Statistical model checking for markov decision processes. In: QEST, pp. 84–93. IEEE Computer Society (2012)
24. Kelmendi, E., Krämer, J., Křetínský, J., Weininger, M.: Value iteration for simple stochastic games: stopping criterion and learning algorithm. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10981, pp. 623–642. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_36
25. J. Kretínský, Michel, F., Michel, L., Pérez, G.A.: Finite-memory near-optimal learning for markov decision processes with long-run average reward. In: UAI of Proceedings of Machine Learning Research, vol. 124, pp. 1149–1158. AUAI Press (2020)
26. Křetínský, J., Pérez, G.A., Raskin, J.-F.: Learning-based mean-payoff optimization in an unknown MDP under omega-regular constraints. In: CONCUR, Dagstuhl, pp. 8:1–8:18 (2018)
27. Kwiatkowska, M., Norman, G., Parker, D.: PRISM: probabilistic symbolic model checker. In: Field, T., Harrison, P.G., Bradley, J., Harder, U. (eds.) TOOLS 2002. LNCS, vol. 2324, pp. 200–204. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-46029-2_13
28. Kwiatkowska, M.Z., Norman, G., Parker, D.: The PRISM benchmark suite. In: QEST, pp. 203–204. IEEE Computer Society (2012)
29. Lassaigne, R., Peyronnet, S.: Approximate planning and verification for large Markov decision processes. In: SAC, pp. 1314–1319. ACM (2012)
30. Puterman, M.L.: Markov decision processes: Discrete stochastic dynamic programming. John Wiley and Sons (1994)
31. Qiu, Q., Qu, Q., Pedram, M.: Stochastic modeling of a power-managed system-construction and optimization. IEEE Trans. CAD Integrated Circuits Syst. **20**(10), 1200–1217 (2001)
32. Sennott, L.I.: Stochastic Dynamic Programming and the Control of Queueing Systems. Wiley-Interscience, New York (1999)
33. Solan, E.: Continuity of the value of competitive markov decision processes. J. Theor. Probab. **16**, 831–845 (2003)
34. Strehl, A.L., Li, L., Wiewiora, E., Langford, J., Littman, M.L.: PAC model-free reinforcement learning. In: ICML, pp. 881–888. ACM (2006)
35. Sutton, R.S., Barto, A.G.: Reinforcement learning - an introduction. Adaptive computation and machine learning. MIT Press (1998)
36. Ortner, R., Jaksch, T., Auer, P.: Near-optimal regret bounds for reinforcement learning. J. Mach. Learn. Res. **11**, 1563–1600 (2010)
37. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, pp. 223–235. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45657-0_17