DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY MUNICH

Master's Thesis in Informatics

Solving the quantitative Reachability Problem on Markov Decision Processes using Learning Algorithms

Tobias Meggendorfer

DEPARTMENT OF INFORMATICS

TECHNICAL UNIVERSITY MUNICH

Master's Thesis in Informatics

Solving the quantitative Reachability Problem on Markov Decision Processes using Learning Algorithms

Lösen des quantitativen Erreichbarkeitsproblems auf Markov Entscheidungsprozessen mittels Learning-Algorithmen

Author:	Tobias Meggendorfer
Supervisor:	Prof. Dr. Jan Křetínský
Advisor:	Prof. Dr. Jan Křetínský
Submission Date:	15.08.2020

I confirm that this master's thesis is my own work and I have documented all sources and material used.

Ergolding, 12.08.2020 Tobias Meggendorfer

Abstract

In this work, we present a general framework for applying machine-learning algorithms to the verification of Markov decision processes (MDPs), based on the ideas of [Brá+14]. The primary goal of the techniques presented in [Brá+14] is to improve performance by avoiding an exhaustive exploration of the state space and instead rely on guidance by heuristics. This approach is significantly extended in this work. Several details of [Brá+14] are refined and errors are fixed.

The presented framework focuses on probabilistic reachability, which is a core property for verification, and is illustrated through two distinct instantiations. The first assumes that full knowledge of the MDP is available, in particular precise transition probabilities. It performs a heuristic-driven partial exploration of the model, yielding precise lower and upper bounds on the required probability. The second tackles the case where we may only sample the MDP without knowing the exact transition dynamics. Here, we obtain probabilistic guarantees, again in terms of both the lower and upper bounds, which provides efficient stopping criteria for the approximation. In particular, the latter is an extension of statistical model-checking (SMC) for unbounded properties in MDPs. In contrast with other related approaches, we do not restrict our attention to time-bounded (finite-horizon) or discounted properties, nor assume any particular structural properties of the MDP.

Contents

1 Introduction

Markov decision processes (MDP) [How60; FV96; Put94] are a well established formalism for modelling, analysis and optimization of probabilistic systems with non-determinism, with a large range of application domains [BK08; KNP11]. For example, MDPs are used as models for concurrent probabilistic systems [CY95] or probabilistic systems operating in open environments [Seg96]. See [Whi85; Whi88; Whi93] for further applications.

In essence, MDP comprise three major parts, namely states, actions, and probabilities. Intuitively, the system evolves as follows: In a particular state, there is a set of actions to choose from. This corresponds to the *non-determinism* of the system. After choosing an action, the system then transitions into the next state according to the probability distribution associated with that action. For example, we may use MDP to represent a robot moving around in a 2D world (sometimes called 'gridworld'). The states then are (bounded, integer) coordinates, representing the current position of the robot. In each state the robot can choose to move in one of the four cardinal directions or carry out some task depending on the current location. To illustrate the inherent randomness, consider a 'move east' action. Choosing this action may move the robot to the next position east of the current one, but it might also be the case that, with some probability, a navigation component of the robot fails and we instead end up in a state north of our current position. In general, the goal is to optimize a given *objective* by choosing optimal actions. For example, we may want to control the robot such that it reaches an interesting research site with maximal probability. We additionally may be interested in minimizing time or power consumption and avoiding dangerous terrain on our way to the site.

The former example describes one of the simplest, yet very important objectives, namely *reachability*. The reachability problem consists of an MDP together with a set of designated target stats. The task is to compute the maximal probability with which the system can reach this set of states. Reachability is of particular interest since in the infinite setting many other objectives, e.g., LTL or long-run average reward can be reduced to variants of reachability. A variety of approaches has been established to solve this problem. In theory, linear programming [CY90; For+11] is the most suitable approach, as it provides exact answers (rational numbers with no representation imprecision) in polynomial time. See [Bar+08] for an application. Unfortunately, LP turns out to be quite inefficient in practice for classical reachability. For systems with more than a few thousand states, linear programming is not very usable, see, e.g., [For+11; Ash+17]. As an alternative, one can apply dynamic programming. Value iteration (VI) [Bel57] is the most prominent variant and is the default method in the probabilistic model checkers PRISM [KNP11] and Storm [Deh+17], even though it only provides an approximate solution, converging in the limit. In contrast, strategy iteration (SI) [How60; Put94; KM17] yields precise answers, but is also used to a lesser extent due to scalability issues. See for example [Ber17] for an

overview of both methods.

Despite value iteration scaling much better than linear programming, systems with more than a few million states remain out of reach, not only because of time-outs, but also memory-outs. Several approaches have been devised to deal with such large state spaces. *Compositional* techniques aim to first analyse parts of the system separately and combine the sub-results to obtain an overall result, e.g., [Cai+10; DH13; HKK13; BKW14; CCD15; BKW18]. Then, there are *abstraction* approaches which try to merge states with equivalent or sufficiently similar behaviour w.r.t. the objective in question, e.g., [DAr+02; HWZ08; Hah+10; Kat+10; Hah+10]. *Reduction* approaches try to eliminate states from the system and restrict computation to a sub-system through structural properties, e.g., [BGC04; BDG06; Cie+08; Día+12; FHM18; Bøn+19]. Moreover, as similar approach, restricting the analysis to a part of the state space is also considered, e.g., asynchronous VI in probabilistic planning [MLG05]. There, only a certain subset of states is considered for analysis, leading to speed ups in orders of magnitude. Another approach is *symbolic computation*, where the model and value functions are compactly represented using *BDD* [Bry86] and *MTBDD* [Bah+97; FMY97]. See [BKH99; KNP04; ZSF12; Wim+10; BBR14; Kle+16] for further details and applications. Finally, *statistical model checking* (SMC) [YS02; Hér+04] is also applicable. The general idea of SMC is to repeatedly sample the system in order to obtain strong statistical guarantees. Thus, SMC approaches can (at most) be *probably approximately correct* (PAC), i.e. yield an answer close to the true value with high probability, but there always is a small chance for a significant error. See [Cha+13; SLL09] for a survey on simulation-based algorithms in MDP and [Roo+17] for an application of SMC to a complex real world problem. This weaker guarantee often comes with a tremendous speed-ups and space savings: Several SMC algorithms have sub-linear or even constant space requirements, often called *model-free* algorithms. By itself, SMC algorithms are restricted to systems without non-determinism, e.g., Markov chains [You05; SVA05b]. A number of approaches tackling the issue of non-determinism have been presented (see related work for extensive details). However, these methods deal with non-determinism by either resolving it uniformly at random or sample several schedulers, both of which can lead to surprising results in certain scenarios [Boh+14]. Note that both approaches can only give a statistical estimate of a lower bound of the true achievable maximal reachability.

Surprisingly, until a few years ago, even standard value iteration as applied in popular model checkers only yielded such lower bounds, without any sound stopping criterion. In [HM14; Brá+14], an error bound was discovered independently. This bound follows from under- and (newly obtained) over-approximations converging to the true value, yielding a straightforward stopping criterion—iterate until upper and lower bound are close enough. Subsequent works included this stopping criterion in model checkers [Bai+17] and developed further sound value iteration approaches [QK18].

These error bounds not only yield a sound stopping criterion, but also enabled hybrid approaches incorporating methods where even convergence is not guaranteed. Value iteration iteratively approximates the value of all states simultaneously, requiring that all states are known. However, the above mentioned asynchronous VI evaluates states at

Algorithm 1 High-level overview of the structure of our algorithms.

Input: MDP $\mathcal{M}$, target states T, precision ε.
Output: Values (l, u) which are ε-optimal.

```
while difference between upper and lower bound in initial state larger than ε do
    Obtain a set of states to update by, e.g., sampling a path.
    for each state and action in this set do
        if this state is a target state then
            set its bounds to 1,
        else
            update action bounds based on the weighted average of its successors.
    Detect end components in relevant area of the system.
return lower and upper bound of the initial state.
```

different paces, potentially omitting a whole set of states some completely. Consequently, convergence is unclear and even its rate is unknown and hard to analyze. Yet, we are able to combine both of these ideas, inspired by *bounded real-time dynamic programming* (BRTDP) [MLG05], to obtain a correct and efficient algorithm, based on asynchronous VI. The ideas developed here have successfully been extended to, e.g., settings with long-run average reward [Ash+17], continuous time [Ash+18], or *stochastic games* (MDP with an adversary) in [Kel+18]. In a further step, [KM19] uses this approach to approximate a subset of states necessary for any analysis up to a given precision.

Note that all the above methods (with the exception of SMC) rely on an exact formalization of the system being available. In particular they require that the transition probabilities are known precisely. We call this situation the *white box* or *complete information* setting. This is a common, valid assumption when verifying, e.g., formally defined protocols involving randomization, but not so much when working with real-world systems comprising difficult dynamics, where the effects of an action often can be approximated at most. As such, these systems can be treated as a *black box*, which accept a next action to take as input and output the subsequent state, sampled from the underlying, unknown distribution associated with the state and action.

The SMC methods mentioned above still are applicable here, however they do not give any guarantees on the maximal achievable performance and instead only yield a (statistical) lower bound. Based on the ideas of *delayed Q-learning* (DQL) [Str+06] (which also only yields lower bounds) we present a PAC *model-free* algorithm, yielding statistical *upper and lower* bounds on the *maximal* reachability. This approach also has been extended to stochastic games in [AKW19], however that approach is model-based.

We mention that many algorithms working with MDP often make assumptions about the structure of the model. For example, it is sometimes required that the model is 'strongly connected' or free of *end components* [De 97] (except trivial ones). In contrast, our techniques are applicable to arbitrary MDP. For each of the two approaches, we first show how MDP with only trivial end components are handled. Then, we augment them with an *on-the-fly* detection of end components, extending the method to arbitrary MDP. Technically, we need to identify such end components in order to ensure convergence of the computed upper bounds.

To provide the reader with a preliminary overview of our approach, we present a high-level pseudo-code in Algorithm 1. As already mentioned, the fundamental idea is to compute lower and upper bounds on the true probability of reaching the target in each state. Essentially, we want to iteratively update these bounds in a converging and correct manner. In the complete information setting, this can be achieved by directly computing the weighted average of the successor bounds. For the limited information setting, we instead aggregate many successor samples. This yields a good approximation of this weighted average with high probability. As mentioned above, we also need to take care of end components. Given complete information, we again can directly solve this problem by adapting exiting graph analysis algorithms. However, with limited information we again need to employ statistical methods. In essence, if we remain inside a particular region of the system for a long enough time, there is a high probability that this region is an end component. This overall process then is repeated until the computed bounds in the initial state are close enough. The exact details of how the set of states to be updated is obtained are abstracted in the complete information setting and we only require some basic properties. We aimed for a sampling-based approach which is guided by the currently computed bounds, however a lot more ideas could be employed here. In contrast, the limited information setting requires a particular kind of sampling approach in order to ensure correctness of, e.g., the end-component detection. We highlight these differences in the respective chapters.

1.1 Related Work

In related fields such as planning and artificial intelligence, many learning-based and heuristic-driven approaches for MDP have been proposed. In the complete information setting, RTDP [BBS95] and BRTDP [MLG05] use very similar approaches, but have no stopping criterion or do not converge in general, respectively. [PGT03] uses upper and lower bounds in the setting of *partially observable MDP* (POMDP). Many other algorithms rely on certain assumptions to ensure convergence, for example by including a *discount factor* [KMN02] or restricting to the *Stochastic Shortest Path* (SSP) problems, whereas we deal with arbitrary MDP without discounting. This is addressed by an approach called FRET [Kol+11], but this only yields a lower bound. Others similarly only provide convergence in the limit [Boz+19; Jon+15], which is usually satisfactory for applications to planning / robotics, where the systems have intractably large or even uncountable state spaces. We are not aware of any attempts at generally applicable methods in the context of probabilistic verification prior to [Brá+14]. An earlier, related paper is [AL09], where heuristic methods are applied to MDP, but for generating counterexamples.

In [HM14], the authors independently discovered a stopping criterion for value iteration on general MDP. The idea behind this criterion is very similar to the upper and lower bounds in this work, but they construct and analyse the whole system. The idea of 'interval iteration', spawned by these two papers, is further developed in [Bai+17; HM18].

As explained, our algorithm based on delayed Q-learning [Str+06] yields PAC results, similar to many approaches from statistical model checking [YS02; Hér+04; SVA04].

SMC is an active area of research with extensive tool support [JLS12; BHH12; Boy+13; Bul+12; Dav+11b; You05; SVA05b] and a lot of subtle pitfalls [KCC05]. In contrast to our work, most algorithms focus on *time-bounded* or discounted properties, e.g., step-bounded reachability, rather than truly unbounded properties. Several approaches try to bridge this gap by transforming unbounded properties into testing of bounded properties, for example [YCZ10; He+10; RP09; SVA05a]. However, these approaches target models without nondeterminism and as such are not applicable to MDP. As a slight extension, [Bog+11] considers MDP with *spurious nondeterminism*, i.e. the way the nondeterminism is resolved does not influence the property of interest.

Adapting SMC techniques to models with nondeterminism such as MDP is an important topic, with several recent papers. One approach is to give nondeterminism a probabilistic interpretation, e.g., resolving it uniformly, as is done in PRISM for MDP [KNP11] and Uppaal SMC for timed automata [Dav+11b; Dav+11a; Lar13]. A second approach, taken for example by recent versions of the modes tool [HH14; DHS18; Bud+18], is to repeatedly sample schedulers, using for example *lightweight scheduler sampling* (LSS) [LST14; DAr+15], and then estimate the performance of these controllers using existing SMC methods. Uppaal Stratego [Dav+15] synthesizes a 'good' scheduler and uses it for subsequent SMC analysis. All of the above methods only yield a lower bound on the true reachability and the quality of this bound is highly dependent on the model. Others aim to indeed quantify over all strategies and approximate the true maximal value, for example [LP15; Hen+12]. The work in those papers deals with the setting of discounted or bounded properties, respectively. In [Hen+12], candidates for optimal schedulers are generated and gradually improved, which does not give upper bounds on the convergence. The nearly simultaneously published [FT14] essentially tackles the same problem. In contrast to our work, their approach is model-based, i.e. the transition probabilities are learned, and is not guided by a heuristic, requiring to learn the whole transition matrix.

Another issue of statistical methods are the analysis of *rare events*. This is, of course, very relevant for SMC approaches in general. They can be addressed using for example importance sampling [JLS12; He+10] or importance splitting [JLS13; BDH17]. We take a rather conservative approach towards rare events and delegate more sophisticated handling of this issue to future work. Note that in particular our BRTDP approach can be combined very easily with advanced techniques due to its template style.

1.2 Differences to the Published Article

Compared to the article [Brá+14], we provide the following changes.

- A complete rewrite, only retaining parts of the proof strategies.
- The related work is updated with recent advances, in particular work based on the original paper [Brá+14].
- The BRTDP approach and related proofs are extended significantly to a generic template, allowing for a variety of implementations of the sampling methods.

- Both variants of the DQL algorithm have been simplified by restructuring and a reduction of variables.
- Several technical issues of the original paper are fixed. Firstly, the proofs in the appendix proved properties of slightly different algorithms, only to conclude with a brief argument that the presented algorithms are not too different from the algorithms proven correct. Some proofs were only given implicitly or assumed to be common knowledge, in particular treatment of collapsed end components etc. Moreover, several small mistakes have been corrected.
- Lemma 16 of the original paper both has a flawed proof and an erroneous statement, which is now fixed: Firstly, the Algorithm as presented potentially never follows an ε optimal strategy, as exemplified in Example 1. Secondly, the proof applies the multiplicative Chernoff bound to variables X_i, which indicate whether the algorithm performed a particular action during a time interval. To apply this bound, the variables would need to be independent, but the X_i are highly dependent. Interestingly, a similar, but different error already is present in [Str+06] (on which the proofs in [Brá+14] are based), where the authors apply the Hoeffding bound to dependent variables in the proof of their Theorem 1.

1.3 Contributions and Structure

In Chapter 2 we set up notation and introduce some known results. We then present our contributions as follows.

- We introduce an extensible framework for efficient reachability on 'complete information' MDP without end components in Chapter 3 and extend it to arbitrary MDP in Chapter 4.
- We introduce a model-free PAC learning algorithm for reachability on 'limited information' MDP without end components in Chapter 5 and extend it to arbitrary MDP in Chapter 6.

We present an experimental evaluation of our approach in Chapter 7. Finally, we conclude in Chapter 8.

2 Preliminaries

As usual, $\mathbb{N}$ and $\mathbb{R}$ refers to the (positive) natural numbers and real numbers, respectively. Given two real numbers $a, b \in \mathbb{R}$ with $a \leq b$, $[a, b] \subseteq \mathbb{R}$ denotes the set of all real numbers between a and b inclusively. For a set S, $\overline{S}$ denotes its complement, while $S^\star$ and S^ω refers to the set of finite and infinite sequences comprising elements of S, respectively.

We assume familiarity with basic notions of probability theory, e.g., *probability spaces* and *probability measures*. A *probability distribution* over a countable set X is a mapping $d : X \to [0, 1]$, such that $\sum_{x \in X} d(x) = 1$. Its *support* is denoted by $\text{supp}(d) = \{x \in X \mid d(x) > 0\}$. $\mathcal{D}(X)$ denotes the set of all probability distributions on X. Some event happens *almost surely* (a.s.) if it happens with probability 1. For readability, we omit precise treatment of probability measures on uncountable sets and instead direct the reader to appropriate literature.

2.1 Markov Systems

Markov decision processes (MDPs) are a widely used formalism to capture both non-determinism (for, e.g., control, concurrency) and probability. First, we introduce Markov chains (MCs), which are purely stochastic and can be seen as a special case of MDP.

Definition 1. A *Markov chain* (MC) is a tuple $\mathsf{M} = (S, \delta)$, where

- S is a set of *states*, and
- $\delta : S \to \mathcal{D}(S)$ is a *transition function* that for each state s yields a probability distribution over successor states.

Note that we do not require the set of states of a Markov chain to be finite. This is mainly due to technical reasons, which become apparent later.

Next, we define MDP, which essentially extend Markov chains with non-determinism.

Definition 2. A *Markov decision process* (MDP) is a tuple $\mathcal{M} = (S, Act, Av, \Delta)$, where

- S is a finite set of *states*,
- Act is a finite set of *actions*,
- $Av : S \to 2^{Act} \setminus \{\emptyset\}$ assigns to every state a non-empty set of *available actions*, and
- $\Delta : S \times Act \to \mathcal{D}(S)$ is a *transition function* that for each state s and (available) action $a \in Av(s)$ yields a probability distribution over successor states.

A state $s \in S$ is called *terminal*, if $\Delta(s, a)(s) = 1$ for all enabled actions $a \in Av(s)$.

Remark 1. We assume w.l.o.g. that actions are unique for each state, i.e. $Av(s) \cap Av(s') = \emptyset$ for $s \neq s'$ and denote the unique state associated with action a in the MDP $\mathcal{M}$ by $\mathsf{state}(a, \mathcal{M})$. This can be achieved in general by replacing Act with $S \times Act$ and adapting Av and Δ.

Note that we assume the set of available actions to be non-empty. This means that a run can never get 'stuck' in a degenerate state without successors.

For ease of notation, we overload functions mapping to distributions $f : Y \to \mathcal{D}(X)$ by $f : Y \times X \to [0, 1]$, where $f(y, x) := f(y)(x)$. For example, instead of $\delta(s)(s')$ and $\Delta(s, a)(s')$ we write $\delta(s, s')$ and $\Delta(s, a, s')$, respectively. Furthermore, given a distribution $d \in \mathcal{D}(X)$ and a function $f : X \to \mathbb{R}$ mapping elements of a set X to real numbers, we write $d\langle f \rangle := \sum_{x \in X} d(x)f(x)$ to denote the weighted sum of f with respect to d. For example, $\delta(s)\langle f \rangle$ and $\Delta(s, a)\langle f \rangle$ denote the weighted sum of f over the successors of s in MC and s with action a in MDP, respectively.

State-Action Pairs

Throughout this work, we often speak about *state-action pairs*. This refers to tuples of the form (s, a) where $s \in S$ and $a \in Av(s)$ or equivalently $a \in Act$ and $s = \mathsf{state}(a, \mathcal{M})$. Due to our restriction that each action is associated with exactly one state, denoting both the state and action is superfluous, strictly speaking. Nevertheless, we keep both elements for consistency with other works, with the exception of Chapter 6, where this notation would introduce significant overhead. Given a set of states $S' \subseteq S$ and an available-action function $Av' : S' \to \mathcal{P}(Act) \setminus \emptyset$ we write, slightly abusing notation, $S' \times Av' = \{(s, a) \mid s \in S', a \in Av'(s)\}$ to denote the set of state-action pairs obtained in S' using Av'. In particular, $S \times Av$ denotes the set of all state-action pairs. Moreover, for a set of state-action pairs K we also write $s \in K$ if there exists an action a such that $(s, a) \in K$. Dually, we also write $a \in K$ if an appropriate state s exists.

Note that there are two isomorphic representations of sets of state-action pairs, namely as a set of pairs $X \subseteq S \times Av$ or as a pair of sets $(R, B) \in 2^S \times 2^{Act}$. We make use of both views and note explicitly when switching from one to another.

Paths & Strategies

An *infinite path* ρ in a Markov chain is an infinite sequence $\rho = s_1 s_2 \cdots \in S^\omega$, such that for every $i \in \mathbb{N}$ we have that $\delta(s_i, s_{i+1}) > 0$. A *finite path* (or *history*) $\varrho = s_1 s_2 \dots s_n \in S^\star$ is a non-empty, finite prefix of an infinite path of length $|\varrho| = n$, ending in some state s_n, denoted by $last(\varrho)$. For simplicity, we define $|\rho| = \infty$ for infinite paths ρ. We use $\rho(i)$ and $\varrho(i)$ to refer to the i-th state s_i in a given (in)finite path. A state s *occurs* in an (in)finite path ρ, denoted by $s \in \rho$, if there exists an $i \leq |\rho|$ such that $s = \rho(i)$. We denote the set of all finite (infinite) paths of an Markov chain M by $\mathsf{FPaths}_\mathsf{M}$ ($\mathsf{Paths}_\mathsf{M}$). Further, we use $\mathsf{FPaths}_{\mathsf{M},s}$ ($\mathsf{Paths}_{\mathsf{M},s}$) to refer to all (in)finite paths starting in state $s \in S$. Observe that in general $\mathsf{FPaths}_\mathsf{M}$ and $\mathsf{Paths}_\mathsf{M}$ are proper subsets of $S^\star$ and S^ω, respectively, as we imposed additional constraints.

Similarly, an *infinite path* in an MDP is some infinite sequence $\rho = s_1 a_1 s_2 a_2 \cdots \in (S \times Av)^\omega$, such that for every $i \in \mathbb{N}$, $a_i \in Av(s_i)$, setting the length $|\rho| = \infty$. *Finite paths* ϱ and $last(\varrho)$ are defined analogously as elements of $(S \times Av)^\star \times S$ and the respective last state. Again, $\rho(i)$ and $\varrho(i)$ refer to the i-th state in an (in)finite path with an analogous definition of a state occurring, $|\varrho|$ denotes the length of a finite path, and we refer to the set of (in)finite paths of an MDP $\mathcal{M}$ by $\mathsf{FPaths}_\mathcal{M}$ ($\mathsf{Paths}_\mathcal{M}$) and write $\mathsf{FPaths}_{\mathcal{M},s}$ ($\mathsf{Paths}_{\mathcal{M},s}$) for all such paths starting in a state $s \in S$. Further, we use $\rho^a(i)$ and $\varrho^a(i)$ to denote the i-th action in the respective path. We say that a state-action pair (s, a) is in an (in)finite path ϱ if there exists an $i < |\varrho|$ with $s = \varrho(i)$ and $a = \varrho^a(i)$.

A Markov chain together with a state $s \in S$ naturally induces a unique probability measure $\mathsf{Pr}_{\mathsf{M},s}$ over infinite paths [BK08, Chapter 10]. For MDP, we first need to eliminate the non-determinism in order to obtain such a probability measure. This is achieved by *strategies* (also called *policy*, *controller*, or *scheduler*).

Definition 3. A strategy on an MDP $\mathcal{M} = (S, Act, Av, \Delta)$ is a function $\pi : \mathsf{FPaths}_\mathcal{M} \to \mathcal{D}(Act)$, such that $\mathrm{supp}(\pi(\varrho)) \subseteq Av(last(\varrho))$ for all $\varrho \in \mathsf{FPaths}_\mathcal{M}$.

Intuitively, a strategy is a 'recipe' describing which step to take in the current state, given the evolution of the system so far. Note that the strategy may yield a distribution on the actions to be taken next.

A strategy π is called *memoryless* (or *stationary*) if it only depends on $last(\varrho)$ for all finite paths ϱ and we identify it with $\pi : S \to \mathcal{D}(Act)$. Similarly, it is called *deterministic*, if it always yields a Dirac distribution, i.e. picks a single action to be played next, and we identify it with $\pi : \mathsf{FPaths}_\mathcal{M} \to Act$. Together, *memoryless deterministic* strategies can be treated as functions $\pi : S \to Act$ mapping each state to an action. We write $\Pi_\mathcal{M}$ to denote the set of all strategies of an MDP $\mathcal{M}$, $\Pi^{\mathsf{M}}_\mathcal{M}$ for memoryless strategies, and $\Pi^{\mathsf{MD}}_\mathcal{M}$ for all memoryless deterministic strategies.

Fixing any strategy π induces a Markov chain $\mathcal{M}^\pi = (\mathsf{FPaths}_\mathcal{M}, \delta^\pi)$, where for some state $\varrho = s_1 a_1 \dots s_n \in \mathsf{FPaths}_\mathcal{M}$, appropriate action $a_{n+1} \in Av(s_n)$ and successor state $s_{n+1} \in \mathrm{supp}(\Delta(s_n, a_{n+1}))$ the successor distribution is defined as $\delta^\pi(\varrho, \varrho a_{n+1} s_{n+1}) = \pi(\varrho, a_{n+1}) \cdot \Delta(s, a_{n+1}, s_{n+1})$. In particular, for any MDP $\mathcal{M}$, strategy $\pi \in \Pi_\mathcal{M}$, and state s, we obtain a measure over paths[(1)] $\mathsf{Pr}_{\mathcal{M}^\pi,s}$, which we refer to as $\mathsf{Pr}^\pi_{\mathcal{M},s}$. Observe that all these measures operate on the same probability space, namely the set of all infinite paths $\mathsf{Paths}_\mathcal{M}$. See [Put94, Section 2.1.6] for further details. Consequently, given a measurable event A, we can define the maximal probability of this event starting from state $\hat{s}$ under any strategy by

$$\mathsf{Pr}^{\sup}_{\mathcal{M},s}[A] := \sup_{\pi \in \Pi_\mathcal{M}} \mathsf{Pr}^\pi_{\mathcal{M},\hat{s}}[A].$$

Note that depending on the structure of A it may be the case that no optimal witness exists and we have to resort to the supremum instead of the maximum. We lift this restriction for our particular use case later on. For a memoryless strategy $\pi \in \Pi^{\mathsf{M}}_\mathcal{M}$, we can identify $\mathcal{M}^\pi$ with a Markov chain over the states of $\mathcal{M}$.

[(1)] Technically, this measure operates on infinite sequences of finite paths, as each state of $\mathcal{M}^\pi$ is a finite path. But, this measure can easily be projected directly on finite paths.

Given an MDP $\mathcal{M}$, memoryless strategy $\pi \in \Pi^{\mathsf{M}}_{\mathcal{M}}$, and a function assigning a value to each state-action pair $f : S \times Av \to \mathbb{R}$, we define $\pi[f] : S \to \mathbb{R}$ as the expected value of taking one step in state s following the strategy π, i.e.

$$\pi[f](s) := \sum_{a \in Av(s), s' \in S} \pi(s,a) \cdot f(s,a).$$

Strongly Connected Components and End Components

A non-empty set of states $C \subseteq S$ in a Markov chain is *strongly connected* if for every pair $s, s' \in C$ there is a non-trivial path from s to s'. Such a set C is a *strongly connected component* (SCC) if it is inclusion maximal, i.e. there exists no strongly connected C' with $C \subsetneq C'$. Thus, each state belongs to at most one SCC. An SCC is called *bottom strongly connected component* (BSCC) if additionally no path leads out of it, i.e. for all $s \in C, s' \in S \setminus C$ we have $\delta(s,s') = 0$. The set of SCCs and BSCCs in an MC M is denoted by $\mathrm{SCC}(\mathsf{M})$ and $\mathrm{BSCC}(\mathsf{M})$, respectively.

The concept of SCCs is generalized to MDPs by so called *(maximal) end components* [De 97]. Intuitively, an end component describes a set of states in which the system can remain forever.

Definition 4. Let $\mathcal{M} = (S, Act, Av, \Delta)$ be an MDP. A pair (R, B), where $\emptyset \neq R \subseteq S$ and $\emptyset \neq B \subseteq \bigcup_{s \in R} Av(s)$, is an *end component* of an MDP $\mathcal{M}$ if

(i) for all $s \in R, a \in B \cap Av(s)$ we have $\mathrm{supp}(\Delta(s,a)) \subseteq R$, and

(ii) for all $s, s' \in R$ there is a finite path $\varrho = sa_0 \dots a_n s' \in \mathsf{FPaths}_{\mathcal{M}} \cap (R \times B)^\star \times R$, i.e. the path stays inside R and only uses actions in B.

An end component (R, B) is a *maximal end component* (MEC) if there is no other end component (R', B') such that $R \subseteq R'$ and $B \subseteq B'$.

By abuse of notation, we identify an end component with the respective set of states, e.g., $s \in E = (R, B)$ means $s \in R$. Observe that given two overlapping ECs (R_1, B_1) and (R_2, B_2) with $R_1 \cap R_2 \neq \emptyset$, their union $(R_1 \cup R_2, B_1 \cup B_2)$ also is an EC. Again, a MEC is *bottom* if there are no outgoing transitions; each state belongs to at most one MEC. The set of ECs of an MDP $\mathcal{M}$ is denoted by $\mathrm{EC}(\mathcal{M})$, the set of MECs by $\mathrm{MEC}(\mathcal{M})$.

Remark 2. For a Markov chain M, the computation of $\mathrm{SCC}(\mathsf{M})$, $\mathrm{BSCC}(\mathsf{M})$ and a topological ordering of the SCCs can be achieved in linear time w.r.t. the number of states and transitions by, e.g., Tarjan's algorithm [Tar72]. Similarly, the MEC decomposition of an MDP can be computed in polynomial time [CY95]. For improved algorithms on general MDP and various special cases see [CH11; CH12; CH14].

Note that these components fully capture the limit behaviour of any Markov chain and decision process, respectively. Intuitively, both of these statements say that a run of such systems eventually remains inside one BSCC or MEC forever, respectively.[2]

[2] The measurability of the sets in the following two lemmas is well known, proofs can be found in, e.g., [BK08, Chapter 10].

Lemma 1 (MC almost-sure absorption)**.** *For any MC* M *and state* s*, we have that* $\mathsf{Pr}_{\mathsf{M},s}[\{\rho \mid \exists R_i \in \mathrm{BSCC}(\mathsf{M}).\exists n_0 \in \mathbb{N}.\forall n > n_0.\rho(n) \in R_i\}] = 1$*.*

Proof. Follows from [BK08, Theorem 10.27]. □

Lemma 2 (MDP almost-sure absorption)**.** *For any MDP* $\mathcal{M}$*, state* s*, and strategy* π*, we have that* $\mathsf{Pr}^{\pi}_{\mathcal{M},s}[\{\rho \mid \exists (R_i, B_i) \in \mathrm{MEC}(\mathcal{M}).\exists n_0 \in \mathbb{N}.\forall n > n_0.\rho(n) \in R_i\}] = 1$*.*

Proof. Follows from [De 97, Theorem 3.2]. □

2.2 Reachability

For an MDP $\mathcal{M} = (S, Act, Av, \Delta)$ and a set of *target states* $T \subseteq S$, *bounded reachability* for step k, denoted by $\Diamond^{\leq k} T = \{\rho \in \mathsf{Paths}_{\mathcal{M}} \mid \exists i \in \{0, \ldots, k\}.\ \rho(i) \in T\}$, is the set of all infinite paths that reach a state in T within k steps. Analogously, *(unbounded) reachability* $\Diamond T = \{\rho \in \mathsf{Paths}_{\mathcal{M}} \mid \exists i \in \mathbb{N}.\ \rho(i) \in T\}$ are all paths which eventually reach the target set T. We overload the $\Diamond$ operator to also accept sets of state-action pairs and sets of actions, with analogous semantics. The sets of paths produced by $\Diamond$ are measurable for any MDP, target set, and step bound [BK08, Sec. 10.1.1].[3] Note that for a set T, both $\Diamond\overline{T}$ and $\overline{\Diamond T}$ are well-defined, however they refer to two different concepts. The former denotes the set of all paths reaching a state not in T, whereas the latter is the set of all paths which never reach T (also called *co-reachability* or *safety*).

Now, it is straightforward to define the *maximal reachability problem* of a given set of states. Given an MDP $\mathcal{M}$, target set T, and state s, we are interested in computing the maximal probability of eventually reaching T, starting in state s. Formally, we want to compute the *value* of state s:

$$\mathcal{V}(s) := \mathsf{Pr}^{\sup}_{\mathcal{M},s}[\Diamond T] = \sup_{\pi \in \Pi_{\mathcal{M}}} \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\Diamond T].$$

It is known that an optimal strategy always exists and memoryless deterministic strategies are sufficient to achieve the optimal value [De 97, Theorem 3.10], i.e.

$$\mathcal{V}(s) = \mathsf{Pr}^{\max}_{\mathcal{M},s}[\Diamond T] = \max_{\pi \in \Pi_{\mathcal{M}}} \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\Diamond T] = \max_{\pi \in \Pi^{\mathsf{MD}}_{\mathcal{M}}} \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\Diamond T].$$

This state value function satisfies a straightforward fixed point equation, namely

$$\mathcal{V}(s) = \begin{cases} 1 & \text{if } s \in T, \\ \max_{a \in Av(s)} \Delta(s,a)\langle\mathcal{V}\rangle & \text{otherwise.} \end{cases} \tag{2.1}$$

Moreover, $\mathcal{V}$ is the *smallest* fixed point of this equation [Put94]. In our approach, we also deal with values of state-action pairs $(s, a) \in S \times Av$, where

$$\mathcal{V}(s,a) := \Delta(s,a)\langle\mathcal{V}\rangle = \textstyle\sum_{s' \in S} \Delta(s,a,s') \cdot \mathcal{V}(s').$$

[3] Recall that we defined MDP to have finite state and action sets.

Intuitively, $\mathcal{V}(s,a)$ is the value in state s when playing action a and then acting optimally. The overall value of s, $\mathcal{V}(s)$, is obtained by choosing an optimal action, i.e. $\mathcal{V}(s) = \max_{a \in Av(s)} \mathcal{V}(s,a)$.

Remark 3. Our algorithms primarily work by approximating these state-action values and derive state-values by the above equation. This may seem counter-intuitive at first, since we could as well directly work with state values and derive state-action values as described above, saving memory. However, our approaches are inspired by *reinforcement learning* [SB98], explained later, which traditionally assigns values to actions. Thus, we stick with this convention in our algorithms as well. Finally, in the limited information setting of Chapters 5 and 6, the algorithms do not have access to the exact transition probabilities and hence cannot exploit the above equation.

See [For+11, Sec. 4] for an in-depth discussion of reachability on finite MDP.

Approximate Solutions

The value of a state $\mathcal{V}(s)$ can, for example, be determined using *linear programming* [CY90; For+11][4] in polynomial time [Kha79; Kar84]. Unfortunately, this approach turns out to be inefficient in practice [HM14; Ash+17]. One way to potentially ease the task is by only considering *approximate solutions*. In particular, on top of an MDP $\mathcal{M}$, starting state $\hat{s}$, and target set T, we assume that we are given a precision requirement $\varepsilon > 0$. We say a strategy π is ε-optimal, if $\mathsf{Pr}^{\pi}_{\mathcal{M},\hat{s}}[\Diamond T] + \varepsilon > \mathcal{V}(s)$. Analogously, a tuple of values (l,u) is *ε-optimal* if $0 \leq u - l < \varepsilon$ and $\mathcal{V}(\hat{s}) \in [l,u]$, i.e. l and u are lower and upper bounds on the value, respectively. All algorithms in this work are designed to efficiently compute such ε-optimal values. Due to technical details, we omit computation of a witness strategy.

2.3 Probabilistic Learning Algorithms

In order to obtain such approximate solutions, we study a class of *learning-based* algorithms that (stochastically) approximate the value function, inspired by approaches from the field of machine learning. Let us fix an MDP $\mathcal{M} = (S, Act, Av, \Delta)$, starting state $\hat{s}$, and target set $T \subseteq S$. Recall that by approximating the state-action values, we approximate the overall value of a state. Inspired by *BRTDP* (bounded real-time dynamic programming) [MLG05][5], we consider algorithms which maintain and update Upper bounds $\mathsf{Up} : S \times Av \to [0,1]$ and Lower bounds $\mathsf{Lo} : S \times Act \to [0,1]$ of these sate-action values $\mathcal{V}(s,a)$. The functions Up and Lo are initialised to appropriate values such that $\mathsf{Lo}(s,a) \leq \mathcal{V}(s,a) \leq \mathsf{Up}(s,a)$ for all $s \in S$ and $a \in Av(s)$. This is trivially satisfied by $\mathsf{Lo}(\cdot,\cdot) = 0$ and $\mathsf{Up}(\cdot,\cdot) = 1$, but some non-trivial bounds obtained by previous computations or domain knowledge can be incorporated. We define the state-bounds by

$$\mathsf{Up}(s) := \max_{a \in Av(s)} \mathsf{Up}(s,a), \qquad \text{and} \qquad \mathsf{Lo}(s) := \max_{a \in Av(s)} \mathsf{Lo}(s,a).$$

[4] See [Sch99] for details on linear programming in general.
[5] See [BBS95] for the 'non-bounded' case *RTDP*.

Clearly, $\mathsf{Lo}(s) \leq \mathcal{V}(s) \leq \mathsf{Up}(s)$, thus we can determine the value of a state ε-precise when these respective bounds are sufficiently close. In particular, if we have that

$$\mathsf{Up}(\hat{s}) - \mathsf{Lo}(\hat{s}) = \max_{a \in Av(\hat{s})} \mathsf{Up}(\hat{s}, a) - \max_{a \in Av(\hat{s})} \mathsf{Lo}(\hat{s}, a) < \varepsilon,$$

the values $(\mathsf{Lo}(s), \mathsf{Up}(s))$ are ε-optimal.

Our *learning algorithms* update the upper and lower bounds by repeatedly selecting 'interesting' / promising state-action pairs of the system $\mathcal{M}$, usually by sampling the system beginning in the starting state $\hat{s}$. As such, they are similar to *Q-learning* [WD92] approaches, a commonly used reinforcement learning technique. By following appropriate sampling heuristics the algorithm learns 'important' areas of the system and focusses computation there, potentially omitting irrelevant parts of the state space without sacrificing correctness. For example, given a state s we propose to select an action a with maximal upper bound $\mathsf{Up}(s, a)$, as such an action is the most 'promising' one. Then, either this action keeps up to its promise, which will eventually be reflected by an increasing lower bound, or the algorithm finds that the upper bound is too high and lowers it. As such, this idea is very similar to *optimism in the face of uncertainty* [Sze10, Section 4.2], [LR85]. We only know that the exact value lies between the upper and lower bound, thus we are optimistic and assume the best value during sampling.

The algorithms repeatedly experience (learning) *episodes*, where each episode consists of several *steps*. One episode corresponds to sampling a path of some length in the system, while one step corresponds to sampling the successor state, i.e. each episode comprises several steps. Throughout this paper, we use $\mathsf{e} \in \mathbb{N}$ exclusively to refer to the e-th episode of some algorithm execution. Later we also refer to distinct steps within episodes by $\mathsf{t} \in \mathbb{N}$. In particular, t denotes to the t-th overall step. Finally, t_e denotes the first step of the e-th episode, i.e. its starting step.

The considered algorithms make heavy use of randomness during their execution. Thus, in order to reason about them, we model them as a stochastic process over an appropriate measure space $(\mathfrak{A}, \mathcal{A}, \mathbb{P}_\mathsf{A})$. The entire state of our algorithms at the beginning of episode e can be derived from the sequences of state-action pairs it considered until episode e.[6] Hence, we use episodes as our primitive objects. We need to consider both finite and infinite episodes, since (i) a single episode may comprise infinitely many state-action pairs and (ii) the algorithm potentially does not terminate, giving rise to an infinite number of episodes. Thus, we set $\mathfrak{A} = ((S \times Av \times S)^\times)^\times$, where $S^\times = S^\star \cup S^\omega$. The tuples $S \times Av \times S$ correspond to the current state, chosen action, and sampled successor state, respectively. The σ-field $\mathcal{A}$ is obtained analogously to the σ-field for Markov chains by considering cylinder sets induced by finite prefixes, see [Put94, Section 2.1.6]. For a given prefix, its probability can be obtained by computing the probability of each episode occurring in the MDP given the current state of the algorithm.

Now that we defined the probability space these algorithms operate in, we can define

[6] Observe that Algorithm 2 and Algorithm 3 are allowed to introduce some further side effects due to their 'template'-structure. We assume w.l.o.g. that these side effects are either deterministic or can be properly incorporated into the above measure space.

notions like almost sure convergence.

Definition 5. Denote by $\mathsf{A}(\varepsilon)$ the instance of learning algorithm A with precision ε. We say that A *converges (almost) surely* if, for every MDP $\mathcal{M}$, starting state $\hat{s}$, target set T, and precision $\varepsilon > 0$, the computation of $\mathsf{A}(\varepsilon)$ terminates (almost) surely and yields ε-optimal values l and u.

We consider a symbolic input encoding, where the MDP's properties are specified implicitly. In particular, we design our algorithms such that they are applicable when the available actions Av and transition function Δ are given as oracles. This means that given a state s we can compute $Av(s)$, and given a state-action pair (s, a) we obtain the successor distribution $\Delta(s, a)$. This allows us to achieve sub-linear runtime for some classes of MDP w.r.t. their number of states and transitions. Note that most practical modelling languages such as the PRISM language [KNP11] or JANI [Bud+17] describe models in such a way.

Since our learning algorithms in essence only rely on being able to repeatedly sample the system we can drastically reduce the knowledge needed about the system. In particular, we consider the setting of *limited information*, where the algorithm only has very restricted access to the system in question. There, we are only provided with bounds on some properties of the MDP, e.g., the number of states, together with an oracle for the available actions and a 'sampling' oracle, which yields a successor according to the underlying, hidden distributions. The algorithm thus can only simulate an execution of the MDP starting from a state s, repeatedly choosing an action from the set of available actions and querying the sampling oracle for a successor. This corresponds to a 'black-box' setting, where we can easily interact with a system and observe the current state, but have very limited knowledge about its internal transition structure, as might be the case with complex physical systems.

Here, we cannot directly apply the ideas of Q-learning, since the value of the sampled successor might not correspond to the actual value of the action. Instead, the algorithm remembers the result of recent visits, *delaying* the learning update. By gathering enough information, the average of these results corresponds to the true value with high confidence. This idea is exploited by *delayed Q-learning* [Str+06]. In this setting, we inherently cannot guarantee almost sure convergence, instead we demand that the algorithm terminates correctly with sufficiently high probability, specified by the *confidence* $\delta > 0$.

Definition 6. Denote by $\mathsf{A}(\varepsilon, \delta)$ the instance of learning algorithm A with precision ε and confidence δ. We say that A is *probably approximately correct* (PAC) if for every MDP $\mathcal{M}$, starting state $\hat{s}$, target set T, precision $\varepsilon > 0$ and confidence $\delta > 0$, with probability at least $1 - \delta$ the computation of $\mathsf{A}(\varepsilon, \delta)$ terminates and yields ε-optimal values l and u. In other words, we require that the set of correct and terminating executions has a measure of at least $1 - \delta$ under $\mathbb{P}_{\mathsf{A}}$.

See [Val84; Ang88; Str+06; Str08] for several, slightly different variants of PAC. Some definitions also require that the result is obtained within a particular time-bound. We prove appropriate bounds for both variants of our PAC approach.

Remark 4. Note that we assume the system to be 'observable' in both settings, i.e. the algorithm can access the *precise* current state of the system and the set of available actions. Extending our methods to *partially observable* systems, e.g. POMDP, is left for future work. Moreover, we also assume that the system can be repeatedly 'reset' into the initial configuration.

3 Complete Information – MDP without End Components

In this section, we treat the case of complete information, i.e. the algorithm has full access to the system, in particular its transition function Δ. Moreover, we assume that the system in question has no MECs, except two distinguished terminal states. This greatly simplifies the reachability problem and allows us to gradually introduce our approach. In Chapter 4, we highlight the difficulties of MECs (see Example 2) and generalize our approach to arbitrary MDP.

3.1 The Ideas of Value Iteration

Our approach is based on ideas related to *value iteration* (VI) [How60]. Thus, we first explain the basic principles of VI. Value iteration is a technique to solve, among others, reachability queries on MDP. It essentially amounts to applying *Bellman iteration* [Bel66] corresponding to the fixed point equation in Equation 2.1 [For+11, Sec. 4.2]. In particular, starting from an initial value vector v_0 with $v_0(s) = 1$ if $s \in T$ and 0 otherwise, we apply the iteration

$$v_{n+1}(s) = \begin{cases} 1 & \text{if } s \in T, \\ \max_{a \in Av(s)} \Delta(s,a)\langle v_n \rangle & \text{otherwise.} \end{cases}$$

It is known that this iteration converges to the true value $\mathcal{V}$ in the limit from below, i.e. for all states s we have (i) $\lim_{n\to\infty} v_n(s) = \mathcal{V}(s)$ and (ii) $v_n(s) \leq v_{n+1}(s) \leq \mathcal{V}(s)$ for all iterations n [Put94, Thm. 7.2.12][1]. It is not difficult to construct a system where convergence up to a given precision takes exponential time [HM14], but in practice VI often is much faster than methods based on *linear programming* (LP)[2], which in theory has worst-case polynomial runtime and yields precise answers [Kar84]. An important practical issue of VI is the absence of a *stopping criterion*, i.e. a straightforward way of determining in general whether the current values $v_n(s)$ are close to the true value function $\mathcal{V}(s)$, as discussed in, e.g., [For+11, Sec. 4.2]. We solve this problem by additionally computing upper bounds, converging to the true value from above.

While the value iteration approach updates all states synchronously, the iteration can also be executed *asynchronously*. This means that we do not have to update the values of all states (or state-action pairs) simultaneously. Instead, the update order may be chosen by heuristics, as long as fairness constraints are satisfied, i.e. eventually all states get updated. This observation is essential for our approach, since we want to focus our computation on 'important' areas.

[1] Note that reachability is a special case of *expected total reward*, obtained by assigning a one-time reward of 1 to each goal state.

[2] See [BK08, Thm. 10.105] for an LP-based solution of reachability.

Algorithm 2 The BRTDP learning algorithm for MDPs without ECs.

Input: MDP $\mathcal{M}$, state $\hat{s}$, precision ε, and initial bounds Up_1 and Lo_1.
Output: ε-optimal values (l, u), i.e. $\mathcal{V}(\hat{s}) \in [l, u]$ and $0 \leq u - l < \varepsilon$.

```
1: e ← 1                                                          ▷ Initialize
2: while Up_e(ŝ) − Lo_e(ŝ) ≥ ε do
3:     ϱ_e ← SamplePairs(M, ŝ, Up_e, Lo_e, ε)                    ▷ Sample pairs to update
4:     Up_{e+1} ← Up_e, Lo_{e+1} ← Lo_e
5:     for all (s, a) ∈ ϱ_e do                                    ▷ Update the upper and lower bounds
6:         Up_{e+1}(s, a) ← Δ(s, a)⟨Up_e⟩
7:         Lo_{e+1}(s, a) ← Δ(s, a)⟨Lo_e⟩
8:     e ← e + 1
9: return (Lo_e(ŝ), Up_e(ŝ))
```

3.2 The No-EC BRTDP Algorithm

With these ideas in mind, we are ready to present our first algorithm. Throughout this section, fix a required precision $\varepsilon > 0$, an MDP $\mathcal{M} = (S, Act, Av, \Delta)$ with two distinguished states $s_+, s_- \in S$, target set $T = \{s_+\}$ and a starting state $\hat{s}$. We assume that $\mathcal{M}$ has no MECs except the two terminal states s_+ and s_-.

Assumption 1. MDP $\mathcal{M}$ has no MECs, except two trivial ones comprising the target state s_+ and sink state s_-, respectively. Formally, we require that $\mathrm{MEC}(\mathcal{M}) = \{(\{s_+\}, Av(s_+)), (\{s_-\}, Av(s_-))\}$.

Observe that with Assumption 1 and $T = \{s_+\}$, we clearly have $\mathcal{V}(s_+) = 1$ and $\mathcal{V}(s_-) = 0$. We define our *BRTDP* approach in Algorithm 2. Recall that we defined $\mathsf{Up}(s) = \max_{a \in Av(s)} \mathsf{Up}(s, a)$ and $\mathsf{Lo}(s)$ analogously. As already mentioned in the introduction, the algorithm repeatedly samples sets of state-action pairs from the system. Based on these experiences, it updates the upper and lower bounds using *Bellman updates* (or *Bellman backups*), corresponding to Equation (2.1), until convergence.

To allow for practical optimization, we leave the sampling method SamplePairs undefined and instead only require some generic properties. A simple implementation is given by sampling a path starting in the initial state and following random actions. However, SamplePairs may use randomization and sophisticated guidance heuristics, as long as it satisfies certain conditions in the limit.

Remark 5. We highlight that SamplePairs is not required to return paths. Instead it can yield any set of state-action pairs. However, when dealing with the limited information setting, we require sampling paths. Thus, it may be instructive to think of SamplePairs as a procedure returning paths.

3.3 Proof of Correctness

In this section, we prove correctness of the algorithm, i.e. that the returned result is correct and that the algorithm terminates. We now first establish correctness of the result, assuming that the received input is sane.

Assumption 2. We have that (i) the given initial bounds Up_1 and Lo_1 are correct, i.e. $\mathsf{Lo}_1(s,a) \leq V(s,a) \leq \mathsf{Up}_1(s,a)$ for all $(s,a) \in S \times Av$, and (ii) $\mathsf{Lo}_1(s_+) = 1$ and $\mathsf{Up}_1(s_-) = 0$.

Lemma 3. *Assume that Assumption 2 holds. Then, during any execution of Algorithm 2 we have for every episode* e *and all state-action pairs* (s,a) *that*

$$\mathsf{Lo}_\mathsf{e}(s,a) \leq \mathsf{Lo}_{\mathsf{e}+1}(s,a) \leq \mathcal{V}(s,a) \leq \mathsf{Up}_{\mathsf{e}+1}(s,a) \leq \mathsf{Up}_\mathsf{e}(s,a).$$

Proof. Initially, we have that $\mathsf{Lo}_1(s,a) \leq \mathcal{V}(s,a) \leq \mathsf{Up}_1(s,a)$ by Assumption 2. The updates in Lines 6 and 7 clearly preserve these inequalities. A simple inductive argument concludes the proof. □

Lemma 4. *Assume that Assumption 2 holds. Then, the result* (l,u) *of Algorithm 2 is correct, i.e. (i)* $0 \leq u - l < \varepsilon$*, and (ii)* $\mathcal{V}(\hat{s}) \in [l,u]$.

Proof. Clearly, (i) immediately follows from Lemma 3 and the main loop condition in Line 2. Similarly, (ii) also follows from Lemma 3. □

In order to prove (almost sure) convergence of Algorithm 2, we need some assumptions on SamplePairs. Intuitively, SamplePairs may not neglect actions which might be the optimal ones. In order to allow for a wide range of implementations for SamplePairs, we present the rather liberal but technical condition of *fairness* in Assumption 3. We further explain each part of this assumption in the following proof of convergence.

Before we present the assumption, we introduce the set of Up-optimal actions, which is also used in the proof. Let $\mathsf{MaxA}_\mathsf{e}(s) := \arg\max_{a \in Av(s)} \mathsf{Up}_\mathsf{e}(s,a)$ the set of actions optimal (w.r.t. Up_e) in state s during episode e. Note that assuming the algorithm does not converge, the set $\mathsf{MaxA}_\mathsf{e}(s)$ may change infinitely often. For example, two equivalent actions may get updated in an alternating fashion. Thus, for each state s, we also define the set of actions that are optimal infinitely often as $\mathsf{MaxA}_\infty(s) := \bigcap_{k=1}^{\infty} \bigcup_{\mathsf{e}=k}^{\infty} \mathsf{MaxA}_\mathsf{e}(s)$. This set is non-empty, since there are only finitely many actions and $\mathsf{MaxA}_\mathsf{e}(s)$ is non-empty for any episode e.

Assumption 3. Let $\{\mathsf{Up}_\mathsf{e}\}_{\mathsf{e}=1}^{\infty}$ and $\{\mathsf{Lo}_\mathsf{e}\}_{\mathsf{e}=1}^{\infty}$ be consistent sequences of upper and lower bounds, i.e. $\mathsf{Lo}_1(s,a) \leq \mathsf{Lo}_2(s,a) \leq \cdots \leq \mathcal{V}(s,a) \leq \cdots \leq \mathsf{Up}_2(s,a) \leq \mathsf{Up}_1(s,a)$ for all state-action pairs (s,a). Moreover, let $\varrho_1, \varrho_2, \cdots \in \mathcal{P}(S \times Act) \setminus \emptyset$ any infinite sequence of non-empty state-action sets produced by $\textsc{SamplePairs}(\mathcal{M}, \hat{s}, \mathsf{Up}_\mathsf{e}, \mathsf{Lo}_\mathsf{e}, \varepsilon)$.

Set $S_\infty = \bigcap_{k=1}^{\infty} \bigcup_{\mathsf{e}=k}^{\infty} \{s \in S \mid s \in \varrho_\mathsf{e}\}$ the set of all states which occur infinitely often. We analogously define the set of actions occurring infinitely often, denoted Act_∞. Then, we have that

1. the initial state is sampled infinitely often, i.e. $\hat{s} \in S_\infty$,
2. all actions which are optimal infinitely often are also sampled infinitely often, i.e. $\mathsf{MaxA}_\infty(s) \subseteq Act_\infty$ for every $s \in S_\infty$, and
3. all successors of optimal actions are sampled infinitely often, i.e. for every $s \in S_\infty$ and $a \in \mathsf{MaxA}_\infty(s)$ we have that $\mathrm{supp}(\Delta(s,a)) \subseteq S_\infty$.

We say SamplePairs almost surely satisfies Assumption 3, if its conditions hold with probability 1.

In essence, the assumption requires that all states which are reachable by following optimal actions are indeed reached infinitely often in the limit: Starting from the initial state (Item 1), we select each optimal action infinitely often (Item 2) and explore all successors of these actions (Item 3). For each of these successors, we again select all optimal actions, etc.

Lemma 5. *Algorithm 2 terminates under Assumptions 1, 2 and 3. It terminates almost surely if Assumption 3 is satisfied almost surely.*

Proof. We prove the second case, i.e. almost sure termination, by contradiction. Assume that Assumptions 1 and 2 hold, and that 3 holds a.s.. Further, assume for contradiction that the set of non-terminating executions of Algorithm 2 has non-zero measure, i.e. the while-loop is executed infinitely often.

Given some execution of Algorithm 3, define $\mathrm{Diff}_{\mathsf{e}}(s,a) := \mathsf{Up}_{\mathsf{e}}(s,a) - \mathsf{Lo}_{\mathsf{e}}(s,a)$. Fix an arbitrary action $a_{\mathsf{e}}^{\max}(s) \in \mathsf{MaxA}_{\mathsf{e}}(s)$ for each episode e. Clearly, for any such action $a_{\mathsf{e}}^{\max}(s)$ we have $\mathrm{Diff}_{\mathsf{e}}(s, a_{\mathsf{e}}^{\max}(s)) = \mathsf{Up}_{\mathsf{e}}(s) - \mathsf{Lo}_{\mathsf{e}}(s, a_{\mathsf{e}}^{\max}) \geq \mathsf{Up}_{\mathsf{e}}(s) - \mathsf{Lo}_{\mathsf{e}}(s)$. By Lemma 3, the limits $\mathsf{Up}_\infty(s,a) := \lim_{\mathsf{e}\to\infty} \mathsf{Up}_{\mathsf{e}}(s,a)$ and $\mathsf{Lo}_\infty(s,a) := \lim_{\mathsf{e}\to\infty} \mathsf{Lo}_{\mathsf{e}}(s,a)$ are well defined and finite for any state-action pair (s,a). Thus, $\mathrm{Diff}(s,a) := \lim_{\mathsf{e}\to\infty} \mathrm{Diff}_{\mathsf{e}}(s,a)$ and $\mathrm{Diff}(s) := \limsup_{\mathsf{e}\to\infty} \mathrm{Diff}_{\mathsf{e}}(s, a_{\mathsf{e}}^{\max}(s))$ is also well defined and finite. We prove that $\mathrm{Diff}(\hat{s}) = 0$ for almost all executions, contradicting the assumption, as then necessarily $\mathsf{Up}_{\mathsf{e}}(\hat{s}) - \mathsf{Lo}_{\mathsf{e}}(\hat{s}) \leq \mathrm{Diff}_{\mathsf{e}}(\hat{s}) < \varepsilon$ for some e a.s.

Observe that the preconditions of Assumption 3 are satisfied through Lemma 3 and Assumption 2, hence we have $\hat{s} \in S_\infty$ a.s. **(I)**. Let S_∞ the set of states seen infinitely often as defined in Assumption 3. By the assumption, we also have that $\mathrm{supp}(\Delta(s,a)) \subseteq S_\infty$ for all $s \in S_\infty, a \in \mathsf{MaxA}_\infty(s)$ a.s. **(II)**.

Now, we identify a witness action $a_{\mathrm{Diff}}(s)$ for the lim sup of $\mathrm{Diff}(s)$, i.e. an action $a_{\mathrm{Diff}}(s)$ such that $\mathrm{Diff}_\infty(s) = \lim_{\mathsf{e}\to\infty} \mathrm{Diff}_{\mathsf{e}}(s, a_{\mathrm{Diff}}(s))$ and derive a fixed-point equation. We have that $\mathsf{Up}_\infty(s,a) = \mathsf{Up}_\infty(s,a')$ for all $s \in S_\infty$ and $a, a' \in \mathsf{MaxA}_\infty(s)$, as otherwise one of the two actions would not be optimal eventually. Consequently, the limit $\lim_{\mathsf{e}\to\infty} \mathsf{Up}_{\mathsf{e}}(s, a_{\mathsf{e}}^{\max})$ is well defined and equals $\mathsf{Up}_\infty(s,a)$ for any $a \in \mathsf{MaxA}_\infty(s)$. Clearly, $\limsup_{\mathsf{e}\to\infty} \mathsf{Lo}_{\mathsf{e}}(s, a_{\mathsf{e}}^{\max})$ also is well defined, hence the lim sup of $\mathrm{Diff}(s)$ distributes over the minus. Recall that for each state-action pair, the limit of $\mathsf{Lo}_\infty(s,a)$ is well defined. Thus, the sequence $\mathsf{Lo}_{\mathsf{e}}(s, a_{\mathsf{e}}^{\max})$ only has finitely many accumulation points and there exists an action $a_{\mathrm{Diff}}(s) \in \mathsf{MaxA}_\infty(s)$ such that $\limsup_{\mathsf{e}\to\infty} \mathsf{Lo}_{\mathsf{e}}(s, a_{\mathsf{e}}^{\max}) = \mathsf{Lo}_\infty(s, a_{\mathrm{Diff}}(s))$. Together, we have that $\mathrm{Diff}(s) = \mathsf{Up}_\infty(s, a_{\mathrm{Diff}}(s)) - \mathsf{Lo}_\infty(s, a_{\mathrm{Diff}}(s))$. Since all states S_∞ and all optimal actions MaxA_∞ are visited infinitely often, we have that $\mathsf{Up}_\infty(s,a) = \Delta(s,a)\langle \mathsf{Up}_\infty \rangle$ and $\mathsf{Lo}_\infty(s,a) = \Delta(s,a)\langle \mathsf{Lo}_\infty \rangle$ for all $s \in S_\infty$ and $a \in \mathsf{MaxA}_\infty(s)$ by the back-propagation in Lines 6 and 7. Consequently, we have that $\mathrm{Diff}(s) = \Delta(s, a_{\mathrm{Diff}}(s))\langle \mathrm{Diff} \rangle$ for all $s \in S_\infty$, since $a_{\mathrm{Diff}}(s) \in \mathsf{MaxA}_\infty(s)$ **(III)**.

Finally, we use Assumption 1 together with the above equation to show that $\mathrm{Diff}(\hat{s}) = 0$. Let now the maximal difference $\mathrm{Diff}_{\max} = \max_{s \in S_\infty} \mathrm{Diff}(s)$ and define the witness states $S_{\mathrm{Diff}} = \{s \in S_\infty \mid \mathrm{Diff}(s) = \mathrm{Diff}_{\max}\}$. Assume for a contradiction that we don't have

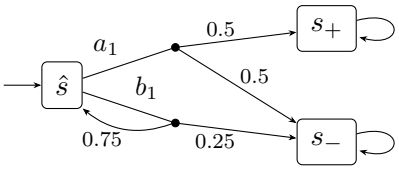

Figure 3.1: Example MDP where following the upper bounds is wrong.

$\mathrm{Diff} = 0$ (a.s.). For $\mathrm{Diff} > 0$ we clearly have $s_+, s_- \notin S_{\mathrm{Diff}}$, as $\mathrm{Diff}(s_+) = \mathrm{Diff}(s_-) = 0$ by Assumption 2 and Lemma 3. Consequently, S_{Diff} cannot contain any EC by Assumption 1. Since S_{Diff} does not contain an EC, there exists some state $s \in S_{\mathrm{Diff}}$ such that for all $a \in Av(s)$ we have $\mathrm{supp}(\Delta(s,a)) \not\subseteq S_{\mathrm{Diff}}$. In other words, for each action $a \in Av(s)$, there exists a state s_a with both $s_a \notin S_{\mathrm{Diff}}$ and $\Delta(s,a,s_a) > 0$. By definition of S_{Diff}, we have that $\mathrm{Diff}(s_a) < \mathrm{Diff}_{\max}$. In particular, we have that $\mathrm{Diff}(s, a_{\mathrm{Diff}}(s)) < \mathrm{Diff}(s)$ **(IV)**. For readability, we abbreviate the witness action obtained in **[III]** by $\overline{a} := a_{\mathrm{Diff}}(s)$. Then

$$\begin{aligned}
\mathrm{Diff}(s) &\overset{\text{[III]}}{=} \Delta(s,\overline{a})\langle \mathrm{Diff}_{\max}\rangle = \textstyle\sum_{s'\in S}\Delta(s,\overline{a},s')\cdot\mathrm{Diff}(s') \\
&\overset{\text{[II]}}{=} \textstyle\sum_{s'\in S_\infty}\Delta(s,\overline{a},s')\cdot\mathrm{Diff}(s') \\
&= \textstyle\sum_{s'\in S_\infty\setminus\{s_{\overline{a}}\}}\Delta(s,\overline{a},s')\cdot\mathrm{Diff}(s') + \Delta(s,\overline{a},s_{\overline{a}})\cdot\mathrm{Diff}(s_{\overline{a}}) \\
&\leq \textstyle\sum_{s'\in S_\infty\setminus\{s_{\overline{a}}\}}\Delta(s,\overline{a},s')\cdot\mathrm{Diff}_{\max} + \Delta(s,\overline{a},s_{\overline{a}})\cdot\mathrm{Diff}(s_{\overline{a}}) \\
&\overset{\text{[IV]}}{<} \textstyle\sum_{s'\in S_\infty\setminus\{s_{\overline{a}}\}}\Delta(s,\overline{a},s')\cdot\mathrm{Diff}_{\max} + \Delta(s,\overline{a},s_{\overline{a}})\cdot\mathrm{Diff}_{\max} \\
&= \mathrm{Diff}_{\max},
\end{aligned}$$

contradicting $s \in S_{\mathrm{Diff}}$ and we have that $\mathrm{Diff}_{\max} = 0$. To conclude the proof, observe that $S_{\mathrm{Diff}} = S_\infty$ a.s., as $0 \leq \mathrm{Diff}(s) \leq \mathrm{Diff}_{\max} = 0$ for all $s \in S_\infty$, and $\mathrm{Diff}(\hat{s}) = 0$ a.s., since $\hat{s} \in S_\infty$ a.s. by **[I]**.

The proof for guaranteed convergence is completely analogous. □

As an immediate consequence of Lemmas 4 and 5, we get the desired correctness.

Theorem 1. *Assume that Assumptions 1, 2, and (almost surely) 3 hold. Then Algorithm 2 is correct and converges (almost surely).*

Remark 6. If an implementation of SamplePairs satisfies Assumption 3 only almost surely, we can easily obtain a surely terminating variant by interleaving it with a deterministic sampling procedure, e.g., a round-robin method.

Example 1. Interestingly, following the optimal upper does not necessarily yield an ε-optimal strategy, as shown by the MDP in Figure 3.1. Assume that initially we take action a_1, setting $\mathsf{Up}_2(\hat{s},a_1) = \mathsf{Lo}_2(\hat{s},a_1) = \frac{1}{2}$. Then, $\mathsf{Up}_2(\hat{s},b_1) = 1 > \mathsf{Up}_2(\hat{s},a_1)$ and we sample b_1, updating $\mathsf{Up}_3(\hat{s},b_1) = \frac{3}{4}$, $\mathsf{Up}_4(\hat{s},b_1) = \frac{3}{4}\cdot\frac{3}{4}$, etc. This continues until the upper bound of b_1 is ε-close to $\frac{1}{2}$, when the algorithm terminates. Nevertheless, we will always have $\mathsf{Up}_\mathsf{e}(\hat{s},b_1) > \mathsf{Up}_\mathsf{e}(\hat{s},a_1)$. Note that one may have to adapt the transition probability $\Delta(\hat{s},b_1,s_-)$ depending on ε. It is straightforward to also apply this example to our DQL approach and as a counterexample to [Brá+14, Lemma 16]. △

Following the maximal *lower* bound yields a strategy achieving at least this value, using results on asynchronous VI [Put94]. We omit formal treatment of this claim, since we are not concerned with extracting a witness strategy. This would entail some technical difficulties in the general cases, which in turn distract from the central results.

4 Complete Information – General Case

In this section, we deal with the case of general MDP, in particular, we allow for arbitrary ECs. We first illustrate with an example the additional difficulties arising when considering general MDPs with non-trivial ECs. In particular, Algorithm 2 does not converge, even on a small example.

Example 2. Consider the MDP depicted in Figure 4.1. Clearly, we can reach the goal $T = \{s_+\}$ with probability $\frac{1}{2}$ by playing a_0 in $\hat{s}$ and then b_1 in s_1. But the EC $(\{\hat{s}, s_1\}, \{a_0, a_1\})$ causes issues for Algorithm 2. When running the algorithm on this example MDP, we eventually have that $\mathsf{Up}(s_1, b_1) = \mathsf{Lo}(s_1, b_1) = \frac{1}{2}$, but $\mathsf{Up}(s_1, a_1) = 1$, since $\mathsf{Up}(\hat{s}) = 1$. Similarly, we keep $\mathsf{Up}(\hat{s}, a_0) = 1$, as $\mathsf{Up}(s_1) = 1$. Informally, $\hat{s}$ and s_1 'promise' each other that the target state might still be reachable with probability 1, but these promises depend on each other cyclically. Removing the internal behaviour of this EC and 'merging' $\hat{s}$ and s_1 into a single state (with only action b_1) solves this issue.

In general, by definition of ECs, every state inside an EC can be reached from any other state with probability 1. Since we are interested in (unbounded) reachability, this means that for an EC there can only be two cases. Either, the EC contains a target state. Then, reaching any state of the EC is (a.s.) equivalent to reaching the target already and we do not need to treat the internal transitions of the EC further. Otherwise, i.e. when the EC does not contain a target state, we can also omit treatment of its internal behaviour and only consider its interaction with outside states. For the remainder of the section, fix an arbitrary MDP $\mathcal{M} = (S, Act, Av, \Delta)$, starting state $\hat{s}$, target set T, and precision $\varepsilon > 0$.

Lemma 6. *Let $(R, B) \in \mathrm{EC}(\mathcal{M})$ be an EC of $\mathcal{M}$. Then, $\mathsf{Pr}^{\max}_{\mathcal{M},s}[\Diamond\{s'\}] = 1$ for any states $s, s' \in R$ and consequently $\mathsf{Pr}^{\max}_{\mathcal{M},s}[\Diamond T] = \mathsf{Pr}^{\max}_{\mathcal{M},s'}[\Diamond T]$ for any target set $T \subseteq S$.*

Proof. Follows directly from [Cie+08, Lemma 1] (observe that the first claim is a special case of the second claim with $T = \{s'\}$). □

In other words, states in the same EC are equivalent for reachability and we can apply a quotienting construction w.r.t. to ECs. This idea has been exploited by the *MEC quotient* construction [De 97; Cie+08; HM14], a preprocessing step where first all MECs are identified and then 'collapsed' into a representative state. However, this approach requires that the whole graph structure of the MDP is known. Constructing the whole graph of the system may be prohibitively expensive or even impossible, as, e.g., in our limited knowledge setting (see Definition 8). Hence, we propose a modification to the BRTDP algorithm, which detects and handles ECs 'on-the-fly'. The algorithm will repeatedly identify ECs and maintain a separate, simplified MDP, which is similar to a MEC quotient.

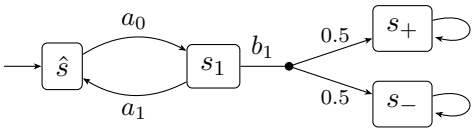

Figure 4.1: Example MDP with an EC where Algorithm 2 does not converge.

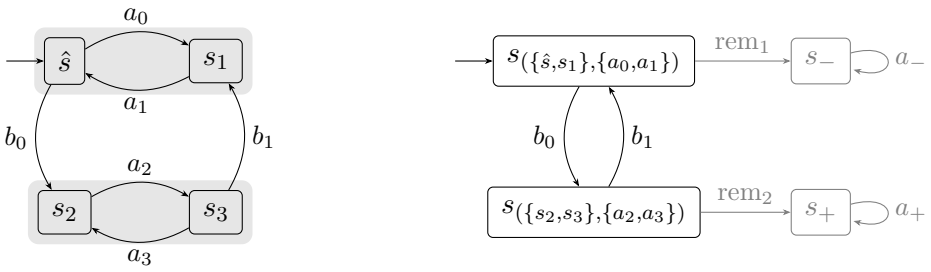

Figure 4.2: Example of an MDP (left) and its collapsed version (right) with $T = \{s_2\}$ and $\mathsf{EC} = \{(\{\hat{s}, s_1\}, \{a_0, a_1\}), (\{s_2, s_3\}, \{a_2, a_3\})\}$.

4.1 Collapsing End Components

As already explained, collapsing an EC can be viewed as replacing it with a single representative state, omitting the internal behaviour of the EC. In the following definition, we introduce the *collapsed MDP*, where end components are merged into representative states. Moreover, we again introduce the special states s_+ and s_-, acting as a target and sink respectively, to avoid corner cases. Many statements in this section are similar to [De 97, Section 6.4] but adapted to our particular use case. Note that our definition of collapsed MDP in particular depends on the target set T.

Definition 7. Let $\mathsf{EC} = \{(R_1, B_1), \ldots, (R_n, B_n)\} \subseteq \mathrm{EC}(\mathcal{M})$ be a (possibly empty) set of ECs in $\mathcal{M}$ with R_i pairwise disjoint and $B_i \neq \emptyset$. Define $R_{\mathsf{EC}} = \bigcup_i R_i$ and $B_{\mathsf{EC}} = \bigcup_i B_i$ the set of all states and actions in EC, respectively.

The *collapsed MDP* $\mathcal{M}^c = (S^c, Act^c, Av^c, \Delta^c) = \mathrm{collapse}(\mathcal{M}, \mathsf{EC}, \hat{s}, T)$ is obtained by

- $S^c = S \setminus R_{\mathsf{EC}} \cup \{s_{(R_i,B_i)}\} \cup \{s_+, s_-\}$, where $s_{(R_i,B_i)} \notin S$ are new *representative* states, s_+ is the new target state, and s_- is a new sink state,
- $Act^c = Act \setminus B_{\mathsf{EC}} \cup \{\mathrm{rem}_i\} \cup \{a_+, a_-\}$, where $\mathrm{rem}_i \notin Act$ are new *remain* actions,
- $Av^c(s)$ is defined by
  - $Av^c(s) = Av(s)$ for $s \in S \setminus R_{\mathsf{EC}}$,[(1)]
  - $Av^c(s_{(R_i,B_i)}) = \bigcup_{s \in R_i} Av(s) \setminus B_i \cup \{\mathrm{rem}_i\}$,
  - $Av^c(s_+) = \{a_+\}$, $Av'(s_-) = \{a_-\}$, and
- Δ^c is defined by
  - $\Delta^c(s^c, a^c, s'^c) = \sum_{s' \in \mathsf{states}(s'^c)} \Delta(\mathsf{state}(a^c, \mathcal{M}), a^c, s')$ for $s^c, s'^c \in S^c \setminus \{s_+, s_-\}$ and $a^c \in Av^c(s^c) \cap B$,
  - $\Delta^c(s_{(R_i,B_i)}, \mathrm{rem}_i) = \{s_+ \mapsto 1\}$ if $T \cap R_i \neq \emptyset$ and $\{s_- \mapsto 1\}$ otherwise, and

[(1)] Recall that actions in B_{EC} are only available for states in R_{EC}, hence $Av(s) \subseteq Act^c$ for other states.

– $\Delta^c(s_+, a_+, s_+) = 1$, $\Delta'(s_-, a_-, s_-) = 1$,

with the following auxiliary functions

- $\mathsf{collapsed} : S \to S^c$ maps states of $\mathcal{M}$ to their corresponding state in the collapsed MDP, i.e. $\mathsf{collapsed}(s) = s_{(R_i,B_i)}$ if $s \in R_i$ for some i and $\mathsf{collapsed}(s) = s$ otherwise,
- $\mathsf{states} : S^c \setminus \{s_+, s_-\} \to 2^S$ maps states in the collapsed MDP to the set of states they represent, i.e. $\mathsf{states}(s^c) = R_i$ if $s^c = s_{(R_i,B_i)}$ for some i and $\mathsf{states}(s^c) = \{s^c\} \subseteq S$ otherwise,
- $\mathsf{equiv} : S \to 2^S$ maps states of $\mathcal{M}$ to all states in their EC, i.e. $\mathsf{equiv}(s) = R_i$ if $s \in R_i$ for some i and $\mathsf{equiv}(s) = \{s\}$ otherwise.

For ease of notation, we extend these auxiliary functions to sets of states in the obvious way, i.e. $\mathsf{collapsed}(R) = \{\mathsf{collapsed}(s) \mid s \in R\}$, $\mathsf{states}(R^c) = \bigcup_{s^c \in R^c} \mathsf{states}(s^c)$, and $\mathsf{equiv}(R) = \bigcup_{s \in R} \mathsf{equiv}(s)$. Finally, if $\hat{s} \in R_i$ for some i, we identify $\hat{s}$ with $s_{(R_i,B_i)}$ for ease of notation. This guarantees that we always have $\hat{s} \in S^c$.

See Figure 4.2 for an example of a collapsed MDP. Observe that given a set EC explicitly, the collapsed MDP can be computed on-the-fly, without constructing the original MDP completely. In particular, given a state s in the MDP $\mathcal{M}$, we can compute the corresponding state $s^c = \mathsf{collapsed}(s)$, and given such a state s^c, we can directly compute $Av^c(s^c)$ and $\Delta^c(s^c, a^c)$ for all actions $a \in Av^c(s^c)$, based on the given set EC.

Now, we prove some useful properties about the collapsed MDP. These properties are rather intuitive, however the corresponding proofs are surprisingly technical and may be skipped. In essence, we prove that (i) there is a correspondence of paths between the original and the collapsed MDP, (ii) there is a correspondence of ECs between the two MDPs, and (iii) the reachability probability is equal on the two MDP.

For the remainder of this section, let $\mathcal{M}^c = (S^c, Act^c, Av^c, \Delta^c) = \mathrm{collapse}(\mathcal{M}, \mathsf{EC}, \hat{s}, T)$ be the collapsed MDP of $\mathcal{M}$, where $\mathsf{EC} = \{(R_i, B_i)\}_{i=1}^n$ is any appropriate set of end components.

Lemma 7. *We have that* $\mathsf{collapsed}(\mathsf{state}(a, \mathcal{M})) = \mathsf{state}(a, \mathcal{M}^c)$ *for all* $a \in Act \cap Act^c$.

Proof. First, observe that $Act \cap Act^c = Act^c \setminus \{a_+, a_-, \mathrm{rem}_i\}$ by definition. The claim follows by a case distinction on $s^c = \mathsf{state}(a, \mathcal{M}^c)$. If $s^c \in S$, then $Av(s^c) = Av^c(s^c)$ and $\mathsf{collapsed}(s^c) = s^c$. If instead $s^c = s_{(R_i,B_i)}$ for some $(R_i, B_i) \in \mathsf{EC}$, we have that $a \in \bigcup_{s \in R_i} Av(s) \setminus B_i$. Thus, there exists a state $s \in R_i$ such that $s = \mathsf{state}(a, \mathcal{M})$. But then by definition $\mathsf{collapsed}(s) = s^c$. □

The following two lemmas show how we can relate paths in the two MDPs with each other. See [De 97, Section 6.4.1] for an alternative view.

Lemma 8. *Let* $\varrho = s_1 a_1 \dots a_{n-1} s_n \in \mathsf{FPaths}_{\mathcal{M}}$ *be a finite path in the MDP* $\mathcal{M}$. *There exists a number* $m \leq n$ *and indices* $i_1, \dots, i_m$ *with* $1 \leq i_j < i_{j+1} \leq n$ *such that* $\varrho^c = \mathsf{collapsed}(s_{i_1}) a_{i_1} \dots a_{i_{m-1}} \mathsf{collapsed}(s_{i_m}) \in \mathsf{FPaths}_{\mathcal{M}^c}$ *is a finite path in the collapsed MDP* $\mathcal{M}^c$ *with* $\mathsf{collapsed}(s_1) = \mathsf{collapsed}(s_{i_1})$ *and* $\mathsf{collapsed}(s_n) = \mathsf{collapsed}(s_{i_m})$.

Proof. We construct the path ϱ^c inductively. Clearly, we start with $i_1 = 1$ and $s_1^c = \mathsf{collapsed}(s_1)$. Now, either all actions of ϱ are in B_{EC}, then by definition of ECs all states of ϱ are within the same EC and we are done. Otherwise, let a be the first action along the path ϱ such that $a \in Act^c$ (i.e. $a \notin B_{\mathsf{EC}}$) and let its index equal j. Set $i_2 = j$, $a_1^c = a$ and $s_2^c = \mathsf{collapsed}(s_{i+1})$. Clearly, $a \in Av(s_1^c)$. Repeat the argument with the path ϱ' equal to the suffix of ϱ starting at $j+1$. □

Lemma 9. *Let $\varrho^c = s_1^c a_1^c \dots a_{m-1}^c s_m^c \in \mathsf{FPaths}_{\mathcal{M}^c}$ be a finite path in the collapsed MDP $\mathcal{M}^c$ not containing the special states s_+, s_-. There exists a finite path $\varrho = s_1 a_1 \dots a_{n-1} s_n \in \mathsf{FPaths}_{\mathcal{M}}$ in the MDP $\mathcal{M}$ with $n \geq m$ and indices $i_1, \dots, i_m$ with $1 \leq i_j < i_{j+1} \leq n$ and*

- *$s_k \in \mathsf{states}(s_j^c)$ for all j and k with $i_j \leq k < i_{j+1}$ (defining $i_{m+1} = n+1$) and*
- *if $s_j^c = s_{(R_i,B_i)}$ then $a_k \in B_i$ for all j and k with $i_j \leq k < i_{j+1} - 1$.*

Proof. Similar to the above proof, we construct the path ϱ inductively. Distinguish two cases for s_1^c. If $s_1^c \in S$, set $s_1 = s_1^c$ and $a_1 = a_1^c$ and repeat the argument with the next step of ϱ^c. Otherwise, we have that $s_1^c = s_{(R_i,B_i)}$ for some EC $(R_i, B_i) \in \mathsf{EC}$. Since (R_i, B_i) is an EC in $\mathcal{M}$, there exists a finite path in $\mathsf{FPaths}_{\mathcal{M}}$ only using actions of B_i from any state in R_i to $\mathsf{state}(a_1^c, \mathcal{M})$. This path corresponds to the first state-action pair in ϱ^c. By definition, there exists a state $s' \in S$ such that $s' \in \mathrm{supp}(\Delta(\mathsf{state}(a_1^c, \mathcal{M}), a_1^c))$ and $\mathsf{collapsed}(s') = s_2^c$. Thus, we can extend the above path by $a_1^c s'$ and repeat the argument. □

Based on the previous lemmas, we can establish a correspondence of end components between the original MDP and its (partly) collapsed version. In particular, for every EC in the original MDP there either exists a single state representing this EC or a new EC in the collapsed MDP.

Lemma 10. *For any EC $(R, B) \in \mathrm{EC}(\mathcal{M})$ in the MDP $\mathcal{M}$ we either have*

1. *an EC (R^c, B^c) in $\mathcal{M}^c$, where $R^c = \mathsf{collapsed}(R)$ and $B^c = B \cap Act^c$, or*
2. *a state $s_{(R',B')} \in S^c$ with $R \subseteq R'$ and $B \subseteq B'$.*

Proof. Observe that Case 2 is trivial by definition, in particular this case is equivalent to $B \subseteq B_i$ for some i. Moreover, Case 1 and Case 2 are mutually exclusive since by construction for any $s_{(R_i,B_i)}$ there is no $B \subseteq B_i$ such that $(\{s_{(R_i,B_i)}\}, B)$ is an EC in $\mathcal{M}^c$. Let thus (R, B) be an EC in the MDP $\mathcal{M}$ with $B \not\subseteq B_i$ for all i. We show that (R^c, B^c) with $R^c = \mathsf{collapsed}(R)$ and $B^c = B \cap Act^c$ is an EC in $\mathcal{M}^c$.

First, we show by contradiction that $B \not\subseteq B_{\mathsf{EC}}$ **(I)**, i.e. B cannot comprise only internal actions of the ECs in EC. Recall that, by assumption, the EC states R_i are disjoint and B_i are subsets of the actions enabled in the respective states of R_i. If (R, B) is an EC with $B \not\subseteq B_i$ for all i, but $B \subseteq B_{\mathsf{EC}}$, (R, B) thus necessarily has to contain states of at least two ECs from EC. Formally, there exist two states $s, s' \in R$ with $s \in R_i$, $s' \in R_j$, and $i \neq j$. Since (R, B) is an EC, there exists a path from s to s' and vice versa, using only actions from B. As $B \subseteq B_{\mathsf{EC}}$, these actions were available in the ECs before already.

Since s and s' are in two, state disjoint ECs, there exists a state s'' and action a in EC (R_i, B_i) with $\mathrm{supp}(\Delta(s'', a)) \not\subseteq R_i$, contradicting the EC condition and proving **[I]**.

Furthermore, we prove that $R^c = \bigcup_{a \in B^c} \mathsf{state}(a, \mathcal{M}^c)$ **(II)**. Observe that by assumption we have $R = \bigcup_{a \in B} \mathsf{state}(a, \mathcal{M})$. By definition of $B^c = B \cap Act^c$, we thus have that $\bigcup_{a^c \in B^c} \mathsf{state}(a^c, \mathcal{M}) \subseteq R$. Consequently

$$\bigcup\nolimits_{a^c \in B^c} \mathsf{collapsed}(\mathsf{state}(a^c, \mathcal{M})) \subseteq \mathsf{collapsed}(R) = R^c$$

Applying Lemma 7 yields $\bigcup_{a^c \in B^c} \mathsf{collapsed}(\mathsf{state}(a^c, \mathcal{M})) = \bigcup_{a^c \in B^c} \mathsf{state}(a^c, \mathcal{M}^c)$, thus $\bigcup_{a^c \in B^c} \mathsf{state}(a^c, \mathcal{M}^c) \subseteq R^c$.

Now, assume for contradiction that there exists a state $s^c \in R^c$ such that $s^c \neq \mathsf{state}(a^c, \mathcal{M}^c)$ for all $a^c \in B^c$. Due to the definition of $\mathcal{M}^c$, we either have $s^c \in S$, $s^c = s_{(R', B')}$ for some EC $(R', B') \in \mathsf{EC}$, or $s^c \in \{s_+, s_-\}$. The third case immediately leads to a contradiction, since $B^c \subseteq Act$ and thus $a_+, a_- \notin B^c$. In the first case, we have that $s^c \notin R_i$ for any i, thus $Av(s^c) = Av^c(s^c) \subseteq Act^c$. Hence, any action a of this state contained in the EC (R, B) is still available in the collapsed MDP and thus also contained in the EC (R^c, B^c). The second case implies, by definition of $R^c = \mathsf{collapsed}(R)$, that there exists an EC $(R_i, B_i) \in \mathsf{EC}$ such that $R_i \cap R \neq \emptyset$. Recall that $Av^c(s_{(R_i, B_i)}) = \bigcup_{s \in R_i} Av(s) \setminus B_i$. The case assumption is thus equivalent to $B^c \cap (\bigcup_{s \in R_i} Av(s) \setminus B_i) = \emptyset$. Inserting the definition of B^c and Act^c yields

$$B \cap (Act \setminus B_{\mathsf{EC}}) \cap (\bigcup\nolimits_{s \in R_i} Av(s) \setminus B_i) = B \cap (\bigcup\nolimits_{s \in R_i} Av(s) \setminus B_i) = \bigcup\nolimits_{s \in R_i \cap R} Av(s) \cap B \setminus B_i = \emptyset.$$

This implies that $Av(s) \cap B \subseteq B_i$ for all $s \in R_i \cap R$, i.e. all such states only have 'internal' actions of the EC (R_i, B_i) available in (R, B). But this implies $R \subseteq R_i$ and $B \subseteq B_i$, contradicting our assumptions. This concludes the proof of **[II]**.

Now, we prove that (R^c, B^c) is a proper EC in $\mathcal{M}^c$, i.e. that (i) $R^c \neq \emptyset$, $\emptyset \neq B^c \subseteq \bigcup_{s^c \in R^c} Av(s^c)$, (ii) for all $s^c \in R^c$, $a \in B^c \cap Av^c(s^c)$ we have $\mathrm{supp}(\Delta^c(s^c, a^c)) \subseteq R^c$, and (iii) for all states $s^c, s'^c \in R^c$ there is a path from s^c to s'^c only using actions from B^c.

For (i), we have $B^c \neq \emptyset$, otherwise $B^c = B \cap Act^c = \emptyset$ implies $B \subseteq B_{\mathsf{EC}}$, contradicting **[I]**. **[II]** yields the second part of the first condition.

To prove (ii), assume a contradiction, i.e. let $s^c \in R^c$, $a \in B^c \cap Av^c(s^c)$ such that $s'^c \in \mathrm{supp}(\Delta^c(s^c, a^c)) \setminus R^c$. Let $s = \mathsf{state}(a^c, \mathcal{M})$ (implying $s^c = \mathsf{collapsed}(s)$). Again, we proceed by a case distinction, this time on the successor s'^c. If $s'^c \in S$, we have that $s'^c \in \mathrm{supp}(\Delta(s, a^c))$, since $s \in R$ and $a^c \in B$ and (R, B) is an EC. Further, $\Delta^c(s^c, a^c, s'^c) = \Delta(s^c, a^c, s'^c)$, thus $s'^c \in \mathrm{supp}(\Delta^c(s^c, a^c))$, contradicting the assumption. If instead $s'^c = s_{(R_i, B_i)}$, then there exists a state $s' \in \mathrm{supp}(\Delta(s, a^c)) \cap R_i$ by definition of Δ^c. But then $s_{(R_i, B_i)} \in R^c$ by definition of R^c, also yielding a contradiction.

Finally, to show (iii), we can directly apply Lemma 8 to obtain the required path as follows. Let $s^c, s'^c \in R^c$ two states and pick two arbitrary $s, s' \in R$ with $\mathsf{collapsed}(s) = s^c$ and $\mathsf{collapsed}(s') = s'^c$. Since (R, B) is an EC, there exists a finite path ϱ from s to s', using only actions of B. By Lemma 8, we get a path ϱ^c from s^c to s'^c using only actions

from $B \cap Act^c = B^c$, concluding the proof of Case 1. □

As expected, the corresponding reverse statement holds true, too, i.e. every EC in the collapsed MDP yields a corresponding EC in the original MDP.

Lemma 11. *For all ECs (R^c, B^c) in $\mathcal{M}^c$ with $s_+, s_- \notin R^c$ we have that (R, B) with $R = \mathsf{states}(R^c)$ and $B = B^c \cup \bigcup_{s_{(R_i,B_i)} \in R^c} B_i$ is an EC in $\mathcal{M}$.*

Proof. Fix an EC (R^c, B^c) in $\mathcal{M}^c$ and set $R = \mathsf{states}(R^c)$ and $B = B^c \cup \bigcup_{s_{(R_i,B_i)} \in R^c} B_i$. We need to prove that (R, B) is an EC in $\mathcal{M}$. Clearly, R and B are non-empty. We show that $R = \bigcup_{a \in B} \mathsf{state}(a, \mathcal{M})$. For any $s \in R$, there exists a $s^c \in R^c$ such that $s \in \mathsf{states}(s^c)$ by definition of R. If $s = s^c$ we have $s \in R^c$ and there exists an action $a^c \in B^c \subseteq B$ with $\mathsf{state}(a^c, \mathcal{M}) = s$. Otherwise, there is an EC $(R_i, B_i) \in \mathsf{EC}$ with $s \in R_i$, $s_{(R_i,B_i)} \in R^c$, and, since (R_i, B_i) is in EC in $\mathcal{M}$, there is an action $a \in B_i \subseteq B$ with $\mathsf{state}(a, \mathcal{M}) = s$. Similarly, for any action $a \in B$ we have that $\mathsf{state}(a, \mathcal{M}) \in R$ by analogous reasoning.

It remains to show that (i) for all $s \in R$, $a \in B \cap Av(s)$ we have $\mathrm{supp}(\Delta(s, a)) \subseteq R$, and (ii) for all states $s, s' \in R$ there is a finite path from s to s' only using actions from B. For (i), we again assume contradiction, i.e. there are states $s \in R$, $s' \in S$ and an action $a \in Av(s) \cap B$ such that $s' \in \mathrm{supp}(\Delta(s, a)) \setminus R$. We again proceed by case distinctions, but now first on a. If $a \in B^c$, then $\mathrm{supp}(\Delta^c(\mathsf{collapsed}(s), a)) \subseteq R^c$, since (R^c, B^c) is an EC. By definition of Δ^c, $\mathsf{collapsed}(s') \in \mathrm{supp}(\Delta^c(\mathsf{collapsed}(s), a))$. Together, this implies $s' \in R$, contradiction. If instead $a \in B_i$ for some EC $(R_i, B_i) \in \mathsf{EC}$, then $s, s' \in R_i \subseteq R$, contradiction. To prove (ii), we can directly apply Lemma 9 to a path from $\mathsf{collapsed}(s)$ to $\mathsf{collapsed}(s')$ in (R^c, B^c), yielding a path from s to s' in (R, B). □

The previous statement implies that if we collapse a MEC of the original MDP, then there can be no EC in the collapsed MDP containing the MEC representative state.

Lemma 12. *Let $\{(R'_i, B'_i)\}_{i=1}^m \subseteq \mathsf{EC} \cap \mathrm{MEC}(\mathcal{M})$ be some MECs of $\mathcal{M}$ in EC. Then, we have that $s_{(R'_i,B'_i)} \notin R^c$ for any EC (R^c, B^c) in $\mathcal{M}^c$.*

Proof. Assume there is such an EC (R^c, B^c) with $s_{(R'_i,B'_i)} \in R^c$. Lemma 11 yields an EC (R, B) with $R'_i \subseteq R$, $B'_i \subsetneq B$, contradiction to (R, B) being a MEC in $\mathcal{M}$. □

Observe that the statement of Lemma 12 does not hold for any EC $(R'_i, B'_i) \in \mathsf{EC}$, since there might be a larger EC containing $s_{(R'_i,B'_i)}$. For example, in Figure 4.2, the collapsed MDP has an EC containing representative states. However, if all MECs are collapsed, the resulting collapsed MDP indeed has no ECs (except the two trivial ones).

Corollary 1. *Let $\mathcal{M}^c = \mathrm{collapse}(\mathcal{M}, \mathrm{MEC}(\mathcal{M}), \hat{s}, T)$ be the collapsed MDP of $\mathcal{M}$ with $\mathsf{EC} = \mathrm{MEC}(\mathcal{M})$. Then, $\mathcal{M}^c$ satisfies Assumption 1.*

Proof. Follows directly from the above Lemma 12. □

Finally, we also get that the reachability probabilities are preserved.

Lemma 13. *Let* $\mathcal{M}^c = (S^c, Act^c, Av^c, \Delta^c) = \text{collapse}(\mathcal{M}, \mathsf{EC}, \hat{s}, T)$ *be the collapsed MDP of* $\mathcal{M}$*, where* $\mathsf{EC} = \{(R_i, B_i)\}_{i=1}^n$ *is any appropriate set of end components. Then, for any state* $s \in S$ *we have*

$$\mathsf{Pr}^{\max}_{\mathcal{M},s}[\Diamond T] = \mathsf{Pr}^{\max}_{\mathcal{M}^c,\mathsf{collapsed}(s)}[\Diamond \mathsf{collapsed}(T)] = \mathsf{Pr}^{\max}_{\mathcal{M}^c,\mathsf{collapsed}(s)}[\Diamond(\{s_+\} \cup (T \cap S^c))].$$

Proof. First, observe that $\mathsf{Pr}^{\max}_{\mathcal{M}^c,s^c}[\Diamond\{s_+\}] = 1$ for any state $s^c = s_{(R_i,B_i)}$ with $R_i \cap T \neq \emptyset$ by definition. Moreover, $T \cap S^c = T \setminus R_{\mathsf{EC}}$, i.e. all target states which are not part of an EC in EC. A state $s^c \in \mathsf{collapsed}(T)$ is of one of these two kinds. Hence, $\mathsf{Pr}^{\max}_{\mathcal{M}^c,\mathsf{collapsed}(s)}[\Diamond(\{s_+\} \cup (T \cap S^c))] = \mathsf{Pr}^{\max}_{\mathcal{M}^c,\mathsf{collapsed}(s)}[\Diamond \mathsf{collapsed}(T)]$, proving the second equality.

For the first equality, we argue how to transform the witness strategies, achieving the same overall reachability probability. Thus, let $\pi \in \Pi^{\mathsf{MD}}_{\mathcal{M}}$ be a (memoryless deterministic) strategy in $\mathcal{M}$ maximizing the probability of reaching T. We define a strategy π^c on $\mathcal{M}^c$ simulating π as follows. Note that π^c does not have to be memoryless or deterministic. For all states $s^c \in S$, i.e. s^c is not a collapsed representative, π^c mimics π, i.e. $\pi^c(s) = \pi(s)$. For the other case, namely $s^c = s_{(R_i,B_i)}$ for some EC $(R_i, B_i) \in \mathsf{EC}$, recall that π^c is allowed to have memory. In particular, it can remember the action a leading to s^c. Clearly, for any such action a and other action $a' \in Av^c(s^c)$ we can compute the probability of a' action being the first action not in B_i under π. Then, π^c simply selects a' in s^c after a with this probability. Moreover, we also need to compute the probability of remaining inside R_i forever, which corresponds to the probability of π^c choosing rem_i. It is easy to see that π^c achieves the same reachability as π.

If we instead start with a strategy in the collapsed MDP $\pi^c \in \Pi^{\mathsf{MD}}_{\mathcal{M}^c}$, we construct the respective strategy π on $\mathcal{M}$ as follows. Again, on states $s \notin R_{\mathsf{EC}}$, we simply replicate the choice of π^c. On states $s_{(R_i,B_i)}$ the strategy π^c chooses a single action $a^c \in Av^c(s_{(R_i,B_i)})$, since it is deterministic. If that action is rem_i, π simply picks any internal $a \in B_i$ in each state R_i. Otherwise, there exists a strategy π' on R_i reaching state $\mathsf{state}(a, \mathcal{M})$ with probability 1. Thus, π mimics π' until that state is reached and then plays a^c, again achieving the same reachability. □

4.2 The General BRTDP Algorithm

Now, we present our modification of Algorithm 2, using the idea of collapsing, to obtain the general approach as shown in Algorithm 3. On top of the previously presented ideas, the algorithm maintains a growing set of ECs and repeatedly collapses the input MDP.

The new auxiliary procedure UpdateECs is supposed to identify ECs in $\mathcal{M}$. As with SamplePairs, we only require some properties instead of giving a concrete implementation. Essentially, UpdateECs should only grow its list of ECs and eventually identify all ECs which are repeatedly visited by SamplePairs.

Assumption 4. Let $\mathsf{EC}_1 \subseteq \text{EC}(\mathcal{M})$ be any initial set of state-disjoint ECs, $\mathsf{EC}_{\mathsf{e}+1} = \text{UpdateECs}(\mathcal{M}, \mathsf{EC}_{\mathsf{e}})$ the identified ECs, and $\mathcal{M}^c_{\mathsf{e}} = \text{collapse}(\mathcal{M}, \mathsf{EC}_{\mathsf{e}}, \hat{s}, T)$ the corresponding collapsed MDPs. Then, for any episode e and EC $(R, B) \in \mathsf{EC}_{\mathsf{e}}$, (R, B) is an EC of $\mathcal{M}$ and there exists $(R', B') \in \mathsf{EC}_{\mathsf{e}+1}$ with $R \subseteq R'$ and $B \subseteq B'$.

Algorithm 3 The BRTDP learning algorithm for general MDPs.

Input: MDP $\mathcal{M}$, state $\hat{s}$, target set T, precision ε, initial bounds Up_1 and Lo_1, and initial set of ECs EC_1.
Output: ε-optimal values (l, u), i.e. $\mathcal{V}(\hat{s}) \in [l, u]$ and $0 \leq u - l < \varepsilon$.

1: $\mathsf{e} \leftarrow 1$, $\mathcal{M}^c_1 \leftarrow \mathrm{collapse}(\mathcal{M}, \mathrm{EC}_1, \hat{s}, T)$
2: $\mathsf{Up}_1(s_+, a_+) \leftarrow 1$, $\mathsf{Lo}_1(s_+, a_+) \leftarrow$, $\mathsf{Up}_1(s_-, a_-) \leftarrow 0$, $\mathsf{Lo}_1(s_-, a_-) = 0$
3: **while** $\mathsf{Up}_\mathsf{e}(\hat{s}) - \mathsf{Lo}_\mathsf{e}(\hat{s}) \geq \varepsilon$ **do**
4: **for all** $(R_j, B_j) \in \mathrm{EC}_\mathsf{e}$ **do** ▷ Initialize bounds of representative states
5: **for all** $a \in Av(s_{(R_j,B_j)}) \setminus \{\mathrm{rem}_j\}$ **do** ▷ Copy bounds for existing actions
6: $\mathsf{Up}_\mathsf{e}(s_{(R_j,B_j)}, a) \leftarrow \mathsf{Up}_\mathsf{e}(\mathsf{state}(a, \mathcal{M}), a)$
7: $\mathsf{Lo}_\mathsf{e}(s_{(R_j,B_j)}, a) \leftarrow \mathsf{Lo}_\mathsf{e}(\mathsf{state}(a, \mathcal{M}), a)$
8: **if** $R_j \cap T = \emptyset$ **then** ▷ Set bounds for remain action
9: $\mathsf{Up}_\mathsf{e}(s_{(R_j,B_j)}, \mathrm{rem}_j) \leftarrow 0$, $\mathsf{Lo}_\mathsf{e}(s_{(R_j,B_j)}, \mathrm{rem}_j) \leftarrow 0$
10: **else**
11: $\mathsf{Up}_\mathsf{e}(s_{(R_j,B_j)}, \mathrm{rem}_j) \leftarrow 1$, $\mathsf{Lo}_\mathsf{e}(s_{(R_j,B_j)}, \mathrm{rem}_j) \leftarrow 1$
12: $\mathsf{Up}_{\mathsf{e}+1} \leftarrow \mathsf{Up}_\mathsf{e}$, $\mathsf{Lo}_{\mathsf{e}+1} \leftarrow \mathsf{Lo}_\mathsf{e}$
13: $\varrho \leftarrow \textsc{SamplePairs}(\mathcal{M}^c_\mathsf{e}, \hat{s}, \mathsf{Up}_\mathsf{e}, \mathsf{Lo}_\mathsf{e}, \varepsilon)$ ▷ Sample a path in collapsed MDP
14: **for all** $(s, a) \in \varrho$ **do** ▷ Update the upper and lower bounds
15: $\mathsf{Up}_{\mathsf{e}+1}(s, a) \leftarrow \Delta(s, a)\langle \mathsf{Up}_\mathsf{e} \rangle$
16: $\mathsf{Lo}_{\mathsf{e}+1}(s, a) \leftarrow \Delta(s, a)\langle \mathsf{Lo}_\mathsf{e} \rangle$
17: $\mathrm{EC}_{\mathsf{e}+1} \leftarrow \textsc{UpdateECs}(\mathcal{M}, \mathrm{EC}_\mathsf{e})$ ▷ Search for new ECs
18: $\mathcal{M}^c_{\mathsf{e}+1} \leftarrow \mathrm{collapse}(\mathcal{M}, \mathrm{EC}_{\mathsf{e}+1}, \hat{s}, T)$ ▷ Update the collapsed MDP
19: $\mathsf{e} \leftarrow \mathsf{e} + 1$
20: **return** $(\mathsf{Lo}_\mathsf{e}(\hat{s}), \mathsf{Up}_\mathsf{e}(\hat{s}))$

Since there are only finitely many states, this assumption implies that eventually EC_e and thus $\mathcal{M}^c_\mathsf{e}$ stabilizes, i.e. there exists some episode $\bar{\mathsf{e}}$ such that for all $\mathsf{e} \geq \bar{\mathsf{e}}$ we have that $\mathsf{EC}_\mathsf{e} = \mathsf{EC}_{\mathsf{e}+1}$ and thus $\mathcal{M}^c_\mathsf{e} = \mathcal{M}^c_{\mathsf{e}+1}$. We call $\bar{\mathsf{e}}$ the *EC-stable episode*.

Assumption 5. Let EC_e and $\mathcal{M}^c_\mathsf{e}$ as in Assumption 4 and assume this assumption holds. Further, let $\varrho_\mathsf{e} \in \mathsf{FPaths}_{\mathcal{M}^c_\mathsf{e}}$ be an infinite series of sets of state-action pairs in $\mathcal{M}^c_\mathsf{e}$ and define $S^c_\infty = \bigcap_{k=1}^\infty \bigcup_{\mathsf{e}=k}^\infty \{s \in S^c_\mathsf{e} \mid s \in \varrho^c_\mathsf{e}\}$ the set of states occurring infinitely often.[2] Then, there exists no EC (R^c, B^c) in $\mathcal{M}^c_{\bar{\mathsf{e}}}$ with $R^c \subseteq S^c_\infty$ except $R^c = \{s_+\}$ or $R^c = \{s_-\}$.

4.3 Proof of Correctness

We now continue to prove correctness and termination of Algorithm 3. First, we argue that the algorithm indeed is well-defined, i.e. it never accesses undefined values.

Lemma 14. *Algorithm 3 is well-defined.*

Proof. We only need to show that the states introduced by the collapsing in Lines 1 and 18 are assigned bounds before being accessed. By definition of the collapsed MDP, we add a state for each EC together with an additional action, and the special states $\{s_+, s_-\}$. The initial collapse in Line 1 adds the special states together with their corresponding actions. Their values are initialized in the following line. Furthermore, the EC collapsing

[2] Observe that due to Assumption 4 we have $S^c_\infty \subseteq S^c_{\bar{\mathsf{e}}}$.

in Lines 1 and 18 adds a state $s_{(R,B)}$ for any EC $(R, B) \in \mathrm{EC}_{\mathsf{e}}$ and a corresponding rem action. Their values are initialized in Lines 4 to 11 and not accessed prior to that. □

For correctness, we again need to assume that the initial inputs are correct, similar to Assumption 2.

Assumption 6. The given initial bounds Up_1 and Lo_1 are correct, i.e. $\mathsf{Lo}_1(s,a) \leq \mathcal{V}(s,a) \leq \mathsf{Up}_1(s,a)$ for all $s \in S, a \in Av(s)$. Furthermore, the given initial set of ECs is correct, i.e. $\mathsf{EC}_1 \subseteq \mathrm{EC}(\mathcal{M})$ and pairwise disjoint.

Lemma 15. *Assume that Assumption 6 holds. Then, during any execution of Algorithm 3 we have for every episode* e*, all states* $s \in S_{\mathsf{e}}$ *and action* $a \in Av^c_{\mathsf{e}}(s)$ *that*

$$\mathsf{Lo}_{\mathsf{e}}(s,a) \leq \mathsf{Lo}_{\mathsf{e}+1}(s,a) \leq \mathcal{V}(s,a) \leq \mathsf{Up}_{\mathsf{e}+1}(s,a) \leq \mathsf{Up}_{\mathsf{e}}(s,a).$$

Proof. We prove that the initialization of values for newly added states is correct. The remaining proof then is completely analogous to the proof of Lemma 3.

Since s_+ is the target in $\mathcal{M}^c$, setting $\mathsf{Lo}_1(s_+, a_+) = 1$ is correct. Analogously, we see that s_- has no outgoing action and thus cannot reach s_+, justifying $\mathsf{Up}_1(s_-, a_-) = 0$.

The correctness of updates for the collapsed states follows from Lemma 13. □

Lemma 16. *The result of Algorithm 3 is correct under Assumption 6, i.e. (i)* $0 \leq u - l < \varepsilon$*, and (ii)* $\mathcal{V}(\hat{s}) \in [l, u]$*.*

Proof. As in Lemma 4, the claims follows directly from the definition of the algorithm and Lemma 15. □

Finally, we can prove termination of our presented algorithm. The proof is very similar to the proof of Lemma 5 and we only need to incorporate the new assumptions about UpdateECs.

Lemma 17. *Algorithm 3 terminates under Assumptions 3, 4, 5, and 6. It terminates almost surely if Assumption 3 is satisfied almost surely.*

Proof. We apply the same reasoning as in Lemma 5 until Assumption 1 is applied in the final part of the proof. Since we don't necessarily explore all of $\mathcal{M}$, $\mathcal{M}^c_{\mathsf{e}}$ may still contain MECs. In the proof, Assumption 1 is used only to show that $S_{\mathrm{Diff}} \subseteq S_\infty$ does not contain MECs. Observe that any non-terminating execution eventually reaches an EC-stable episode $\bar{\mathsf{e}}$, thus the collapsed MDP considered by the algorithm does not change. Now, S_∞ in the previous proof exactly corresponds to S^c_∞ of Assumption 5, which yields that again there is no EC in S^c_∞. Thus, we can continue to apply the previous proof's reasoning. □

Again, we get the overall soundness as a direct consequence.

Theorem 2. *Assume that (almost surely) Assumptions 3, 4, 5, and 6 hold. Then Algorithm 3 is correct and converges (almost surely).*

4.4 Relation to Interval Iteration

In this section, we briefly argue how our BRTDP algorithm presented in Algorithm 3 generalizes both the original BRTDP algorithm of [Brá+14] and the interval iteration algorithm of [HM14]. To this end, we give a brief overview of interval iteration. The algorithm first identifies all MECs and constructs a quotient similar to the one we presented in Section 4.1. Then, each state is initialized with straightforward upper and lower bounds. These bounds then are iterated globally according to the Bellman operator. We can emulate this behaviour by directly yielding the set of all MECs in UpdateECs and returning $S^c \times Av^c$ on each call to SamplePairs. All variants of [Brá+14] can be obtained by choosing the appropriate path sampling heuristics for SamplePairs.

5 Limited Information – MDP without End Components

We adapt our approach to the setting of limited information, where we can access the system only as a 'black box' and only are given some bounds on the shape of the system (see Section 2.3). Intuitively, since we are interested in an ε-precise solution, we can repeatedly sample the system to learn the transition probabilities with high confidence. By adapting our previous ideas, we can enhance this approach to only learn 'interesting' transitions. Since we can never bound the transition probabilities with absolute certainty, we aim for a *probably approximately correct* algorithm, which gives an ε-optimal solution with probability at least $1 - \delta$.

5.1 Definition of Limited Information

We formally define the limited information setting.

Definition 8. Let $\mathcal{M} = (S, Act, Av, \Delta)$ be some MDP, $\hat{s} \in S$ a starting state, and $T \subseteq S$ a target set. An algorithm has *limited information* if it can access

- the starting state $\hat{s}$,
- a target oracle for T, i.e. given a state s it can query whether $s \in T$,
- an upper bound K of the number of states, $K \geq |S|$,
- a lower bound q on the transition probabilities under any uniform strategy, $0 < q \leq p_{\min} = \min\{|Av(s)|^{-1} \cdot \Delta(s, a, s') \mid s \in S, a \in Av(s), s' \in \mathrm{supp}(\Delta(s, a))\}$,
- an oracle for the set of available actions Av, and
- a successor oracle succ, which given a state-action pair yields a successor state, sampled according to the underlying, hidden probability distribution Δ.

To solve this problem, we combine the BRTDP approach with *delayed Q-learning* (DQL) [Str+06]. In essence, DQL temporarily accumulates sampled values for each state-action pair and only attempts an update after a certain delay, i.e. after enough samples have been gathered for a particular pair. Intuitively, with a large enough delay, the average of the sampled values is close to the true average with high confidence. Moreover, the attempted update is only successful if the value is changed by at least some margin. If instead the update fails, another update is only allowed if any other value in the system has changed significantly. This way, we can bound the total number of attempted updates and thus control the overall probability of any 'wrong' update occurring. We explain all these ideas in more detail later on.

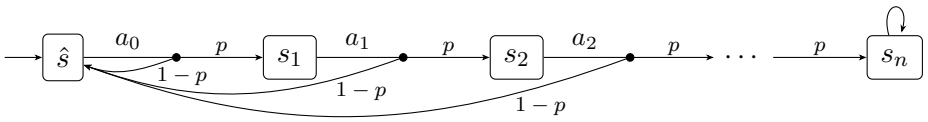

Figure 5.1: Example MDP to explain the choices and interpretations of some constants.

5.2 The No-EC DQL Algorithm

First, we again restrict ourselves to the case of no end components, as these pose an additional difficulty. Thus, we assume the MDP $\mathcal{M}$ satisfies Assumption 1 and instead of a target state oracle, the algorithm is explicitly given the special states s_+ and s_-. We present our DQL-based approach in Algorithm 4. While it is similar in spirit to Algorithm 2, we give a concrete instantiation of SamplePairs, since this setting needs a lot of additional guarantees. Note that we only store a single value per action, while the MDP potentially has $|S|$ transitions per action. As such, this algorithm satisfies the conditions of a *model-free* algorithm. As noted by [Str+06], the term 'model-free' has no standardized definition, and we instead give an intuitive explanation. Namely, that the algorithm's space complexity is asymptotically smaller than the model, thus algorithm is 'free' of knowledge of the model.

The algorithm contains several auxiliary variables. Most are values kept for each state-action pair, and separate for both the upper and lower bound. We give a brief intuition for each variable, where $\circ \in \{\mathsf{Up}, \mathsf{Lo}\}$ and (s, a) is a state-action pair in $\mathcal{M}$:

- $\mathtt{t}$: The number of steps the algorithm took so far, increased by 1 after each iteration of the main loop, as already mentioned in the preliminaries.
- $s_\mathtt{t}, a_\mathtt{t}, s'_\mathtt{t}$: The state, action, and the sampled successor state in step $\mathtt{t}$, respectively.
- $\mathsf{Up}_\mathtt{t}(s,a)$ and $\mathsf{Lo}_\mathtt{t}(s,a)$: The (estimated) upper and lower bounds for the state-action pair (s,a) at step $\mathtt{t}$. Note that in contrast to the previous algorithm, the upper and lower bounds are updated at each step instead of each episode.
- $\mathtt{learn}^\circ_\mathtt{t}(s,a)$: A three-valued flag (`yes`, `once`, or `no`) indicating whether the algorithm currently tries to learn and update the $\circ$-bounds for (s,a). The meaning of `once` is explained later on. We additionally use the Decrease function for convenience, which is defined by $\mathtt{yes} \mapsto \mathtt{once}$, $\mathtt{once} \mapsto \mathtt{no}$, and $\mathtt{no} \mapsto \mathtt{no}$.
- $\mathtt{count}^\circ_\mathtt{t}(s,a)$: The number of times a value for (s,a) was experienced. When $\mathtt{count}^\circ_\mathtt{t}(s,a)$ is large enough, we can attempt an update with sufficient confidence.
- $\mathtt{acc}^\circ_\mathtt{t}(s,a)$: The accumulated sampled values of the last $\mathtt{count}^\circ_\mathtt{t}(s,a)$ visits to (s,a). We want $\mathtt{acc}^\circ_\mathtt{t}(s,a)/\mathtt{count}^\circ_\mathtt{t}(s,a)$ to approximate the true $\circ$-bound.

Moreover, the algorithm contains the two constants $\overline{\varepsilon}$ and $\overline{m}$. We define their value (and the value of another constant, used for readability) as follows.

$$\overline{\varepsilon} = \frac{\varepsilon}{2} \cdot \frac{p_{\min}^{|S|}}{3|S|} \qquad \overline{\xi} = 2|Act| \left(1 + \frac{|Act|}{\overline{\varepsilon}}\right) \qquad \overline{m} = \frac{1}{2\overline{\varepsilon}^2} \ln\left(\frac{8}{\delta}\overline{\xi}\right)$$

Algorithm 4 The DQL learning algorithm for MDPs without ECs.

Input: Inputs as given in Definition 8 satisfying Assumption 1, special states s_+, s_-, precision ε, and confidence δ.

Output: Values (l, u) which are ε-optimal, i.e. $\mathcal{V}(\hat{s}) \in [l, u]$ and $0 \leq u - l < \varepsilon$, with probability at least $1 - \delta$.

$\mathsf{Up}_1(\cdot,\cdot) \leftarrow 1$, $\mathsf{Lo}_1(\cdot,\cdot) \leftarrow 0$, $\mathsf{Up}_1(s_-,\cdot) \leftarrow 0$, $\mathsf{Lo}_1(s_+,\cdot) \leftarrow 1$
for $\circ \in \{\mathsf{Up}, \mathsf{Lo}\}$ **do**
    $\mathtt{learn}^{\circ}_1(\cdot,\cdot) \leftarrow \mathtt{yes}$, $\mathtt{acc}^{\circ}_1(\cdot,\cdot) \leftarrow 0$, $\mathtt{count}^{\circ}_1(\cdot,\cdot) \leftarrow 0$
$\mathsf{e} \leftarrow 1, \mathsf{t} \leftarrow 1$
while $\mathsf{Up}_\mathsf{t}(\hat{s}) - \mathsf{Lo}_\mathsf{t}(\hat{s}) \geq \varepsilon$ **do**
    for $s \in S$ **do** $\mathsf{MaxA}_\mathsf{e}(s) \leftarrow \arg\max_{a \in Av(s)} \mathsf{Up}_\mathsf{t}(s,a)$
    $s_\mathsf{t} \leftarrow \hat{s}$
    while $s_\mathsf{t} \notin \{s_+, s_-\}$ **do** ▷ Experience the current learning episode
        $a_\mathsf{t} \leftarrow$ sampled uniformly from $\mathsf{MaxA}_\mathsf{e}(s_\mathsf{t})$ ▷ Pick an action
        $s'_\mathsf{t} \leftarrow \mathsf{succ}(s_\mathsf{t}, a_\mathsf{t})$ ▷ Query successor oracle
        ▷ Update bound estimates
        for $\circ \in \{\mathsf{Up}, \mathsf{Lo}\}$ **do**
            if $\mathtt{learn}^{\circ}_\mathsf{t}(s_\mathsf{t}, a_\mathsf{t}) \neq \mathtt{no}$ **then**
                $\mathtt{count}^{\circ}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow \mathtt{count}^{\circ}_\mathsf{t}(s_\mathsf{t}, a_\mathsf{t}) + 1$
                $\mathtt{acc}^{\circ}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow \mathtt{acc}^{\circ}_\mathsf{t}(s_\mathsf{t}, a_\mathsf{t}) + \bigcirc_\mathsf{t}(s'_\mathsf{t})$
        ▷ Learn upper bounds
        if $\mathtt{count}^{\mathsf{Up}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) = \overline{m}$ **then** ▷ Attempt update of Up
            if $\mathtt{acc}^{\mathsf{Up}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t})/\overline{m} < \mathsf{Up}_\mathsf{t}(s_\mathsf{t}, a_\mathsf{t}) - 2\overline{\varepsilon}$ **then**
                $\mathsf{Up}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow \mathtt{acc}^{\mathsf{Up}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t})/\overline{m} + \overline{\varepsilon}$ ▷ Successful update
                $\mathtt{learn}^{\mathsf{Up}}_{\mathsf{t}+1}(\cdot,\cdot) \leftarrow \mathtt{yes}$
            else
                $\mathtt{learn}^{\mathsf{Up}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow \textsc{Decrease}(\mathtt{learn}^{\mathsf{Up}}_\mathsf{t}(s_\mathsf{t}, a_\mathsf{t}))$ ▷ Failed update
            $\mathtt{count}^{\mathsf{Up}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow 0$, $\mathtt{acc}^{\mathsf{Up}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow 0$
        ▷ Learn lower bounds
        if $\mathtt{count}^{\mathsf{Lo}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) = \overline{m}$ **then** ▷ Attempt update of Lo
            if $\mathtt{acc}^{\mathsf{Lo}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t})/\overline{m} > \mathsf{Lo}_\mathsf{t}(s_\mathsf{t}, a_\mathsf{t}) + 2\overline{\varepsilon}$ **then**
                $\mathsf{Lo}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow \mathtt{acc}^{\mathsf{Lo}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t})/\overline{m} - \overline{\varepsilon}$ ▷ Successful update
                $\mathtt{learn}^{\mathsf{Lo}}_{\mathsf{t}+1}(\cdot,\cdot) \leftarrow \mathtt{yes}$
            else
                $\mathtt{learn}^{\mathsf{Lo}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow \textsc{Decrease}(\mathtt{learn}^{\mathsf{Lo}}_\mathsf{t}(s_\mathsf{t}, a_\mathsf{t}))$ ▷ Failed update
            $\mathtt{count}^{\mathsf{Lo}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow 0$, $\mathtt{acc}^{\mathsf{Lo}}_{\mathsf{t}+1}(s_\mathsf{t}, a_\mathsf{t}) \leftarrow 0$
        $s_{\mathsf{t}+1} \leftarrow s'_\mathsf{t}$, $\mathsf{t} \leftarrow \mathsf{t} + 1$ ▷ Increase step counter
    $\mathsf{e} \leftarrow \mathsf{e} + 1$ ▷ Increase episode counter
return $(\mathsf{Lo}_\mathsf{t}(\hat{s}), \mathsf{Up}_\mathsf{t}(\hat{s}))$

We call $\overline{\varepsilon}$ the *update step* (the smallest update considered by the algorithm), $\overline{\xi}$ the *update count* (the maximal possible number of update attempts, mainly introduced for readability), and $\overline{m}$ the *update delay* (the number of steps between updates). These three constants are used throughout this and the following section. Note that the update delay $\overline{m}$ is closely related to the worst-case *mixing rate* of the MDP, i.e. how fast information propagates through the system. This is illustrated in Figure 5.1. In order to propagate any information about state s_n to the initial state $\hat{s}$, we need $|S|$ steps. Moreover, only a fraction $p^{|S|}$ of the information is propagated after this many steps. Intuitively, this means that we need to visit a state-action pair often enough, i.e. $\overline{m}$ times, before an update to ensure that relevant information has propagated already. Dually, if a state-action pair was visited often enough and new information does not differ from the previous information by more than $\overline{\varepsilon}$, there is no new information to be propagated and we assume that the values of this state-action pair are converged.

Inside the main loop, the algorithm repeats two steps to obtain a path. First, an action maximizing the upper bounds (at the beginning of the episode) is randomly picked. More precisely, we again consider the set $\mathsf{MaxA}_{\mathsf{e}}(s) := \arg\max_{a \in Av(s)} \mathsf{Up}_{\mathsf{t}_{\mathsf{e}}}(s,a)$ and uniformly select an action thereof. To obtain the successor, we query the successor oracle with the given action to obtain the successor s'. In other words, in episode e the algorithm samples a path in the MDP using a memoryless strategy randomizing uniformly over $\mathsf{MaxA}_{\mathsf{e}}(s)$ in each state. We call this strategy the *sampling strategy* $\pi_{\mathsf{e}}(s,a) := |\mathsf{MaxA}_{\mathsf{e}}(s)|^{-1}$ if $a \in \mathsf{MaxA}_{\mathsf{e}}(s)$ and 0 otherwise. We will later on introduce the upper bound maximizing strategy π_{t}, which selects among Up-optimal actions at the current step t. Note that if the algorithm follow this strategy π_{t} while sampling, the samples would not be obtained from a memoryless strategy in general, since an update might happen while sampling and thus change the strategy. One might be tempted to solve this issue by first sampling a path until s_+ or s_- is reached and then propagating the values. However this path might be of exponential size; this already occurs for the structurally simple example in Figure 5.1.

After sampling a tuple (s,a,s'), the algorithm learns from this 'experience'. It does so by learning upper and lower bounds separately, depending on the respective `learn` flags, which are explained later. In case one of the bounds should be learned ($\mathtt{learn}^{\circ}_{\mathsf{t}}(s,a) \neq \mathtt{no}$), the accumulator is updated with the newly observed values, i.e. the respective bound of the successor s'. Furthermore, if the algorithm has gathered enough information, i.e. this pair has been experienced $\overline{m}$ times, an update of (s,a)'s estimate is attempted. Note that the update attempt may only happen when the respective `learn` is `yes` or `once`. By choosing $\overline{m}$ large enough, the information we gathered about the bounds of (s,a) is a faithful approximation of the true expected value over its successors. If the newly learned estimate, i.e. the average over the last $\overline{m}$ experiences of (s,a), significantly differs from the current estimate stored in Up or Lo, the current estimates are updated conservatively. If instead this new estimate is close to the current estimate, the algorithm marks this state-action pair as (potentially) converged by 'decreasing' its `learn` flag, as specified by the DECREASE function.

The learned bounds of a particular pair depend on the bounds of other state-action pairs. In particular, whenever some $\circ$-bound is changed anywhere, we may need to re-learn

the values for other state-action pairs. This is taken care of by globally resetting the learn flags to `yes` in Lines 20 and 28. This is the main difference to the subsequent algorithm [AKW19], where the samples are used instead to learn upper and lower bounds on the transition probabilities and the actual values are propagated according to these bounds, trading memory for speed of convergence.

The need for the intermediate value `once` of `learn` arises from the asynchronicity of the updates. Suppose an update of some pair (s, a) succeeds and we reset all learn values to `yes`. However, for some other state-action pair (s', a') we are very close to an update, too. Then, the values which will be used for an attempted update of (s', a') were mostly learned before the update of (s, a). However, if for example s is a successor of (s', a'), the values of (s', a') may be influenced significantly by the update of (s, a). Hence, we need to learn the value of (s', a') `once` more in order to be on the safe side. A different solution approach would be to simply reset all `count` and `acc` values after every successful update, however this would be much less efficient: If we again consider the above example, it might be the case that the values we gathered for (s', a') before the update of (s, a) already are sufficient for a successful update, discarding them would slow down convergence drastically.

5.3 Proof of Correctness

We now prove that the result returned by Algorithm 4 is probably approximately correct. We first prove correctness of the result by showing that the computed bounds are faithful upper and lower bounds in Lemma 21. However, we cannot guarantee that this always the case due to statistical outliers. Thus we first obtain bounds on the probability of these outliers. Then, in order to prove termination with high probability, we argue that by our choice of constants the propagation of values is probably correct. This means that whenever we update the bounds of a state-action pair (s, a), the updated value is close to the true average under $\Delta(s, a)$. Finally, we show that with high probability an update will occur as long as the bounds are not ε-close.

Lemma 18. *The number of successful updates of* Up *and* Lo *is bounded by* $\frac{|Act|}{\overline{\varepsilon}}$ *each.*

Proof. Let $a \in Act$ be some action and $s = \mathsf{state}(a, \mathcal{M})$ the associated state. The upper bound of (s, a) is initialized to 1 or 0, similar for the lower bound. Whenever $\mathsf{Up}_\mathsf{t}(s, a)$ is updated in Line 19, its value is decreased by at least $\overline{\varepsilon}$: We have that $\mathtt{acc}_\mathsf{t}^{\mathsf{Up}}(s, a)/m < \mathsf{Up}_\mathsf{t}(s, a) - 2\overline{\varepsilon}$, hence $\mathtt{acc}_\mathsf{t}^{\mathsf{Up}}(s, a)/m + \overline{\varepsilon} < \mathsf{Up}_\mathsf{t}(s, a) - \overline{\varepsilon}$. Thus, $\mathsf{Up}_{\mathsf{t}+1}(s, a) < \mathsf{Up}_\mathsf{t}(s, a) - \overline{\varepsilon}$. Analogously, $\mathsf{Lo}_\mathsf{t}(s, a)$ is always increased by at least $\overline{\varepsilon}$ whenever updated.

Moreover, $\mathtt{acc}_\mathsf{t}^{\mathsf{Up}}(s, a) \geq 0$ and $\mathtt{acc}_\mathsf{t}^{\mathsf{Lo}}(s, a) \leq \overline{m}$ by initialization and update of these values, hence we never set $\mathsf{Up}_\mathsf{t}(s, a)$ to a negative value and $\mathsf{Lo}_\mathsf{t}(s, a)$ is always smaller or equal to 1. Consequently, we change the value of $\mathsf{Up}_\mathsf{t}(s, a)$ and $\mathsf{Lo}_\mathsf{t}(s, a)$ at most $\frac{1}{\overline{\varepsilon}}$ times and there are at most $\frac{|Act|}{\overline{\varepsilon}}$ successful updates to the upper and lower bounds, respectively. Note that we do not necessarily have $\mathsf{Up}_\mathsf{t}(s, a) \leq \mathsf{Lo}_\mathsf{t}(s, a)$ for all executions of the algorithm, hence there are at most $\frac{|Act|}{\overline{\varepsilon}}$ updates for each of the bounds. □

Observe that this implies that for every execution, eventually there will be no more successful updates of Up and the sampling strategy π_e does not change. This fact will be

used in some of the subsequent proofs. Moreover, we can use the above result to show that similarly, the number of attempted updates is bounded.

Lemma 19. *The number of attempted updates of the upper bounds* Up *and lower bounds* Lo *is bounded by* $\overline{\xi} = 2|Act|(1 + \frac{|Act|}{\overline{\varepsilon}})$*, respectively.*

Proof. Let $(s, a) \in S \times Av$ be a state-action pair. Suppose an update of $\mathsf{Up}_\mathtt{t}(s, a)$ is attempted at step $\mathtt{t}$, i.e. $a_\mathtt{t} = a$, $\mathtt{count}_\mathtt{t}(s, a) = \overline{m} - 1$, and $\mathtt{learn}_\mathtt{t}^{\mathsf{Up}}(s, a) \neq \mathtt{no}$. Then, either the update is successful or $\mathtt{learn}_{\mathtt{t}+1}^{\mathsf{Up}}(s, a)$ is updated with DECREASE. The learn flag is only set to $\mathtt{yes}$ again if some other upper bound is successfully updated. Analogous reasoning applies to the lower bounds.

By Lemma 18, there are at most $\frac{|Act|}{\overline{\varepsilon}}$ successful updates to either bounds in total. If an update of a particular state-action pair is attempted, it either succeeds or fails. In the latter case, at most one more update of this state-action pair will be attempted until an other update succeeds. Hence, for a particular state-action pair (s, a) we have in the worst case two attempted Up-updates after every successful Up-update (of *any* pair). Together, there are at most $2 + 2\frac{|Act|}{\overline{\varepsilon}}$ (two more attempts can occur after the last successful update). Since there are $|Act|$ state-action pairs in total, the statement follows. □

Assumption 7. Suppose an Up-update of the state-action pair (s, a) is attempted at step $\mathtt{t}$. Let $k_1 < k_2 < \ldots < k_{\overline{m}} = \mathtt{t}$ be the steps of the $\overline{m}$ most recent visits to (s, a). Then $\frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \mathcal{V}(s'_{k_i}) \geq \mathcal{V}(s, a) - \overline{\varepsilon}$. Analogously, for an attempted Lo-update, we have $\frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \mathcal{V}(s'_{k_i}) \leq \mathcal{V}(s, a) + \overline{\varepsilon}$.

Lemma 20. *The probability that Assumption 7 is violated during the execution of Algorithm 4 is bounded by* $\frac{\delta}{4}$*.*

Proof. We show that the claim for the upper bound is violated with probability at most $\frac{\delta}{8}$. The lower bound part follows analogously and the overall claim via union bound.

Let (s, a) and k_i as in Assumption 7, i.e. an Up-update of (s, a) is attempted at step $k_m = \mathtt{t}$. First, observe that due to the Markov property, the successor state under (s, a) does not depend on the algorithm's execution. Hence, the states s'_{k_i}, i.e. the successor states after each visit of (s, a), are distributed i.i.d. according to the underlying probability distribution $\Delta(s, a)$. Define $Y_i = \mathcal{V}(s'_{k_i})$. Clearly, Y_i are i.i.d., since the actual value of a state $\mathcal{V}(s)$ is independent of the algorithm's execution. Moreover, $\mathbb{E}[Y_i] = \mathcal{V}(s, a)$, since $\mathcal{V}$ satisfies the fixed point conditions $\mathcal{V}(s, a) = \Delta(s, a)\langle\mathcal{V}\rangle$. Define the empirical average $\underline{Y} = \frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} Y_i$. Observe that $\mathbb{E}[\underline{Y}] = \frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \mathbb{E}[Y_i] = \mathcal{V}(s, a)$. By the Hoeffding bound [Hoe94] we have that

$$\mathbb{P}_\mathsf{A}\left[\mathbb{E}\left[\underline{Y}\right] - \underline{Y} > \overline{\varepsilon}\right] \leq e^{-2\overline{m}\overline{\varepsilon}^2} = \frac{\delta}{8} \cdot \overline{\xi}^{-1}$$

By reordering, we obtain that $\mathbb{P}_\mathsf{A}[\mathcal{V}(s, a) - \overline{\varepsilon} > \frac{1}{m} \sum_{i=1}^{m} \mathcal{V}(s_i)] \leq \frac{\delta}{8} \cdot \overline{\xi}^{-1}$. Consequently, by

employing the union bound and Lemma 19, we see that

$$\begin{aligned}
&\mathbb{P}_{\mathsf{A}}\left[\text{`}\frac{1}{\overline{m}}\sum_{i=1}^{\overline{m}}\mathcal{V}(s_{k_i}) < \mathcal{V}(s,a) - \overline{\varepsilon} \text{ for some } k_1\text{'}\right] \\
&\quad = \mathbb{P}_{\mathsf{A}}\left[\bigcup_{k_1}\text{`}\frac{1}{\overline{m}}\sum_{i=1}^{\overline{m}}\mathcal{V}(s_{k_i}) < \mathcal{V}(s,a) - \overline{\varepsilon} \text{ for } k_1\text{'}\right] \\
&\quad \leq \sum_{k_1}\frac{\delta}{8}\cdot\overline{\xi}^{-1} \leq \frac{\delta}{8}.
\end{aligned}$$

□

Lemma 21. *Assume that Assumption 7 holds. Then, during any execution of Algorithm 4 we have for every step* t*, all states* $s \in S_{\mathsf{e}}$ *and action* $a \in Av_{\mathsf{e}}(s)$ *that*

$$\mathsf{Lo}_{\mathsf{t}}(s,a) \leq \mathsf{Lo}_{\mathsf{t}+1}(s,a) \leq \mathcal{V}(s,a) \leq \mathsf{Up}_{\mathsf{t}+1}(s,a) \leq \mathsf{Up}_{\mathsf{t}}(s,a).$$

Proof. First, by definition of the algorithm we clearly have that Up can only decrease and Lo can only increase. It remains to show that $\mathsf{Lo}_{\mathsf{t}}(s,a) \leq \mathcal{V}(s,a) \leq \mathsf{Up}_{\mathsf{t}}(s,a)$. We proceed by induction on the step t. For $\mathsf{t} = 0$, the statement clearly holds, since $\mathsf{Up}_1(s,a) = 1$ for all states except the special state s_-, which by assumption cannot reach the target s_+. Analogously, the statement holds for $\mathsf{Lo}_1(s,a)$. Now, fix an arbitrary step t. We have that $\mathsf{Up}_{\mathsf{t}'}(s,a) \geq \mathcal{V}(s,a)$ for all steps $\mathsf{t}' \leq \mathsf{t}$ **(IH)**. Assume that (s,a) is the state-action pair sampled at step t. If no successful update takes place there is nothing to prove, since the values of Up and Lo do not change. Otherwise, Assumption 7 is applicable and we get

$$\mathsf{Up}_{\mathsf{t}+1}(s,a) = \frac{1}{\overline{m}}\sum_{i=1}^{\overline{m}}\mathsf{Up}_{k_i}(s_{k_i}) + \overline{\varepsilon} \overset{\textbf{[IH]}}{\geq} \frac{1}{\overline{m}}\sum_{i=1}^{\overline{m}}\mathcal{V}(s_{k_i}) + \overline{\varepsilon} \geq \mathcal{V}(s,a).$$

Analogously, we have $\mathsf{Lo}_{\mathsf{t}+1}(s,a) \leq \mathcal{V}(s,a)$. □

This gives us correctness of the returned result with high confidence if the algorithm terminates. It remains to show that the algorithm also terminates with high probability.

To this end, we introduce the upper bound maximizing strategy π_{t} which selects in each state s uniformly among all actions maximal with respect to the *current* upper bounds, i.e. $\mathsf{Up}_{\mathsf{t}}(s,\cdot)$. This allows us to reason about the current value at step t. Note that this strategy differs from the sampling strategy π_{e}, since π_{t} might change during an episode. However, once there are no updates to upper bounds, we have that $\pi_{\mathsf{e}} = \pi_{\mathsf{t}}$. We use this fact in the final convergence proof. Once the two strategies align, we can transfer properties proven with respect to π_{t} to the actual sampling behaviour of the algorithm.

Using this strategy, we define the set of nearly converged state-action pairs.

Definition 9. For every step t, define $\mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}, \mathcal{K}_{\mathsf{t}}^{\mathsf{Lo}} \subseteq S \times Av$ by

$$\begin{aligned}
\mathcal{K}_{\mathsf{t}}^{\mathsf{Up}} &:= \{(s,a) \mid \mathsf{Up}_{\mathsf{t}}(s,a) - \Delta(s,a)\langle\pi_{\mathsf{t}}[\mathsf{Up}_{\mathsf{t}}]\rangle \leq 3\overline{\varepsilon}\} \text{ and} \\
\mathcal{K}_{\mathsf{t}}^{\mathsf{Lo}} &:= \{(s,a) \mid \Delta(s,a)\langle\pi_{\mathsf{t}}[\mathsf{Lo}_{\mathsf{t}}]\rangle - \mathsf{Lo}_{\mathsf{t}}(s,a) \leq 3\overline{\varepsilon}\},
\end{aligned}$$

i.e. all state-action pairs whose Up- or Lo-value is close to the respective value of its successors under π_{t}. If $(s,a) \in \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$, we say that (s,a) is Up-*converged at step* t, analogously $(s,a) \in \mathcal{K}_{\mathsf{t}}^{\mathsf{Lo}}$ is called Lo-*converged at step* t.

The approach for the convergence proof is to show that (with high probability) (i) if an update of some bound fails, the current bound is consistent with its successors, i.e. the respective pair is converged, and (ii) we visit non-converged pairs only finitely often. These two facts then are combined non-trivially to obtain the convergence result.

Lemma 22. *We have for every step* t *and state* s *that*

$$\pi_{\mathsf{t}}[\mathsf{Up}_{\mathsf{t}}](s) = \mathsf{Up}_{\mathsf{t}}(s) \quad \textit{and} \quad \pi_{\mathsf{t}}[\mathsf{Lo}_{\mathsf{t}}](s) \leq \mathsf{Lo}_{\mathsf{t}}(s).$$

Moreover, if $(s,a) \notin \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$*, then* $(s,a) \notin \mathcal{K}_{\mathsf{t}'}^{\mathsf{Up}}$ *for all* $\mathsf{t}' > \mathsf{t}$ *until an* Up*-update of* (s,a) *succeeds. If no more updates of upper bounds take place, the analogous statement holds for the lower bounds, too.*

Proof. Since the strategy π_{t} maximizes the upper bound we have

$$\pi_{\mathsf{t}}[\mathsf{Up}_{\mathsf{t}}](s) = \sum\nolimits_{a \in Av(s)} \pi_{\mathsf{t}}(s,a) \cdot \mathsf{Up}_{\mathsf{t}}(s,a) = \max\nolimits_{a \in Av(s)} \mathsf{Up}_{\mathsf{t}}(s,a) = \mathsf{Up}_{\mathsf{t}}(s).$$

On the other hand, we trivially have that $\pi_{\mathsf{t}}[\mathsf{Lo}_{\mathsf{t}}](s) \leq \mathsf{Lo}_{\mathsf{t}}(s)$, as $\mathsf{Lo}_{\mathsf{t}}(s)$ is the maximum over all actions.

For the second claim, recall that Up-values can only decrease. If $(s,a) \notin \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$, we have $\mathsf{Up}_{\mathsf{t}}(s,a) > 3\overline{\varepsilon} + \Delta(s,a)\langle \pi_{\mathsf{t}}[\mathsf{Up}_{\mathsf{t}}] \rangle = 3\overline{\varepsilon} + \Delta(s,a)\langle \mathsf{Up}_{\mathsf{t}} \rangle$. Since (i) $\mathsf{Up}_{\mathsf{t}}(s,a) = \mathsf{Up}_{\mathsf{t}+1}(s,a)$ unless a successful Up-update of (s,a) occurs and (ii) $\mathsf{Up}_{\mathsf{t}}(s) \geq \mathsf{Up}_{\mathsf{t}+1}(s)$ for all states s, we obtain the claim.

The lower bound statement is proven analogously, noting that once upper bounds remain fixed the only way to change $\mathcal{K}_{\mathsf{t}}^{\mathsf{Lo}}$ is a successful update of some lower bound. □

Assumption 8. Suppose an update of the upper bound (lower bound) of the state-action pair (s,a) is attempted at step t. Let $k_1 < k_2 < \ldots < k_{\overline{m}} = \mathsf{t}$ be the steps of the $\overline{m}$ most recent visits to (s,a). If (s,a) is not Up-converged (Lo-converged) at step k_1, the update at step t is successful.

Intuitively, this assumption says that whenever the bound for a state-action pair is significantly different from its successors and we visit that pair often enough, we obtain a significantly better estimate. We cannot guarantee this surely due to outliers, but we bound the probability of this assumption being violated using our choice of the delay $\overline{m}$.

Lemma 23. *The probability that Assumption 8 is violated during the execution of Algorithm 4 is bounded by* $\frac{\delta}{4}$*.*

Proof. As in Lemma 20, we prove that such an attempted update of the upper bounds fails with probability at most $\frac{\delta}{8}$. The same bound can be obtained for the lower bound variant with a mostly analogous proof. Throughout the proof, we point out the key differences between the two proofs. The overall result follows using the union bound.

Let (s,a) and k_i as in Assumption 8, i.e. $(s,a) \notin \mathcal{K}_{k_1}^{\mathsf{Up}}$ and an update of the upper bound is attempted at step t **(I)**. Define $X_i = \pi_{k_1}[\mathsf{Up}_{k_1}](s'_{k_i})$. Observe that all X_i are defined

using Up_{k_1} and π_{k_1} (instead of Up_{k_i} and π_{k_i}) and thus are i.i.d. Again, we define the empirical average $\underline{X} = \frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} X_i$ and apply the Hoeffding bound to obtain

$$\mathbb{P}_{\mathsf{A}}[\underline{X} - \mathbb{E}[\underline{X}] \geq \overline{\varepsilon}] \leq e^{-2\overline{m}\overline{\varepsilon}^2} = \frac{\delta}{8} \cdot \overline{\xi}^{-1}.$$

Since the X_i are i.i.d., we have that $\mathbb{E}[\underline{X}] = \mathbb{E}[X_i]$ for all $1 \leq i \leq \overline{m}$, in particular $\mathbb{E}[\underline{X}] = \mathbb{E}[X_1]$. Thus, the probability that $\underline{X} - \mathbb{E}[X_1] \geq \overline{\varepsilon}$ is at most $\frac{\delta}{8} \cdot \overline{\xi}^{-1}$ **(II)**. For the lower bound proof, we analogously define $X_i = \pi_{k_1}[\mathsf{Lo}_{k_1}](s'_{k_i})$ and prove that $\mathbb{E}[X_1] - \underline{X} \geq \overline{\varepsilon}$ with the same probability.

Now, we show that if $\underline{X} - \mathbb{E}[X_1] < \overline{\varepsilon}$ the update at step t will be successful **(III)**. Recall that an update is successful when the $\overline{m}$ most recent samples significantly differ from the currently stored value, i.e. when the currently stored value $\mathsf{Up}_{\mathsf{t}}(s,a)$ is significantly larger than the newly learned value. We have that

$$\begin{aligned}
\mathsf{Up}_{\mathsf{t}}(s,a) - \frac{1}{m}\sum\nolimits_{i=1}^{\overline{m}} \mathsf{Up}_{k_i}(s'_{k_i}) &\geq \mathsf{Up}_{\mathsf{t}}(s,a) - \frac{1}{m}\sum\nolimits_{i=1}^{\overline{m}} \mathsf{Up}_{k_1}(s'_{k_i}) && (5.1)\\
&= \mathsf{Up}_{\mathsf{t}}(s,a) - \frac{1}{m}\sum\nolimits_{i=1}^{\overline{m}} \pi_{k_1}[\mathsf{Up}_{k_1}](s'_{k_i}) && (5.2)\\
&> \mathsf{Up}_{\mathsf{t}}(s,a) - \mathbb{E}[X_1] - \overline{\varepsilon} && (5.3)\\
&= \mathsf{Up}_{k_1}(s,a) - \mathbb{E}[X_1] - \overline{\varepsilon} && (5.4)\\
&= \mathsf{Up}_{k_1}(s,a) - \Delta(s,a)\langle \pi_{k_1}[\mathsf{Up}_{k_1}]\rangle - \overline{\varepsilon} && (5.5)\\
&> 2\overline{\varepsilon}. && (5.6)
\end{aligned}$$

Inequality 5.1 follows from the fact that Up-values can only decrease over time by definition of the algorithm. Equality 5.2 follows directly from Lemma 22. Inequality 5.3 follows from the above derivation. Equality 5.4 follows from the the fact that $\mathsf{Up}_{k_i}(s,a) = \mathsf{Up}_{k_1}(s,a)$ for all $1 \leq i \leq \overline{m}$, since an update is attempted at step $k_{\overline{m}} = t$ (and thus not prior to that point). Consequently, there can be no update attempts in the previous $\overline{m}-1$ visits and consequently the value of $\mathsf{Up}_{k_i}(s,a)$ does not change between k_1 and $k_{\overline{m}}$. Equality 5.5 follows directly from the definition of X_1. Finally, Inequality 5.6 follows from the assumption that (s,a) is not Up-converged at step k_1 **[I]**, i.e. $\mathsf{Up}_{k_1}(s,a) - \Delta(s,a)\langle \pi_{k_1}[\mathsf{Up}_{k_1}]\rangle > 3\overline{\varepsilon}$.

For the lower bound, we prove a similar result:

$$\begin{aligned}
\frac{1}{m}\sum\nolimits_{i=1}^{\overline{m}} \mathsf{Lo}_{k_i}(s'_{k_i}) - \mathsf{Lo}_{\mathsf{t}}(s,a) &\geq \frac{1}{m}\sum\nolimits_{i=1}^{\overline{m}} \mathsf{Lo}_{k_1}(s'_{k_i}) - \mathsf{Lo}_{\mathsf{t}}(s,a)\\
&\geq \frac{1}{m}\sum\nolimits_{i=1}^{\overline{m}} \pi_{k_1}[\mathsf{Lo}_{k_1}](s'_{k_i}) - \mathsf{Lo}_{\mathsf{t}}(s,a)\\
&> \mathbb{E}[X_1] - \overline{\varepsilon} - \mathsf{Lo}_{\mathsf{t}}(s,a)\\
&= \mathbb{E}[X_1] - \overline{\varepsilon} - \mathsf{Lo}_{k_1}(s,a)\\
&= \Delta(s,a)\langle \pi_{k_1}[\mathsf{Lo}_{k_1}]\rangle - \overline{\varepsilon} - \mathsf{Lo}_{k_1}(s,a)\\
&> 2\overline{\varepsilon}.
\end{aligned}$$

The only major difference lies in the second inequality (corresponding to Equality 5.2), where we instead use the fact that $\pi_{\mathsf{t}}[\mathsf{Lo}_{\mathsf{t}}](s) \leq \mathsf{Lo}_{\mathsf{t}}(s)$.

To conclude the proof, we extend the above argument to all steps k_1 satisfying the preconditions of the assumption. By Lemma 19, the number of attempted updates to Up and Lo is bounded by $\overline{\xi}$, respectively. Clearly, the number of such steps k_1 is bounded by the number of attempted updates **(IV)**. Together, we get that

$$\begin{aligned}
&\mathbb{P}_{\mathsf{A}}\left[\text{`Assumption 8 is violated for } \mathsf{Up}\text{'}\right] \\
&\quad= \mathbb{P}_{\mathsf{A}}\left[\bigcup\nolimits_{k_1} \text{`}k_1 \text{ satisfies condition } \mathbf{[I]}\text{, but the } \mathsf{Up}\text{-update fails'}\right] \\
&\quad\leq \sum\nolimits_{k_1} \mathbb{P}_{\mathsf{A}}\left[\text{`}k_1 \text{ satisfies condition } \mathbf{[I]}\text{, but the } \mathsf{Up}\text{-update fails'}\right] \\
&\quad\overset{\mathbf{[III]}}{\leq} \sum\nolimits_{k_1} \mathbb{P}_{\mathsf{A}}\left[\text{`}\underline{X} - \mathbb{E}[X_1] \geq \bar{\varepsilon} \text{ for } k_1\text{'}\right] \\
&\quad\overset{\mathbf{[II]}}{\leq} \sum\nolimits_{k_1} \frac{\delta}{8} \cdot \overline{\xi}^{-1} \overset{\mathbf{[IV]}}{\leq} \frac{\delta}{8}.
\end{aligned}$$

□

Lemma 24. *Assume that Assumption 8 holds. If an attempted* Up*-update of* (s,a) *at step* $\mathtt{t}$ *fails and* $\mathtt{learn}^{\mathsf{Up}}_{\mathtt{t}+1}(s,a) = \mathtt{no}$*, then* $(s,a) \in \mathcal{K}^{\mathsf{Up}}_{\mathtt{t}+1}$*. If no more updates of upper bounds take place, the analogous statement holds for the lower bounds, too.*

Proof. We prove the statement for the upper bound, with the corresponding lower bound statement following analogously. Assume an unsuccessful Up-update of (s,a) occurs at step $\mathtt{t}$ and let $k_1 < k_2 < \ldots < k_{\overline{m}} = \mathtt{t}$ be the $\overline{m}$ most recent visits to (s,a). We consider the following cases:

1. If $(s,a) \notin \mathcal{K}^{\mathsf{Up}}_{k_1}$, then by Assumption 8 the Up-update of (s,a) at step $\mathtt{t}$ will be successful and there is nothing to prove.

2. We have $(s,a) \in \mathcal{K}^{\mathsf{Up}}_{k_1}$ and there exists $i \in \{2,\ldots,\overline{m}\}$ such that (s,a) is not Up-converged at step k_i. It follows that there must have been a successful update of some Up-value between steps k_1 and $k_{\overline{m}}$, say step $\mathtt{t}'$. By Line 20, $\mathtt{learn}^{\mathsf{Up}}_{\mathtt{t}'+1}(s,a)$ is set to $\mathtt{yes}$ and there is nothing to prove.

3. For the last case, we have that for all $i \in \{1,\ldots,\overline{m}\}$ that (s,a) is Up-converged at step k_i, particularly $(s,a) \in \mathcal{K}^{\mathsf{Up}}_{k_{\overline{m}}} = \mathcal{K}^{\mathsf{Up}}_{\mathtt{t}}$. As the attempt to update the Up-value of (s,a) at step $\mathtt{t}$ was unsuccessful, we have that $\mathcal{K}^{\mathsf{Up}}_{\mathtt{t}} = \mathcal{K}^{\mathsf{Up}}_{\mathtt{t}+1}$.

For the proof of the lower bound statement, observe that $\mathcal{K}^{\mathsf{Lo}}_{\mathtt{t}}$ may be changed by a successful update of $\mathsf{Up}_{\mathtt{t}}$. Hence, the above reasoning can only be followed once upper bounds do not change. □

Lemma 25. *Assume that Assumption 8 holds. Then, there are at most* $2\overline{m} \cdot \frac{|Act|}{\bar{\varepsilon}}$ *visits to state-action pairs which are not* Up*-converged. Moreover, once the upper bounds are not updated any more, there are at most* $2\overline{m} \cdot \frac{|Act|}{\bar{\varepsilon}}$ *visits to state-action pairs which are not* Lo*-converged.*

Proof. We show that whenever a state-action pair (s,a) is not Up-converged at step $\mathtt{t}$, then in at most $2\overline{m}$ more visits to (s,a) a successful Up-update will occur. Assume that (s,a) is visited at step $\mathtt{t}$ and it is not Up-converged. We distinguish two cases.

1. $\texttt{learn}_{\mathsf{t}}^{\mathsf{Up}}(s,a) = \texttt{no}$: This implies that the last attempted Up-update of (s,a) was not successful. Let t' be the step of this attempt, $\mathsf{t}' < \mathsf{t}$. We have $\texttt{learn}_{\mathsf{t}'+1}^{\mathsf{Up}}(s,a) = \texttt{no}$. By Lemma 24, we have that $(s,a) \in \mathcal{K}_{\mathsf{t}'+1}^{\mathsf{Up}}$. Since we assumed $(s,a) \notin \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$, there was a successful update of some Up-value between t' and t, otherwise we would have $\mathcal{K}_{\mathsf{t}'+1}^{\mathsf{Up}} = \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$. Consequently, we have $\texttt{learn}_{\mathsf{t}+1}^{\mathsf{Up}}(s,a) = \texttt{yes}$. By Assumption 8 the next attempted to Up-update of (s,a) will be successful. This attempt will occur after $\overline{m}$ more visits to (s,a).

2. $\texttt{learn}_{\mathsf{t}}^{\mathsf{Up}}(s,a) \neq \texttt{no}$: By construction of the algorithm, we have that in at most $\overline{m}-1$ more visits to (s,a), an Up-update of (s,a) will be attempted. Suppose this attempt takes place at step t', $\mathsf{t}' \geq \mathsf{t}$ and the most $\overline{m}$ recent visits to (s,a) prior to t' happened at steps $k_1 < k_2 < \ldots < k_{\overline{m}} = \mathsf{t}'$. Note that we do not necessarily have that $\mathsf{t} = k_1$, but surely $\mathsf{t} \in \{k_1, \ldots, k_{\overline{m}}\}$. If the Up-update at step t' succeeds, there is nothing to prove, hence assume that this update fails. There are two possibilities:

   a) If (s,a) is not Up-converged at step k_1, then by Assumption 8 the Up-update at step t' will be successful, contradicting the assumption.

   b) If instead (s,a) is Up-converged at step k_1, we have that $\mathcal{K}_{k_1}^{\mathsf{Up}} \neq \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$, since we assumed that $(s,a) \notin \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$. Consequently, there was a successful Up-update of some other state-action pair at some step t'' with $k_1 < \mathsf{t}'' \leq \mathsf{t}$ and thus $\texttt{learn}_{\mathsf{t}''+1}^{\mathsf{Up}}(s,a) = \texttt{yes}$. Moreover, we necessarily have that no Up-update of (s,a) is attempted after t''. Together, we have that $\texttt{learn}_{\mathsf{t}'+1}^{\mathsf{Up}}(s,a) = \texttt{once}$ even though the attempted Up-update at step t' fails. By Lemma 22, we have that $(s,a) \notin \mathcal{K}_{\mathsf{t}'+1}^{\mathsf{Up}}$, as $(s,a) \notin \mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}$ and no successful Up-update of (s,a) occurred between t and t'. By Assumption 8 the next attempt to update Up-value of (s,a) will succeed.

By Lemma 18, the number of successful Up-updates is bounded by $\frac{|Act|}{\overline{\varepsilon}}$, and by the previous arguments we have that if for some t the pair (s,a) is not Up-converged then in at most $2\overline{m}$ more visits to (s,a), there will be a successful update to $\mathsf{Up}(s,a)$. Hence, there can be at most $2\overline{m} \cdot \frac{|Act|}{\overline{\varepsilon}}$ steps t such that the current state-action pair is not Up-converged. Once no more Up-updates take place, π_{t} remains fixed and $\mathcal{K}_{\mathsf{t}}^{\mathsf{Lo}}$ only changes due to successful updates of the lower bounds, yielding an analogous proof for Lo. □

As a last auxiliary lemma, we show that whenever the probability of reaching a non-converged pair is low, we necessarily are close to the optimal value.

Lemma 26. *Assume that Assumption 7 holds and fix a step* t*. Then, we have for every state* $s \in S$ *that*

$$\mathsf{Up}_{\mathsf{t}}(s) - 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} - \mathsf{Pr}_{\mathcal{M},s}^{\pi_{\mathsf{t}}}[\Diamond \overline{\mathcal{K}_{\mathsf{t}}^{\mathsf{Up}}}] \leq \mathsf{Pr}_{\mathcal{M},s}^{\pi_{\mathsf{t}}}[\Diamond\{s_+\}] \leq \mathsf{Lo}_{\mathsf{t}}(s) + 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} + \mathsf{Pr}_{\mathcal{M},s}^{\pi_{\mathsf{t}}}[\Diamond \overline{\mathcal{K}_{\mathsf{t}}^{\mathsf{Lo}}}].$$

Proof. The central idea of this proof is to apply Lemma 46 twice, with $X(s,a) = \mathsf{Up}_{\mathsf{t}}(s,a)$ and $X(s,a) = \mathsf{Lo}_{\mathsf{t}}(s,a)$, respectively.

For the first application, set $\kappa_l = -1$, $\kappa_u = 3\overline{\varepsilon}$, and $\pi = \pi_\mathsf{t}$. Then, $\mathcal{K} = \mathcal{K}_\mathsf{t}^{\mathsf{Up}}$ and

$$\mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}',s}[\Diamond\{s_+\}] - \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M},s}[\Diamond\overline{\mathcal{K}_\mathsf{t}^{\mathsf{Up}}}] \leq \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M},s}[\Diamond\{s_+\}] \tag{5.7}$$

since $\mathcal{M}'$ and $\mathcal{M}$ are equivalent on $\mathcal{K}_\mathsf{t}^{\mathsf{Up}}$. The lemma then yields that

$$\pi_\mathsf{t}[\mathsf{Up}_\mathsf{t}](s) - \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}'_t,s}[\Diamond\{s_+\}] \leq 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|}. \tag{5.8}$$

Recall that π_t is a strategy randomizing uniformly over some of the available actions in each state, hence $\delta_{\min}(\pi)$ is at least $p_{\min}$. For the second application, we dually set $\kappa_l = -3\overline{\varepsilon}$, $\kappa_u = 1$, and $\pi = \pi_\mathsf{t}$. Again, we have $\mathcal{K} = \mathcal{K}_\mathsf{t}^{\mathsf{Lo}}$ and

$$\mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M},s}[\Diamond\{s_+\}] \leq \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}',s}[\Diamond\{s_+\}] + \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M},s}[\Diamond\overline{\mathcal{K}_\mathsf{t}^{\mathsf{Lo}}}]. \tag{5.9}$$

The lemma gives us

$$\mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}'_t,s}[\Diamond\{s_+\}] - \pi_\mathsf{t}[\mathsf{Lo}_\mathsf{t}](s) \leq 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|}. \tag{5.10}$$

Now, recall that $\pi_\mathsf{t}[\mathsf{Up}_\mathsf{t}](s) = \mathsf{Up}_\mathsf{t}(s)$ and $\pi_\mathsf{t}[\mathsf{Lo}_\mathsf{t}](s) \leq \mathsf{Lo}_\mathsf{t}(s)$ **(I)** due to Lemma 22. Together, we have

$$\mathsf{Up}_\mathsf{t}(s) - 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} \overset{\textbf{(I)}}{=} \pi_\mathsf{t}[\mathsf{Up}_\mathsf{t}](s) - 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} \overset{(5.8)}{\leq} \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}'_t,s}[\Diamond\{s_+\}],$$

$$\mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}',s}[\Diamond\{s_+\}] - \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M},s}[\Diamond\overline{\mathcal{K}_\mathsf{t}^{\mathsf{Up}}}] \overset{(5.7)}{\leq} \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M},s}[\Diamond\{s_+\}] \overset{(5.9)}{\leq} \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}',s}[\Diamond\{s_+\}] + \mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M},s}[\Diamond\overline{\mathcal{K}_\mathsf{t}^{\mathsf{Lo}}}], \text{ and}$$

$$\mathsf{Pr}^{\pi_\mathsf{t}}_{\mathcal{M}'_t,s}[\Diamond\{s_+\}] \overset{(5.10)}{\leq} 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} + \pi_\mathsf{t}[\mathsf{Lo}_\mathsf{t}](s) \overset{\textbf{(I)}}{\leq} 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} + \mathsf{Lo}_\mathsf{t}(s). \qquad \square$$

Combining all the above statements now yields the overall result.

Theorem 3. *Algorithm 4 terminates and yields a correct result with probability at least* $1 - \delta$ *after at most* $\mathcal{O}(\mathrm{POLY}(|Act|, p_{\min}^{-|S|}, \varepsilon^{-1}, \ln \delta))$ *steps.*

Proof. We only consider executions where Assumptions 7 and 8 hold. By Lemmas 20 and 23 together with the union bound, this happens with probability at least $1 - \frac{\delta}{2}$.

Now, observe that if the algorithm terminates at some step t, we have that $\mathsf{Up}_\mathsf{t}(\hat{s}) - \mathsf{Lo}_\mathsf{t}(\hat{s}) < \varepsilon$ by definition. With Lemma 21, we have $\mathsf{Lo}_\mathsf{t}(\hat{s}) \leq \mathcal{V}(\hat{s}) \leq \mathsf{Up}_\mathsf{t}(\hat{s})$ and reordering yields the result.

We show by contradiction that the algorithm terminates for almost all considered executions. Thus, assume that the execution does not halt with non-zero probability. Since the MDP $\mathcal{M}$ satisfies Assumption 1, almost all episodes eventually visit either s_+ or s_- due to Lemma 2 and thus are of finite length. This implies that almost all executions for which the algorithm does not terminate comprise infinitely many episodes. We restrict our attention to only those executions.

Recall that due to Lemma 19, there are only finitely many attempted updates on almost all considered executions. Consequently, on these executions the algorithm eventually does not change Up, since no successful updates can occur from some step t onwards.

This means that all following samples are obtained by sampling according to the strategy π_{t}. Note that both the time of convergence and the actual strategy π_{t} depends on the execution $\mathfrak{a}$. Thus, we need to employ Lemma 48—the algorithm clearly qualifies as Markov process, since its evolution only depends on its current valuations. More precisely, it is not difficult to see that the whole execution of the algorithm (with fixed inputs) can be modelled as a (very unwieldy) countable Markov chain, showing that the considered properties are measurable. In particular, they are reachability objectives on this induced Markov chain.

Let us now consider the set of executions for which the upper bounds eventually converge and moreover $\mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Up}}_{\mathsf{t}}}] \geq \rho > 0$ infinitely often. Assume that this set of executions has a non-zero measure. By Lemma 48, on almost all of these executions $\overline{\mathcal{K}^{\mathsf{Up}}_{\mathsf{t}}}$ is also reached infinitely often, contradicting Lemma 25. For the lower bounds, we can prove a completely analogous statement. Consequently, $\mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Up}}_{\mathsf{t}}}] \to 0$ and $\mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{t}}}] \to 0$ on almost all considered executions.

Inserting the definition of $\overline{\varepsilon}$, we have for a sufficiently large step t that

$$\mathsf{Up}_{\mathsf{t}}(\hat{s}) - \frac{\varepsilon}{2} < \mathsf{Up}_{\mathsf{t}}(\hat{s}) - 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} - \mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Up}}_{\mathsf{t}}}]$$

and dually

$$\mathsf{Lo}_{\mathsf{t}}(\hat{s}) + 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} + \mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{t}}}] < \mathsf{Lo}_{\mathsf{t}}(\hat{s}) + \frac{\varepsilon}{2}$$

for all considered executions. Thus, by Lemma 26, we have

$$\mathsf{Up}_{\mathsf{t}}(\hat{s}) - \frac{\varepsilon}{2} < \mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M},\hat{s}}[\Diamond\{s_+\}] < \mathsf{Lo}_{\mathsf{t}}(\hat{s}) + \frac{\varepsilon}{2},$$

i.e. $\mathsf{Up}_{\mathsf{t}}(\hat{s}) - \mathsf{Lo}_{\mathsf{t}}(\hat{s}) < \varepsilon$, contradicting the assumption.

We have proven that the result is approximately correct with probability $1 - \frac{\delta}{2}$. Now, we additionally need to prove the step bound. To this end, we first bound the number of sampled paths and then bound the length of each path. Central to the following proof is Lemma 25, bounding the number of visits to non-converged state-action pairs. First, we treat the upper bounds. Observe that the probability of visiting an non-Up-converged state-action pair either is 0 or at least $p_{\min}^{|S|}$ (due to Lemma 43). Moreover, while this probability may fluctuate, once it reaches 0 it remains at 0, since then the sampling strategy does not change and all pairs reachable under this strategy are Up-converged. So, in the worst case, the probability of reaching such a pair is exactly $p_{\min}^{|S|}$ until they are visited often enough. We model this process as a series of Bernoulli trials X_i, equalling 1 if at least one Up-update happens while sampling the i-th path.[1] While the exact probabilities is not independent, they are always at least as large as the success probability $p \coloneqq p_{\min}^{|S|}$ of these trials (or 0 if all reachable pairs are Up-converged). Hence, we approximate the number of trials we need to perform until we observe at least $c \coloneqq 2\overline{m} \cdot \frac{|Act|}{\overline{\varepsilon}}$ successes with high probability—then, all upper bounds necessarily are converged by Lemma 25. Now, we

[1] We deliberately use i instead of e to emphasize that X_i does not operate on the probability space of the algorithm $(\mathfrak{A}, \mathcal{A}, \mathbb{P}_{\mathsf{A}})$. Instead, they represent a crude under-approximation to allow for a feasible analysis.

are essentially dealing with a binomially distributed variable $X_n = \sum_{i=1}^{n} X_i$ and want to find an n such that $\mathbb{P}[X_n \geq c] \geq 1 - \frac{\delta}{4}$. Since we are interested in the limit behaviour, we can apply the de Moivre–Laplace theorem, allowing us to replace this binomial distribution with an appropriate normal distribution. Thus, we obtain

$$\mathbb{P}[X_n \geq c] \approx 1 - \Phi\left(\frac{c - np}{\sqrt{np(1-p)}}\right),$$

and rearranging yields

$$n^{-\frac{1}{2}}(c - np) \approx \Phi^{-1}\left(\frac{\delta}{4}\right) \cdot \sqrt{p(1-p)}.$$

For readability, we set $a \coloneqq \Phi^{-1}\left(\frac{\delta}{4}\right)$. Solving for n gives us

$$n \approx \frac{c}{p} - \frac{a}{2p}\sqrt{(1-p)^2a^2 + 4c(1-p)} + \frac{(1-p)r^2}{2p}.$$

Inserting the definitions yields that $n \in \mathcal{O}(\mathrm{POLY}(|Act|, p_{\min}^{-|S|}, \varepsilon^{-1}, \ln \delta))$. This bounds the number of paths sampled by the algorithm. We furthermore prove that the length of all those paths is polynomial with high probability. To this end, we employ Lemma 44. Recall that sampling of a path stops once we reach one of the two special states s_+ and s_-. Due to Assumption 1, the probability of eventually reaching them is 1. Hence, $\mathsf{Pr}^{\pi_\mathsf{e}}_{\mathcal{M},\hat{s}}[\Diamond^{\leq N}\{s_+, s_-\}] \geq 1 - \tau$, where $N \geq \ln(\frac{2}{\tau}) \cdot |S| p_{\min}^{-|S|}$ for any sampling strategy π_e In other words, the probability of a sampled path being longer than N is at most τ. Then, by the union bound, the probability of any of the n paths being longer than N is at most $n \cdot \tau$. By choosing $\tau = \frac{\delta}{4n}$, this happens with probability at most $\frac{\delta}{4}$. Then, $\ln(\frac{2}{\tau}) = \ln(8n) - \ln(\delta)$, i.e. the length of each path again is bounded by a polynomial in the input values. Together, we obtain the results, since polynomials are closed under multiplication. □

6 Limited Information – General Case

As before, MECs pose an additional challenge, since they introduce superfluous upper fixed points. The key difference to the full information setting is that MECs cannot be directly identified. Instead, we identify a set of state-action pairs as an end component if it occurs sufficiently often. By bounding the probability of falsely identifying such a set as an end component, we can replicate the previous proof structure.

6.1 Collapsing End Components with Limited Information

Before we present the complete algorithm, we first show how we identify end components in this section.

Definition 10. Let $\mathcal{M} = (S, Act, Av, \Delta)$ be an MDP, $\rho \in \mathsf{Paths}_{\mathcal{M}}$ and $i, j \geq 0$. Define

$$Appear(\rho, i, j) = \{(s, a) \in S \times Av \mid |\{k \mid k \leq j \wedge \varrho^a(k) = a\}| \geq i\}$$

as the state-action pairs which appear at least i times on the path ρ during the first j steps. We overload the definition of *Appear* to also accept finite paths of sufficient length. Moreover, we also define *Appear* for paths of Markov chains, which yields the states occurring more than i times.

For notational convenience, we identify the result of *Appear* with the corresponding state-action tuple (R, B) since we will use these results as candidates for end components. If we choose i and j correctly, we can prove that *Appear* is an EC with high probability.

Lemma 27. *Let $\mathcal{M} = (S, Act, Av, \Delta)$ be an MDP, $\hat{s} \in S$ an initial state, $T \subseteq S$ a set of target states, and $\pi \in \Pi^{\mathsf{MD}}_{\mathcal{M}}$ a memoryless strategy on $\mathcal{M}$ such that $\mathsf{Pr}^{\pi}_{\mathcal{M},s}[\Diamond \overline{T}] = 0$ for all $s \in T$, i.e. T is absorbing under π. Set $S_\pi = \bigcup_{s \in S} \mathrm{supp}(\pi(s))$, $\kappa = |S_\pi| + 1$, and pick $i \geq \kappa$. Then either $\mathsf{Pr}^{\pi}_{\mathcal{M},\hat{s}}[\Diamond^{\leq 2i^3} T] = 1$ or*

$$\mathsf{Pr}^{\pi}_{\mathcal{M},\hat{s}}\Big[App_i \mid \overline{\Diamond^{\leq 2i^3} T}\Big] \geq 1 - 2(1 + i^2) \cdot e^{-(i-1)\frac{\delta_{\min}(\pi)^\kappa}{\kappa}} \cdot \delta_{\min}(\pi)^{-\kappa},$$

where $App_i = \{\rho \in \mathsf{Paths}_{\mathcal{M}} \mid Appear(\rho, i, 2i^3) \in \mathrm{EC}(\mathcal{M})\}$.

Informally, this lemma shows that, when sampling according to a memoryless strategy, paths of sufficient length either end up in an already known set of ECs or frequently reappearing state-action pairs also form an EC with high probability.

Proof. If $\mathsf{Pr}^{\pi}_{\mathcal{M},\hat{s}}[\Diamond^{\leq 2i^3} T] = 1$, there is nothing to prove, hence we assume the opposite, i.e. that $\mathsf{Pr}^{\pi}_{\mathcal{M},\hat{s}}[\overline{\Diamond^{\leq 2i^3} T}] > 0$ **(I)**.

Given an MDP, a designated initial state $\hat{s}$, and a memoryless strategy, we can construct a finite state Markov chain which exactly captures the behaviour of the MDP under the given strategy. We define the Markov chain $\mathsf{M}_\pi = (\{\hat{s}\} \cup S_\pi, \delta_\pi)$, where δ_π is defined as

$$\delta(\hat{s}, a) = \pi(\hat{s}, a) \text{ for } a \in \mathrm{supp}(\pi(\hat{s}))$$
$$\delta(a, a') = \Delta(\mathsf{state}(a, \mathcal{M}), a, \mathsf{state}(a', \mathcal{M})) \cdot \pi(\mathsf{state}(a', \mathcal{M}), a').$$

In other words, $\delta(a, a')$ equals the probability of reaching some state s' after playing action a and then continuing with action a'. As such, the paths in M_π exactly correspond to the paths in $\mathcal{M}$ following π. Furthermore, it is easy to see that each BSCC of M_π corresponds to an end component in $\mathcal{M}$. Observe that, by definition, κ equals the number of states in M_π **(II)** and $\delta_{\min}(\pi)$ equals the smallest positive transition probability in M_π **(III)**. For readability, we define $c = \exp(-\delta_{\min}(\pi)^\kappa/\kappa)$.

Let $\mathit{App}_{i,\pi} \subseteq \mathsf{Paths}_{\mathsf{M}_\pi}$ be the event corresponding to App_i in the Markov chain M_π. Informally, $\mathit{App}_{i,\pi}$ denotes the set of all (infinite) paths ρ which within $2i^3$ steps (i) visit all states of some BSCC at least i times, and (ii) all other states at most $i-1$ times, i.e. all paths such that $\mathit{Appear}(\rho, i, 2i^3)$ is a BSCC of M_π. We now show that

$$\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}[\mathit{App}_{i,\pi} \mid \overline{\Diamond^{\leq 2i^3} T}] \geq 1 - 2c^i i^3 \cdot \delta_{\min}(\pi)^{-\kappa},$$

i.e. the probability of $\mathit{App}_{i,\pi}$ given that T is not reached within $2i^3$ steps is at least $1 - 2c^i i^3 \cdot \delta_{\min}(\pi)^{-\kappa}$. Since the paths of M_π exactly correspond to paths obtained in $\mathcal{M}$ by following the strategy π, this proves the claim.

First, we show that **(IV)**

$$\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\Big[\mathit{App}_{i,\pi}\Big] \geq 1 - 2(1+i^2) \cdot c^{i-1}.$$

Let $B = \bigcup_{R \in \mathrm{BSCC}(\mathsf{M}_\pi)} R$ be the set of all states in BSCCs of M_π. We have that $\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}[\Diamond B] = 1$ by Lemma 1. We apply Lemma 44 with $N = i-1$ and $\tau = 2c^{i-1}$. By **[II]** and **[III]** we have

$$|S_\pi| \cdot \ln\left(\frac{2}{\tau}\right) \cdot \delta_{\min}(\pi)^{-|S_\pi|} = \kappa \cdot \ln\left(\exp\left((i-1) \cdot \frac{\delta_{\min}(\pi)^\kappa}{\kappa}\right)\right) \cdot \delta_{\min}(\pi)^{-\kappa} = i - 1.$$

Thus $\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}[\Diamond^{\leq i-1} B] \geq 1 - 2c^{i-1}$. In other words, an infinite path of M_π starting in $\hat{s}$ does not visit a BSCC of M_π within $i-1$ steps with probability at most $2c^{i-1}$.

Now, let $R = \{s_1, \ldots, s_n\} \subseteq B$ be some BSCC of M_π and fix two states $s_i, s_j \in R$. Since R is an BSCC, we have $\mathsf{Pr}_{\mathsf{M}_\pi,s_i}[\Diamond\{s_j\}] = 1$, and we can apply Lemma 44 again to obtain that $\mathsf{Pr}_{\mathsf{M}_\pi,s_i}[\Diamond^{\leq i}\{s_j\}] \geq 1 - 2c^{i-1}$. Consequently, the probability of visiting all states of R, one after another, with at most $i-1$ steps between visiting the respective next state, is at least $1 - n \cdot 2c^{i-1}$. Repeating this argument, with probability at least $1 - i \cdot n \cdot 2c^{i-1} \geq 1 - i \cdot \kappa \cdot 2c^{i-1}$, this round trip is successful i times in a row and has a length of at most $i \cdot n \cdot (i-1) \leq i^2\kappa \leq i^3$. Using the union bound again, we get that with probability at least $1 - 2c^{i-1} - i\kappa \cdot 2c^{i-1} = 1 - 2c^{i-1}(1 + i\kappa) \geq 1 - 2(1+i^2) \cdot c^{i-1}$ a path

of length i^3 ends up in a BSCC within $i-1$ steps and then visits all states of the BSCC at least i times, proving **[IV]**.

Let $T_\pi = \{a \in S_\pi \mid \mathsf{state}(a, \mathcal{M}) \in T\}$ the states of M_π corresponding to the given state set T. Recall that we assumed that $\mathsf{Pr}^\pi_{\mathcal{M},s}[\Diamond \overline{T}] = 0$ for $s \in T$, i.e. $\mathsf{Pr}_{\mathsf{M},a}[\Diamond \overline{T_\pi}] = 0$ for all $a \in T_\pi$. Consequently, each BSCC of M_π either is contained in T_π or disjoint from it: Assume that there exists a BSCC R with states $a, a' \in R$ where $a \in T_\pi$ and $a' \notin T_\pi$. Since R is a BSCC, we have $\mathsf{Pr}_{\mathsf{M}_\pi,a}[\Diamond\{a'\}] = 1$, contradicting $\mathsf{Pr}_{\mathsf{M}_\pi,a}[\Diamond \overline{T_\pi}] = 0$.

Due to **[I]**, there exists at least one BSCC which is disjoint from T_π—otherwise any run would eventually end up in T_π. Let s be some state in this BSCC. By construction, there exists a path of length at most κ **[II]** from $\hat{s}$ to s, and thus the probability of reaching such a BSCC is bounded from below by $\delta_{\min}(\pi)^\kappa$, using **[III]**. Formally, we have **(V)**

$$\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T_\pi}\right] > \delta_{\min}(\pi)^\kappa.$$

Finally, we obtain

$$\begin{aligned}
\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[App_{i,\pi} \mid \overline{\Diamond^{\leq 2i^3} T}\right] &\overset{\text{[I]}}{=} \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[App_{i,\pi} \cap \overline{\Diamond^{\leq 2i^3} T}\right] / \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] \\
&= \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[App_{i,\pi} \setminus \Diamond^{\leq 2i^3} T\right] / \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] \\
&= (\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[App_{i,\pi}\right] - \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[App_{i,\pi} \cap \Diamond^{\leq 2i^3} T\right]) / \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] \\
&\geq (\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[App_{i,\pi}\right] - \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\Diamond^{\leq 2i^3} T\right]) / \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] \\
&\overset{\text{[IV]}}{\geq} (1 - 2c^{i-1}(1+i^2) - (1 - \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right])) / \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] \\
&= (\mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] - 2c^{i-1}(1+i^2)) / \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] \\
&= 1 - (2c^{i-1}(1+i^2)) / \mathsf{Pr}_{\mathsf{M}_\pi,\hat{s}}\left[\overline{\Diamond^{\leq 2i^3} T}\right] \\
&\overset{\text{[V]}}{\geq} 1 - 2(1+i^2) \cdot c^{i-1} \cdot \delta_{\min}(\pi)^{-\kappa}.
\end{aligned}$$

□

6.2 The General DQL Algorithm

We define the general DQL algorithm in Algorithm 5. Essentially, the algorithm works similar to the previous Algorithm 4. The main difference is that it further employs Lemma 27 to detect whether the current sample is stuck in a yet to be discovered EC. To this end, the algorithm introduces a small set of additional auxiliary variables, necessary to track representative states similar to the collapsed MDP of Chapter 4. In particular, $\mathsf{collapsed_e}$ stores the representatives of each state. Since we might discover growing ECs, this representative might be part of another already discovered EC. Thus, we use $\mathsf{rep_e}$ to resolve the current representative of a given state s by repeatedly applying $\mathsf{collapsed_e}$ until a fixed point is reached. Additionally, $\mathsf{Z_e}$ contains all states which are part of a bottom EC without a target state. We choose the parameter i, controlling the length of each sample and when to check for an EC, such that

$$|Act| \cdot 2(1+\mathsf{i}^2) \cdot e^{-(\mathsf{i}-1)\frac{p_{\min}(\pi)^{|S|+1}}{|S|+1}} \cdot p_{\min}(\pi)^{-(|S|+1)} \leq \frac{\delta}{4} \qquad \text{and} \qquad \mathsf{i} \geq |Act|. \tag{6.1}$$

Algorithm 5 The DQL learning algorithm for general MDPs.

Input: Inputs as given in Definition 8, precision ε, and confidence δ.
Output: Values (l, u) which are ε-optimal, i.e. $\mathcal{V}(\hat{s}) \in [l, u]$ and $0 \leq u - l < \varepsilon$, with probability at least $1 - \delta$.

Initialize all variables as in Algorithm 4.
$\mathsf{e} \leftarrow 1, \mathsf{t} \leftarrow 1$
for $s \in S$ **do** $\mathsf{collapsed}_{\mathsf{e}}(s) \leftarrow s$
$S_1 \leftarrow S$, $Av_1 \leftarrow Av$, $T_1 \leftarrow T$, $\mathsf{Z}_1 \leftarrow \emptyset$
while $\mathsf{Up}_{\mathsf{t}}(\hat{s}) - \mathsf{Lo}_{\mathsf{t}}(\hat{s}) \geq \varepsilon$ **do**
    for $s \in S_{\mathsf{e}}$ **do** $\mathsf{MaxA}_{\mathsf{e}}(s) \leftarrow \arg\max_{a \in Av_{\mathsf{e}}(s)} \mathsf{Up}_{\mathsf{t}}(a)$
    $s_{\mathsf{t}} \leftarrow \hat{s}$, $\mathsf{t}_{\mathsf{e}} \leftarrow \mathsf{t}$
    while $s_{\mathsf{t}} \notin T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}}$ and $\mathsf{t} - \mathsf{t}_{\mathsf{e}} < 2\mathsf{i}^3$ **do**
        $a_{\mathsf{t}} \leftarrow$ sampled uniformly from $\mathsf{MaxA}_{\mathsf{e}}(s_{\mathsf{t}})$ ▷ Pick an action
        $s''_{\mathsf{t}} \leftarrow \mathsf{succ}(a_{\mathsf{t}})$ ▷ Query successor oracle
        $s'_{\mathsf{t}} \leftarrow \mathsf{rep}_{\mathsf{e}}(s''_{\mathsf{t}})$
        Perform updates as in Algorithm 4 ▷ Update Bounds
        $s_{\mathsf{t}+1} \leftarrow s'_{\mathsf{t}}$, $\mathsf{t} \leftarrow \mathsf{t} + 1$
    if $\mathsf{t} - \mathsf{t}_{\mathsf{e}} \geq 2\mathsf{i}^3$ **then** ▷ Update ECs
        $(\mathsf{R}, \mathsf{B}) \leftarrow Appear(s_{\mathsf{t}_{\mathsf{e}}} a_{\mathsf{t}_{\mathsf{e}}} s_{\mathsf{t}_{\mathsf{e}}+1} \dots a_{\mathsf{t}-1} s_{\mathsf{t}}, \mathsf{i}, 2\mathsf{i}^3)$
        $C \leftarrow \bigcup_{s \in \mathsf{R}} Av_{\mathsf{e}}(s) \setminus B$
        if $\mathsf{B} \neq \emptyset$ **then**
            if $T_{\mathsf{e}} \cap \mathsf{R} \neq \emptyset$ **then**
                $T_{\mathsf{e}+1} \leftarrow T_{\mathsf{e}} \cup \mathsf{R}$
                for $a \in \mathsf{B}$ **do** $\mathsf{Lo}_{\mathsf{t}}(a) \leftarrow 1$
            else if $C = \emptyset$ **then**
                $\mathsf{Z}_{\mathsf{e}+1} \leftarrow \mathsf{Z}_{\mathsf{e}} \cup \mathsf{R}$
                for $a \in \mathsf{B}$ **do** $\mathsf{Up}_{\mathsf{t}}(a) \leftarrow 0$
            else
                $S_{\mathsf{e}+1} \leftarrow (S_{\mathsf{e}} \setminus R) \cup \{s_{(R,B)}\}$
                $Av_{\mathsf{e}+1}(s_{(R,B)}) \leftarrow C$
                for $s \in \mathsf{R} \cup \{s_{(\mathsf{R},\mathsf{B})}\}$ **do** $\mathsf{collapsed}_{\mathsf{e}+1}(s) \leftarrow s_{(\mathsf{R},\mathsf{B})}$
                if $\hat{s} \in \mathsf{R}$ **then** $\hat{s} \leftarrow s_{(\mathsf{R},\mathsf{B})}$
    $\mathsf{e} \leftarrow \mathsf{e} + 1$
return $(\mathsf{Lo}_{\mathsf{t}}(\hat{s}), \mathsf{Up}_{\mathsf{t}}(\hat{s}))$

This technical choice becomes more apparent in the proof of Lemma 35. Note that such an i always exists since the left side of the first inequality converges to 0 for $\mathsf{i} \to \infty$. Moreover, we can find such an i using the values provided by the limited information setting as defined in Definition 8.

Remark 7. Note that in contrast to the previous sections, the domain of the upper bound Up and lower bound Lo functions are actions instead of state-action pairs. We deliberately make this change to simplify notation since the algorithm frequently changes the state associated with an action.

Remark 8. We implicitly assume that we can continue sampling with an action of our choice: When we collapse, for example, an EC (R, B) with states $s, s' \in \mathsf{R}$ into a single

representative state, we might enter the EC in state s but then continue sampling with an action $a \in Av(s')$. This is not an essential restriction: Upon entering an already detected EC, we can simply pause the algorithm and randomly pick actions in B until we reach the state enabling the next action mandated by the algorithm.

6.3 Proof of Correctness

Now, to prove correctness of the algorithm, we again can reuse a lot of the previous reasoning. However, we need to invest significant effort in the treatment of end components. First of all, we again prove that the algorithm is well-defined.

Lemma 28. *During all episodes, we have that $Av_{\mathsf{e}}(s) \cap Av_{\mathsf{e}}(s') = \emptyset$ for all states $s, s' \in S_{\mathsf{e}}$ with $s \neq s'$.*

Proof. The algorithm only modifies the set of available actions Av_{e} whenever a new representative state $s_{(R,B)}$ is added. In this case, we have $Av_{\mathsf{e}+1}(s_{(R,B)}) \leftarrow C \subseteq \bigcup_{s \in R} Av_{\mathsf{e}}(s)$ and all states of R are removed. □

Lemma 29. *Algorithm 5 is well defined.*

Proof. To prove this statement, we have to show that (i) no undefined values are accessed, (ii) all assignments are free of contradictions, and (iii) we require no more information than given by Definition 8.

For (i) and (ii), observe that when assigning the next episode's variables, we only use the variables of the current episode. Since we copy all unchanged variables, we only need to take care of the newly introduced arguments, i.e. the representative states $s_{(R,B)}$. Such a state is only added in Line 25. In the following lines, we define the state's actions Av, which is non-empty and disjoint from other states by Lemma 28. As no new actions are added, the action values in $s_{(R,B)}$ still are defined. Observe that in Line 10 the successor oracle is only given states of the original MDP. Claim (iii) follows immediately. □

Now, we show several statements related to the newly added handling of end components. Our goal is to show that the algorithm essentially samples from a collapsed MDP where the ECs identified by the algorithm are collapsed. Then, we replicate the proof ideas of the EC-free DQL algorithm on this collapsed MDP in order to again obtain the correctness.

Lemma 30. *Algorithm 5 enters Line 15 at most $|Act|$ times.*

Proof. First, observe that due to the pigeon-hole principle, B never is empty: By (6.1), our choice of i is larger than $|Act|$, thus a path of length at least i^2 contains at least one action i times. Consequently, whenever the algorithm enters Line 15, B is non-empty. Initially, the size of B is bounded by $\sum_{s \in S_1} |Av_1(s)| = |Act|$. We show that in any of the three cases, we remove at least one action which can never occur again as part of B. Consequently, after at most $|Act|$ visits to Line 15, B would necessarily be empty, contradicting the above.

Whenever a state is added to either T_{e} or Z_{e}, this state and its actions will not be considered again—in particular, it will not occur as part of B. For the third case, we

show that the number of available actions $\sum_{s \in S_\mathsf{e}} |Av_\mathsf{e}(s)|$ is reduced whenever a new representative state is added. In that case, we have $C \leftarrow \bigcup_{s \in R} Av_\mathsf{e}(s) \setminus B$, $S_{\mathsf{e}+1} \leftarrow (S_\mathsf{e} \setminus R) \cup \{s_{(R,B)}\}$, and $Av_{\mathsf{e}+1}(s_{(R,B)}) \leftarrow C$. By construction of the algorithm and definition of *Appear*, we have $\emptyset \neq B \subseteq \bigcup_{s \in R} Av_\mathsf{e}(s)$. Using Lemma 28 we thus have $|C| < |\bigcup_{s \in R} Av_\mathsf{e}(s)|$. Consequently, $\sum_{s \in S_{\mathsf{e}+1}} |Av_{\mathsf{e}+1}(s)| < \sum_{s \in S_\mathsf{e}} |Av_\mathsf{e}(s)|$. □

Lemma 31. *Algorithm 5 either terminates or experiences an infinite number of episodes.*

Proof. Since the length of each episode is limited, i.e. the loop of Line 8 always terminates after a bounded number of steps, we only need to show that all other loops terminate. All for-loops iterate over (sub-)sets of states or actions, which are finite by assumption. The only remaining loop is the computation of rep_e in Line 11, where the representative state is resolved. Observe that by construction of the algorithm, we either have that $\mathsf{collapsed}_\mathsf{e}(s) = s$ or $\mathsf{collapsed}_\mathsf{e}(s) = s_{(R,B)}$ with $s \in R$. Since we only modify $\mathsf{collapsed}$ when a new representative state is added, this happens only finitely often, due to Lemma 30. □

Lemma 32. *If we add a representative state $s_{(R,B)}$ in Line 25 after an episode e the bounds of any action $a \in B$ are not changed after episode e.*

Proof. During each episode e, we only consider states in S_e and actions which are available in such states, as the call to rep_e in Line 11 always yields an element of the current state set S_e due to Lemma 33. Since all states corresponding to actions in B are removed when adding a representative state $s_{(R,B)}$ and these actions are not enabled in the newly added state, they do not appear again. □

Lemma 33. *For any execution of the algorithm, we always have that $\mathsf{rep}_\mathsf{e}(s) \in S_\mathsf{e}$ for any state $s \in S$.*

Proof. Follows from a simple inductive proof: Initially, we have $\mathsf{rep}_1(s) = \mathsf{collapsed}_1(s) = s$ for all $s \in S_1$ by definition. Whenever we modify S_e, i.e. remove some states R and add a representative $s_{(R,B)}$, we set $\mathsf{collapsed}_{\mathsf{e}+1}(s) \leftarrow s_{(R,B)} \in S_{\mathsf{e}+1}$ for all $s \in R$. □

In order to properly reason about the paths sampled by the algorithm, we introduce a special MDP which corresponds to the current 'view' of the given MDP.

Definition 11. For any episode e, we define the *sampling MDP* $\mathcal{M}_\mathsf{e} = (S_\mathsf{e}, Act_\mathsf{e}, Av_\mathsf{e}, \Delta_\mathsf{e})$, where

$$\Delta_\mathsf{e}(s,a) = \{s \mapsto 1\} \quad \text{for } s \in S_\mathsf{e} \cap (T_\mathsf{e} \cup \mathsf{Z}_\mathsf{e}),\ a \in Av_\mathsf{e}(s), \text{ and}$$
$$\Delta_\mathsf{e}(s,a,s') = \sum\nolimits_{\{s'' \in S \mid \mathsf{rep}_\mathsf{e}(s'') = s'\}} \Delta(\mathsf{state}(a,\mathcal{M}), a, s'') \quad \text{for other states } s,\ a \in Av_\mathsf{e}(s),$$

and $Act_\mathsf{e} = \bigcup_{s \in S_\mathsf{e}} Av_\mathsf{e}(s)$.

Note that the sampling MDP is well-defined due to Lemmas 28 and 33.

Lemma 34. *Fix an execution of the algorithm until some episode e and let ϱ be the finite path sampled by the algorithm during episode e. The probability of sampling this path equals the probability of obtaining this path on $\mathcal{M}_\mathsf{e}$ following the strategy π_e starting in state $\hat{s}$.*

Proof. We prove by induction over the path ϱ, using the Markov property. In particular, we show that for any finite prefix, the probability of selecting action a and then reaching state s' in the next step is equal in both the algorithm and the sampling MDP. Observe that we always have $\hat{s} \in S_{\mathsf{e}}$ due to Line 28 and the induction start is trivial.

For the induction step, suppose we are in a state s. By construction of the algorithm, $s \notin T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}}$. The algorithm now uniformly selects an action a from $\mathsf{MaxA}_{\mathsf{e}}(s)$, i.e. with probability $|\mathsf{MaxA}_{\mathsf{e}}(s)|^{-1}$ for any such action. Then, a successor $s'' \in S$ is sampled according to $\mathsf{succ}(s,a)$, i.e. with probability $\Delta(s,a,s'')$. The overall successor then equals $s' = \mathsf{rep}_{\mathsf{e}}(s'')$. We have $s' \in S_{\mathsf{e}}$ by Lemma 33. Hence, a state $s' \in S_{\mathsf{e}}$ is sampled with probability $\sum_{\{s'' \in S \mid \mathsf{rep}_{\mathsf{e}}(s'') = s'\}} \Delta(s,a,s'')$, just as in the MDP $\mathcal{M}_{\mathsf{e}}$ under strategy π_{e}. □

Assumption 9. Whenever the algorithm reaches Line 15, (R, B) is an EC of $\mathcal{M}_{\mathsf{e}}$.

Lemma 35. *The probability that Assumption 9 is violated during the execution of Algorithm 5 is bounded by* $\frac{\delta}{4}$.

Proof. We apply Lemma 27 with $\mathcal{M} = \mathcal{M}_{\mathsf{e}}$, $T = T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}}$ and $\pi = \pi_{\mathsf{e}}$. By construction of $\mathcal{M}_{\mathsf{e}}$ and the choice of T, we have that π_{e} trivially satisfies the condition of this lemma, since each state in T only has self-loops in $\mathcal{M}_{\mathsf{e}}$. Clearly, we have that $|S_\pi| \leq \sum_{s \in S_{\mathsf{e}}} |Av(s)| \leq |Act|$, since no actions are added during the execution of the algorithm. Consequently, we have that either $\mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\lozenge^{\leq 2i^3}(T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}})] = 1$ or

$$\mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}\left[App_{\mathsf{i}} \mid \overline{\lozenge^{\leq 2i^3}(T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}})}\right] \geq 1 - 2(1+\mathsf{i}^2) \cdot e^{-(\mathsf{i}-1)\frac{p_{\min}(\pi)^{|S|+1}}{|S|+1}} \cdot p_{\min}(\pi)^{-(|S|+1)},$$

where App_{i} are all paths $\rho \in \mathsf{Paths}_{\mathcal{M}_{\mathsf{e}}}$ such that $Appear(\rho, \mathsf{i}, 2\mathsf{i}^3)$ is an EC in $\mathcal{M}_{\mathsf{e}}$.

Now, observe that the algorithm only enters Line 15 if after $2\mathsf{i}^3$ steps neither T_{e} nor Z_{e} is reached. By applying Lemma 34, we get that the probability of (R,B) being an EC given that Line 15 is entered exactly equals $\mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[App_{\mathsf{i}} \mid \overline{\lozenge^{\leq 2\mathsf{i}^3}(T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}})}]$. Since Line 15 is entered at most $|Act|$ times due to Lemma 30, the statement follows by inserting the definition of i from (6.1). □

Lemma 36. *Assume that Assumption 9 holds and fix some episode* e. *Let* $s \in S_{\mathsf{e}}$ *some state of the MDP* $\mathcal{M}_{\mathsf{e}}$ *and* $s' \in S$ *such that* $\mathsf{rep}_{\mathsf{e}}(s') = s$ *Then,* s *and* s' *have the same value, i.e.*

$$\mathcal{V}_{\mathsf{e}}(s) = \mathsf{Pr}^{\max}_{\mathcal{M}_{\mathsf{e}},s}[\lozenge T_{\mathsf{e}}] = \mathsf{Pr}^{\max}_{\mathcal{M},s'}[\lozenge T] = \mathcal{V}(s')$$

Proof. We prove by induction over the episode number. Initially, we have that $\mathcal{M}_1$ is quite similar to the original MDP $\mathcal{M}$. Recall that $\mathsf{Z}_1 = \emptyset$ and $\mathsf{rep}_1(s) = s$ for all states. Hence, the only difference lies in the transition function of all states $s \in T$. These only have self-loops in $\mathcal{M}_1$, while in $\mathcal{M}$ they may have arbitrary transitions. This is irrelevant for the value of the states, since it equals 1 in both cases.

Now fix an arbitrary episode e. We have that $\mathcal{V}_{\mathsf{e}}(s) = \mathcal{V}(s')$ **(IH)** for any two states s, s' as in the claim. $\mathcal{M}_{\mathsf{e}}$ is only modified when Line 15 is entered. Let (R,B) the identified set of states and actions. Due to Assumption 9, (R,B) is an EC of $\mathcal{M}_{\mathsf{e}}$. We distinguish the three cases in the algorithm:

- $T_{\mathsf{e}} \cap R \neq \emptyset$: Since (R, B) is an EC, any state $s \in R$ can reach T_{e} with probability one. Hence $\mathcal{V}_{\mathsf{e}+1}(s) = 1 = \mathcal{V}_{\mathsf{e}}(s) = \mathcal{V}(s')$ **[IH]**. In particular, by adding all states of R to $T_{\mathsf{e}+1}$, we do not change their value.

- $C = \emptyset$: In this case, once in R, this EC cannot be left, i.e. $\mathsf{Pr}^{\max}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond \overline{R}] = 0$ for all $s \in R$. Consequently, we have that $\mathcal{V}_{\mathsf{e}}(s) = 0 = \mathcal{V}(s')$ **[IH]**. This value is unchanged by adding the states of R to $\mathsf{Z}_{\mathsf{e}+1}$ and thus introducing a self-loop in $\mathcal{M}_{\mathsf{e}}$.

- Add a representative state: By assumption, we have that $\mathsf{rep}_{\mathsf{e}}(s') \in R$ and thus $\mathsf{rep}_{\mathsf{e}+1}(s') = s_{(R,B)}$. We need to prove that $\mathcal{V}_{\mathsf{e}+1}(s_{(R,B)}) = \mathcal{V}(s')$. As (R, B) is an EC by assumption, each state in R has the same value by Lemma 6. The representative state $s_{(R,B)}$ has this value by applying the same reasoning as in Lemma 13. □

Lemma 37. *Assume that Assumption 9 holds and fix some episode* e*. For any EC* $(R, B) \in \mathrm{EC}(\mathcal{M}_{\mathsf{e}})$ *there exists an EC* $(R', B') \in \mathrm{EC}(\mathcal{M}_{\mathsf{e}'})$ *with* $B \subseteq B'$ *for any* $\mathsf{e}' \leq \mathsf{e}$*.*

Proof. Note that we do not necessarily have that $R \subseteq R'$, since some states of the EC may have been replaced by a representative state.

We prove by induction on the episode e. Fix any such episode e and EC $(R, B) \in \mathrm{EC}(\mathcal{M}_{\mathsf{e}+1})$. We only modify the MDP $\mathcal{M}_{\mathsf{e}}$ when the algorithm enters Line 15, hence w.l.o.g. we assume that this happened in episode e. Let (R, B) be the set of states and actions identified in Line 15 during episode e. By Assumption 9, (R, B) is an EC of $\mathcal{M}_{\mathsf{e}}$. We distinguish the three cases in the algorithm:

- $T_{\mathsf{e}} \cap \mathsf{R} \neq \emptyset$: Then, all actions in B are changed to a self-loop in $\mathcal{M}_{\mathsf{e}+1}$ and hence we either have $B = \{a\} \subseteq \mathsf{B}$ or $B \cap \mathsf{B} = \emptyset$. In the former case, (R, B) satisfies the conditions of the claim. In the latter, the EC (R, B) already existed in $\mathcal{M}_{\mathsf{e}}$, since no state or action of (R, B) was modified.

- $C = \emptyset$: Analogously to the above, all actions in B are now a self-loop in $\mathcal{M}_{\mathsf{e}+1}$ and the same reasoning applies.

- Add a representative state: If $s_{(\mathsf{R},\mathsf{B})} \notin R$, we necessarily have that $\mathsf{B} \cap B = \emptyset$. Hence, the EC (R, B) again already existed in $\mathcal{M}_{\mathsf{e}}$, since none of its components was modified by this step. If instead $s_{(\mathsf{R},\mathsf{B})} \in R$, we have that $(\mathsf{R} \cup R, \mathsf{B} \cup B)$ is an EC in $\mathcal{M}_{\mathsf{e}}$, following the same reasoning as in Lemma 11. □

Lemma 38. *Assume that Assumption 9 holds and fix some step* t *with corresponding episode* e*. Let* $(R, B) \in \mathrm{EC}(\mathcal{M}_{\mathsf{e}})$ *be any EC in* $\mathcal{M}_{\mathsf{e}}$*. For any* $a \in B$ *we have that (i) if* $\mathsf{state}(a, \mathcal{M}_{\mathsf{e}}) \in \mathsf{Z}_{\mathsf{e}}$*, then* $\mathsf{Up}_{\mathsf{t}}(a) = 0$ *and (ii)* $\mathsf{Up}_{\mathsf{t}}(a) = 1$ *otherwise.*

Proof. Item (i) immediately follows from the definition of the algorithm and $\mathcal{M}_{\mathsf{e}}$. When a state is added to Z_{e}, we set $\mathsf{Up}_{\mathsf{t}}(a) = 0$ for all its actions. We prove Item (ii) by induction, showing that the statement holds for all ECs at each step t. Initially, we have $\mathsf{Up}_1(a) = 1$ for all actions by definition of the algorithm. For the induction step fix some step t. We have that $\mathsf{Up}_{\mathsf{t}'}(a) = 1$ for all actions a in all ECs without zero-states for all $\mathsf{t}' \leq \mathsf{t}$ **(IH)**. Now, let e' be the episode of step $\mathsf{t}+1$ and fix any EC (R, B) in $\mathcal{M}_{\mathsf{e}'}$ with $R \cap \mathsf{Z}_{\mathsf{e}} = \emptyset$. By

repeatedly applying Lemma 37, there exists an EC $(R_{\mathsf{e}'}, B_{\mathsf{e}'}) \in \mathrm{EC}(\mathcal{M}_{\mathsf{e}'})$ with $B \subseteq B_{\mathsf{e}'}$ for all $\mathsf{e}' \leq \mathsf{e}$. Since we have no zero-states in the EC in step $\mathsf{t}+1$, none of the $R_{\mathsf{e}'}$ contain zero-states either, by construction of the algorithm and $\mathcal{M}_{\mathsf{e}}$. Thus, the induction hypothesis [**IH**] is applicable and we have that $\mathsf{Up}_{\mathsf{t}'}(a) = 1$ for any action $a \in B_{\mathsf{e}'}$ and $\mathsf{t}' \leq \mathsf{t}$. Hence, we necessarily have that $\mathsf{Up}_{\mathsf{t}'}(s) = 1$ for all $s \in R_{\mathsf{e}'}$ and $\mathsf{t}' \leq \mathsf{t}$ (also using Lemma 32). Whenever any action $a \in B$ is selected at any step $\mathsf{t}' \leq \mathsf{t}$ during episode $\mathsf{e}' \leq \mathsf{e}$, all of its successors are part of the EC $(R_{\mathsf{e}'}, B_{\mathsf{e}'})$, thus $\mathsf{Up}_{\mathsf{t}'}(s) = 1$ for all successors by the above reasoning. Consequently, we always add a value of 1 to $\mathsf{acc}_{\mathsf{t}}^{\mathsf{Up}}(a)$ and whenever an Up-update is attempted for action a at some step $\mathsf{t}' \leq \mathsf{t}$, we would set $\mathsf{Up}_{\mathsf{t}'}(a) = 1$. □

Lemma 39. *Assume that Assumption 9 holds and fix some step* t *with corresponding episode* e. *Let* $\mathsf{t}' \geq \mathsf{t}$ *with episode* $\mathsf{e}' \geq \mathsf{e}$. *We have for any state* $s \in S$ *that* $\mathsf{Up}_{\mathsf{t}'}(\mathsf{rep}_{\mathsf{e}'}(s)) \leq \mathsf{Up}_{\mathsf{t}}(\mathsf{rep}_{\mathsf{e}}(s))$ *and* $\mathsf{Lo}_{\mathsf{t}}(\mathsf{rep}_{\mathsf{e}}(s)) \leq \mathsf{Lo}_{\mathsf{t}'}(\mathsf{rep}_{\mathsf{e}'}(s))$.

Proof. The bounds of actions are modified by (i) the usual update, which only increases or decreases, respectively (ii) in Lines 23 and 20, where upper bounds are set to 0 and lower bounds set to 1, or (iii) when an EC is collapsed and thus the set of available actions is modified in Line 26. Cases (i) and (ii) preserve monotonicity of the state bound by definition. Case (iii) is proven separately for upper and lower bounds, with the proof of the lower bound being significantly more involved. For the upper bounds, observe that $Av_{\mathsf{e}'}(s) \subseteq Av_{\mathsf{e}}(s)$ by definition, i.e. we never add new actions to any state. Consequently, the maximum over the set of available actions does not increase. For the lower bounds, we have to show that while collapsing ECs and thus removing actions, we never remove all those which are optimal w.r.t. the lower bound, i.e. all actions $a \in Av_{\mathsf{e}}(s)$ with $\mathsf{Lo}_{\mathsf{t}}(a) = \mathsf{Lo}_{\mathsf{t}}(s)$.

We proceed by additionally proving an auxiliary statement by induction on the step t in parallel. In particular, we prove that for any step t with corresponding episode e (i) the statement of the lemma holds **(IH1)** and (ii) $\mathsf{Lo}_{\mathsf{t}}(a) \leq \max_{s \in R, a' \in Av_{\mathsf{e}}(s) \setminus B} \mathsf{Lo}_{\mathsf{t}}(a')$ for all actions $a \in B$ (or 0 if no such actions a' exist) in all ECs $(R, B) \in \mathrm{EC}(\mathcal{M}_{\mathsf{e}})$ without a target state, i.e. $R \cap T_{\mathsf{e}} = \emptyset$. **(IH2)**.

Initially, we have $\mathsf{Lo}_1(a) = 0$ by definition of the algorithm and both statements trivially hold. For the induction step fix some time step t. We first treat the case when the lower bound of action an action a is successfully updated in step t and later on deal with the case of an EC being collapsed. Note that [**IH1**] trivially holds in this case, since the value of a is never decreased. We only need to show the second statement [**IH2**], thus assume that the updated action a is an internal action of some EC (R, B), i.e. $a \in B$. For readability, denote $C = \bigcup_{s \in R} Av_{\mathsf{e}}(s) \setminus B$ the set of outgoing actions of (R, B). If $C = \emptyset$, the statement follows directly: Since all lower bounds are initialized to zero, the EC does not contain any target states by assumption, and there are no outgoing actions, the algorithm never updates the lower bound of any action in B to a non-zero value. Thus, assume that $C \neq \emptyset$. By applying [**IH2**] to all states of the EC (R, B), we get that $\max_{a' \in C} \mathsf{Lo}_{\mathsf{t}}(a') = \max_{s \in R} \mathsf{Lo}_{\mathsf{t}}(s)$ **(I)**. Furthermore, let $k_1 < \ldots < k_{\overline{m}} = \mathsf{t}$ the steps of the most recent visits to a with corresponding episodes $\mathsf{e}_1 \leq \ldots \leq \mathsf{e}_{\overline{m}} = \mathsf{e}$ and sampled successors s'_{k_i}. Now, let $R_i = \mathsf{rep}_{\mathsf{e}_i}(\mathsf{states}_{\mathsf{e}}(R))$ for $1 \leq i \leq \overline{m}$ the set of states in episode e_i

which eventually are collapsed to R. By applying the reasoning of Lemma 37 and 11, there exists a set set of actions B_i with $B \subseteq B_i$ such that (R_i, B_i) is an EC in $\mathcal{M}_{\mathsf{e}_i}$ and thus $s'_{k_i} \in R_i$ **(II)**, since $a \in B_i$. By construction, we have that $\mathsf{rep}_{\mathsf{e}}(R_i) = R$ **(III)**. Finally, we observe that the value of the outgoing actions does not decrease, hence the value we assign to a in step t satisfies

$$\begin{aligned}
\mathsf{Lo}_{\mathsf{t}+1}(a) + \bar{\varepsilon} &\overset{\text{def}}{=} \frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \mathsf{Lo}_{k_i}(s'_{k_i}) \\
&\overset{\textbf{[II]}}{\leq} \frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \max_{s \in R_i} \mathsf{Lo}_{k_i}(s) \\
&\overset{\textbf{[IH1]}}{\leq} \frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \max_{s \in R_i} \mathsf{Lo}_{\mathsf{t}}(\mathsf{rep}_{\mathsf{e}}(s)) \\
&\overset{\textbf{[III]}}{=} \frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \max_{s \in R_{\overline{m}}} \mathsf{Lo}_{\mathsf{t}}(s) \\
&= \max_{s \in R_{\overline{m}}} \mathsf{Lo}_{\mathsf{t}}(s) \\
&\overset{\textbf{[I]}}{=} \max_{a' \in C_{\overline{m}}} \mathsf{Lo}_{\mathsf{t}}(a').
\end{aligned}$$

This concludes proof of the first part.

For the second part, i.e. when a set of states is collapsed by the algorithm, we have that the collapsed set (R, B) is an EC by Assumption 9 and B are only internal actions. If the collapsed EC contains target states, the statement trivially holds. Otherwise, we apply the result of the first part and get that the lower bound assigned to any action in B is less or equal to outgoing actions. Thus, removing the actions in B from the set of available actions does not reduce the value of the obtained representative state. □

With basic properties about the sampling MDP in place, we can now mimic the previous idea of defining 'converged' state-action pairs and, using those, show that the algorithm eventually converges with high probability.

Definition 12. For every step t during episode e, define $\mathcal{K}^{\mathsf{Up}}_{\mathsf{t}}, \mathcal{K}^{\mathsf{Lo}}_{\mathsf{t}} \subseteq Act_{\mathsf{e}}$ by

$$\begin{aligned}
\mathcal{K}^{\mathsf{Up}}_{\mathsf{t}} &:= \{a \mid \mathsf{Up}_{\mathsf{t}}(a) - \Delta_{\mathsf{e}}(\mathsf{state}(a, \mathcal{M}_{\mathsf{e}}), a)\langle \pi_{\mathsf{t}}[\mathsf{Up}_{\mathsf{t}}]\rangle \leq 3\bar{\varepsilon}\} \text{ and} \\
\mathcal{K}^{\mathsf{Lo}}_{\mathsf{t}} &:= \{a \mid \Delta_{\mathsf{e}}(\mathsf{state}(a, \mathcal{M}_{\mathsf{e}}), a)\langle \pi_{\mathsf{t}}[\mathsf{Lo}_{\mathsf{t}}]\rangle - \mathsf{Lo}_{\mathsf{t}}(a) \leq 3\bar{\varepsilon}\}.
\end{aligned}$$

Again, we say that action a is Up-*converged (*Lo-*converged) at step* t if $a \in \mathcal{K}^{\mathsf{Up}}_{\mathsf{t}}$ ($a \in \mathcal{K}^{\mathsf{Lo}}_{\mathsf{t}}$).

Assumption 10. Suppose an Up-update of the action a is attempted at step t. Let $k_1 < k_2 < \ldots < k_{\overline{m}} = \mathsf{t}$ be the steps of the $\overline{m}$ most recent visits to a, and $e_1 \leq e_2 \leq \ldots \leq e_{\overline{m}}$ the respective episodes. Then $\frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \mathcal{V}_{\mathsf{e}_i}(s'_{k_i}) \geq \mathcal{V}_{\mathsf{e}_{\overline{m}}}(a) - \bar{\varepsilon}$. Analogously, for an attempted Lo-update, we have $\frac{1}{\overline{m}} \sum_{i=1}^{\overline{m}} \mathcal{V}_{\mathsf{e}_i}(s'_{k_i}) \leq \mathcal{V}_{\mathsf{e}_{\overline{m}}}(a) + \bar{\varepsilon}$.

Assumption 11. Suppose an update of the upper bound (lower bound) of the action a is attempted at step t. Let $k_1 < k_2 < \ldots < k_{\overline{m}} = \mathsf{t}$ be the steps of the $\overline{m}$ most recent visits to a. If a is not Up-converged (Lo-converged) at step k_1, the update at step t is successful.

We replicate most of the statements from the previous DQL algorithm.

Lemma 40. *The following properties hold for Algorithm 5.*

1. *The number of successful updates of* Up *and* Lo *is bounded by* $\frac{|Act|}{\overline{\varepsilon}}$ *each.*

2. *The number of attempted updates of* Up *and* Lo *is bounded by* $\overline{\xi}$.

3. *Assume that Assumption 9 holds. Then, the probability that Assumption 10 is violated during the execution of Algorithm 4 is bounded by* $\frac{\delta}{4}$.

4. *Assume that Assumptions 9 and 10 hold. Then, we have* $\mathsf{Lo}_\mathsf{t}(a) \leq \mathcal{V}_\mathsf{e}(a) \leq \mathsf{Up}_\mathsf{t}(a)$ *for all episodes* e*, steps* $\mathsf{t} \geq \mathsf{t}_\mathsf{e}$*, and actions* $a \in Act_\mathsf{e}$.

5. *We have for every step* t *in episode* e *and state* $s \in S_\mathsf{e}$ *that*

$$\pi_\mathsf{t}[\mathsf{Up}_\mathsf{t}](s) = \mathsf{Up}_\mathsf{t}(s) \quad \textit{and} \quad \pi_\mathsf{t}[\mathsf{Lo}_\mathsf{t}](s) \leq \mathsf{Lo}_\mathsf{t}(s).$$

6. *If* $a \notin \mathcal{K}^{\mathsf{Up}}_\mathsf{t}$*, then* $a \notin \mathcal{K}^{\mathsf{Up}}_{\mathsf{t}'}$ *for all* $\mathsf{t}' \geq \mathsf{t}$ *until an* Up*-update of action* a *succeeds or the upper bound is set to* 0 *in Line 23.*

7. *The probability that Assumption 11 is violated during the execution of Algorithm 4 is bounded by* $\frac{\delta}{4}$.

8. *Assume that Assumption 11 holds. If an attempted* Up*-update of action* a *at step* t *fails and* $\mathtt{learn}^{\mathsf{Up}}_{\mathsf{t}+1}(a) = \mathsf{false}$*, then* $a \in \mathcal{K}^{\mathsf{Up}}_{\mathsf{t}+1}$*. Once no more updates of* Up *succeed, the analogous statement holds true for the lower bounds.*

9. *Assume that Assumption 11 holds. Then, there are at most* $2\overline{m} \cdot \frac{|Act|}{\overline{\varepsilon}}$ *visits to state-action pairs which are not* Up*-converged. Once the upper bounds are not updated any more, there are at most* $2\overline{m} \cdot \frac{|Act|}{\overline{\varepsilon}}$ *visits to state-action pairs which are not* Lo*-converged.*

Proof. Items 1 and 2 follow directly as in Lemmas 18 and 19. The only additional observation to make is that the algorithm never adds new actions and that the changes to the bounds outside of Line 12 never reset the progress of an action's bounds.

Item 3 can be proven completely analogous to Lemma 20, since this proof only relies on the Markov property of the successor sampling. We only need to adjust the definition of the Y_i slightly to incorporate the modifications of the algorithm. Let thus $s'_{k_i} \in S$ denote the states obtained by the successor oracle in Line 10. By Lemma 36 we have that $\mathcal{V}_\mathsf{e}(\mathsf{rep}_{\mathsf{e}_i}(s'_{k_i})) = \mathcal{V}_\mathsf{e}(s''_{k_i})$, and thus $Y_i = \mathcal{V}_\mathsf{e}(\mathsf{rep}_{\mathsf{e}_i}(s'_{k_i}))$ still are i.i.d.

For Item 4, we first show that all newly introduced updates of Up and Lo are correct. Using Assumption 9, we prove the two special cases. The algorithm sets $\mathsf{Up}_\mathsf{t}(a) \leftarrow 0$ if an EC (R, B) without outgoing transitions and no target state is identified. In this case, we clearly have that $\mathcal{V}_\mathsf{e}(a) = 0$ for all $s \in R$. Similarly, setting $\mathsf{Lo}_\mathsf{e}(a) \leftarrow 1$ when any state in the EC (R, B) is an accepting state is correct, since clearly $\mathcal{V}_\mathsf{e}(a) = 1$ for all $s \in R$, $a \in Av_\mathsf{e} \cap B$. Due to Lemma 36, copying the respective bounds to the representative state $s_{(R,B)}$ (which happens implicitly in Line 26) is correct, too. Now, we can follow the same reasoning as in Lemma 21.

Items 5 and 6 can be proven as in Lemma 22.

Item 7 is proven analogous to Item 3, following the proof of Lemma 23. Again, this statement only depends on the sampled successors. We define $X_i = \pi_{k_1}[\mathsf{Up}_{k_1}](\mathsf{rep}_{\mathsf{e}_1}(s''_{k_i}))$. Since we don't modify the underlying transition probabilities, from which s''_{k_i} is obtained, these X_i are i.i.d. again and we can apply the same reasoning. To conclude the proof as before, we need to employ Lemma 39. Note that since we only speak about the actual computed bounds Up and Lo, we do not need to employ Lemma 36.

Item 8 follows directly as in Lemma 24. Similarly, Item 9 follows as in Lemma 25, using Item 1 instead of Lemma 18. □

In the proof of correctness for the no-EC DQL algorithm, we applied Lemma 46 directly on the MDP to obtain bounds on the reachability of s_+ based on the values of Up and Lo in Lemma 26. Now, we cannot apply this lemma directly on either $\mathcal{M}$ or $\mathcal{M}_\mathsf{e}$ since both may contain ECs. Hence, we apply the lemma on an MDP derived from $\mathcal{M}_\mathsf{e}$ to obtain a similar result. Let us thus first define the set of all actions in 'non-final' ECs as

$$E_\mathsf{e} = \bigcup\nolimits_{\{(R,B)\in \mathrm{EC}(\mathcal{M}_\mathsf{e}) \mid R\cap(T_\mathsf{e}\cup \mathsf{Z}_\mathsf{e})=\emptyset\}} B.$$

Lemma 41. *Assume that Assumptions 9 and 10 hold and fix an episode* e*. Then, we have for every state* $s \in S_\mathsf{e}$

$$\mathsf{Up}_\mathsf{e}(s) - 3\overline{\varepsilon}\cdot|S|p_{\min}^{-|S|} - \mathsf{Pr}^{\pi_\mathsf{e}}_{\mathcal{M}_\mathsf{e},s}[\Diamond\overline{\mathcal{K}^{\mathsf{Up}}_\mathsf{e}}] - \mathsf{Pr}^{\pi_\mathsf{e}}_{\mathcal{M}_\mathsf{e},s}[\Diamond E_\mathsf{e}] \leq \mathsf{Pr}^{\pi_\mathsf{e}}_{\mathcal{M}_\mathsf{e},s}[\Diamond T_\mathsf{e}].$$

Proof. We first want to derive an MDP from $\mathcal{M}_\mathsf{e}$ without any ECs but still capturing its behaviour. For this, recall that there are two kinds of ECs in $\mathcal{M}_\mathsf{e}$. Firstly, there are ECs which correspond to ECs in the original $\mathcal{M}$. Secondly, we get a self-loop EC for each identified target- or zero-state, i.e. states in T_e or Z_e. We define the derived MDP $\mathcal{M}'_\mathsf{e} = (S_\mathsf{e}\cup\{s_+,s_-\}, Act_\mathsf{e}\cup\{a_+,a_-\}, \Delta'_\mathsf{e}, Av'_\mathsf{e})$, where

$$\begin{aligned}
\Delta'_\mathsf{e}(s_\circ,a_\circ) &= \{s_\circ \mapsto 1\} \quad \text{for } \circ\in\{+,-\}\\
\Delta'_\mathsf{e}(s,a) &= \{s_+ \mapsto 1\} \quad \text{for all } s\in T_\mathsf{e},\ a\in Av_\mathsf{e}(s),\\
\Delta'_\mathsf{e}(s,a) &= \{s_- \mapsto 1\} \quad \text{for all } s\in \mathsf{Z}_\mathsf{e},\ a\in Av_\mathsf{e}(s),\\
\Delta'_\mathsf{e}(s,a) &= \{s_+ \mapsto 1\} \quad \text{for all } a\in E,\ s=\mathsf{state}(a,\mathcal{M}_\mathsf{e}),\\
\Delta'_\mathsf{e}(s,a) &= \Delta_\mathsf{e}(s,a) \quad \text{for all other } s\in S_\mathsf{e},\ a\in Av_\mathsf{e}(s),
\end{aligned}$$

and $Av'_\mathsf{e}(s) = Av_\mathsf{e}(s)$ for $s\in S_\mathsf{e}$ and $Av'_\mathsf{e}(s_\circ) = \{a_\circ\}$ for $\circ\in\{+,-\}$. In essence, $\mathcal{M}'_\mathsf{e}$ equals $\mathcal{M}_\mathsf{e}$ except that we (i) added the special states s_+ and s_-, (ii) all states in T_e and Z_e move to s_+ and s_-, respectively, and (iii) all actions in ECs outside of T_e and Z_e move to s_+, in the spirit of Lemma 38.

Clearly, $\mathcal{M}'_\mathsf{e}$ has no ECs except the special states s_+ and s_- and thus satisfies Assumption 1. Moreover, the probability of reaching s_+ in $\mathcal{M}'_\mathsf{e}$ equals the probability of reaching $T_\mathsf{e}\cup E_\mathsf{e}$ in $\mathcal{M}_\mathsf{e}$ by construction of $\mathcal{M}'_\mathsf{e}$ **(I)**.

Now, we extend π_e to select action $a_\circ$ in the special state $s_\circ$ to obtain π'_e. Furthermore, we set $X(s,a) = \mathsf{Up}_\mathsf{e}(a)$ for all states $s\in S_\mathsf{e}$, $a\in Av_\mathsf{e}(s)$, $X(s_+,a_+)=1$, and $X(s_-,a_-)=0$.

We apply Lemma 46 with $\mathcal{M} = \mathcal{M}'_{\mathsf{e}}$, $\pi = \pi'_{\mathsf{e}}$, $\kappa_l = -1$, and $\kappa_u = 3\bar{\varepsilon}$. As a result, for each state $s \in S_{\mathsf{e}}$ we have

$$\pi'_{\mathsf{e}}[X](s) - \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}',s}[\Diamond\{s_+\}] \leq 3\bar{\varepsilon} \cdot |S| p_{\min}^{-|S|},$$

where $\mathcal{M}'$ is the MDP defined in the lemma. Observe that for $s \in S_{\mathsf{e}}$ **(II)**

$$\pi'_{\mathsf{e}}[X](s) = \sum\nolimits_{a \in Av'_{\mathsf{e}}(s)} \pi'_{\mathsf{e}}(s,a) \cdot X(s,a) = \sum\nolimits_{a \in Av_{\mathsf{e}}(s)} \pi_{\mathsf{e}}(s,a) \cdot \mathsf{Up}_{\mathsf{e}}(a) = \pi_{\mathsf{e}}[\mathsf{Up}_{\mathsf{e}}](s).$$

To analyse how $\mathcal{M}'$ and $\mathcal{M}'_{\mathsf{e}}$ are related, we first need to derive the structure of $\mathcal{K}$ from the lemma. Thus, we now prove that $\mathcal{K} = \mathcal{K}_{\mathsf{e}}^{\mathsf{Up}} \cup \{a_+, a_-\}$. Recall that $\mathcal{K} = \{a \in Act_{\mathsf{e}} \cup \{a_+, a_-\} \mid X(s,a) - \Delta'_{\mathsf{e}}(s,a)\langle \pi'_{\mathsf{e}}[X]\rangle \leq 3\bar{\varepsilon}\}$ and

$$\Delta'_{\mathsf{e}}(s,a)\langle \pi'_{\mathsf{e}}[X]\rangle = \sum\nolimits_{s' \in S_{\mathsf{e}} \cup \{s_+, s_-\}} \Delta'_{\mathsf{e}}(s,a,s') \cdot \sum\nolimits_{a' \in Av'_{\mathsf{e}}(s')} \pi(s',a') \cdot X(s',a').$$

Clearly, a_+ and a_- satisfy the requirements due to their self-loop. Furthermore, we have $\pi'_{\mathsf{e}}[X](s_+) = 1$, $\pi'_{\mathsf{e}}[X](s_-) = 0$ **(III)**. Now, let $a \in Act_{\mathsf{e}}$ and $s \in S_{\mathsf{e}}$ the corresponding state. By definition, we have $X(s,a) = \mathsf{Up}_{\mathsf{e}}(a)$, hence we need to show that $\Delta'_{\mathsf{e}}(s,a)\langle \pi'_{\mathsf{e}}[X]\rangle = \Delta_{\mathsf{e}}(s,a)\langle \pi_{\mathsf{e}}[\mathsf{Up}_{\mathsf{e}}]\rangle$. We proceed with a case distinction.

- $s \in T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}}$: By definition of the algorithm, we have $\mathsf{Up}_{\mathsf{e}}(s) = 1$ or 0, respectively. The unique successor under any action $a \in Av_{\mathsf{e}}(s)$ in $\mathcal{M}_{\mathsf{e}}$ equals s by definition, thus $\Delta_{\mathsf{e}}(s,a)\langle \pi_{\mathsf{e}}[\mathsf{Up}_{\mathsf{e}}]\rangle = \mathsf{Up}_{\mathsf{e}}(s)$. In $\mathcal{M}'_{\mathsf{e}}$, the unique successor equals s_+ or s_-, respectively. Thus, with **[III]**, we have $\pi'_{\mathsf{e}}[X](s) = \pi_{\mathsf{e}}[\mathsf{Up}_{\mathsf{e}}](s)$. The claim follows.
- $a \in E$: Note that this case implies that $s \notin T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}}$. Due to Lemma 38, we have that $\mathsf{Up}_{\mathsf{e}}(a) = 1$ for all such actions. Recall that π_{e} follows actions maximizing Up_{e}. Consequently, $\pi_{\mathsf{e}}[\mathsf{Up}_{\mathsf{e}}](s') = \pi'_{\mathsf{e}}[X](s') = \mathsf{Up}_{\mathsf{e}}(s') = 1$ for all states s' inside an non-trivial EC of $\mathcal{M}_{\mathsf{e}}$. Thus, we also have $\Delta_{\mathsf{e}}(s,a)\langle \pi_{\mathsf{e}}[\mathsf{Up}_{\mathsf{e}}]\rangle = 1$. From the definition of $\mathcal{M}'_{\mathsf{e}}$ and **[III]**, we directly get $\Delta'_{\mathsf{e}}(s,a)\langle \pi'_{\mathsf{e}}[X]\rangle = 1$.
- $s \notin T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}}$, $a \notin E$: By definition, we have $\Delta_{\mathsf{e}}(s,a) = \Delta'_{\mathsf{e}}(s,a)$. Together with **[II]** and **[III]**, the statement follows.

Recall that $\mathcal{M}'$ is defined as $\mathcal{M}'_{\mathsf{e}}$ except that $\Delta'(s,a) = \{s_+ \mapsto X(s,a), s_- \mapsto 1 - X(s,a)\}$ for all $a \notin \mathcal{K}$. Hence, as in Lemma 26, we get that for all states $s \in S_{\mathsf{e}}$

$$\mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}',s}[\Diamond\{s_+\}] - \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}] \leq \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\{s_+\}],$$

and thus with **[I]** we get **(IV)**

$$\pi'_{\mathsf{e}}[X](s) - 3\bar{\varepsilon} \cdot |S| p_{\min}^{-|S|} - \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}] \leq \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond(T_{\mathsf{e}} \cup E_{\mathsf{e}})].$$

Further, we have $\pi'_{\mathsf{e}}[X](s) = \pi_{\mathsf{e}}[\mathsf{Up}_{\mathsf{e}}](s) = \mathsf{Up}_{\mathsf{e}}(s)$ by Lemma 40, Item 5 **(V)**.

To conclude the proof, we show that $\mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}] \leq \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}]$ **(VI)**. To this end, observe that (i) for each state $s \in S_{\mathsf{e}}$ and action $a \in Av_{\mathsf{e}}(s)$ we either have $\Delta_{\mathsf{e}}(s,a) =$

$\Delta'_{\mathsf{e}}(s,a)$ or $\operatorname{supp}\Delta'_{\mathsf{e}}(s,a) \subseteq \{s_+, s_-\}$ and (ii) the added states s_+ and s_- are absorbing. Thus, each run reaching $\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}$ in $\mathcal{M}'_{\mathsf{e}}$ has a corresponding, equally probable path in $\mathcal{M}_{\mathsf{e}}$.

The overall claim follows by combining the above equations and applying a union bound.

$$\begin{aligned}
&\mathsf{Up}_{\mathsf{e}}(s) - 3\overline{\varepsilon}\cdot|S|p_{\min}^{-|S|} - \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}] \\
&\quad\overset{\text{[V]}}{=} \pi'_{\mathsf{e}}[X](s) - 3\overline{\varepsilon}\cdot|S|p_{\min}^{-|S|} - \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}] \\
&\quad\overset{\text{[VI]}}{\leq} \pi'_{\mathsf{e}}[X](s) - 3\overline{\varepsilon}\cdot|S|p_{\min}^{-|S|} - \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Up}}}] \\
&\quad\overset{\text{[IV]}}{\leq} \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond(T_{\mathsf{e}}\cup E_{\mathsf{e}})].
\end{aligned}$$

□

Lemma 42. *Assume that Assumptions 9 and 10 hold and fix an episode* e*. Then, we have for every state* $s \in S_{\mathsf{e}}$

$$\mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond T_{\mathsf{e}}] \leq \mathsf{Lo}_{\mathsf{e}}(s) + 3\overline{\varepsilon}\cdot|S|p_{\min}^{-|S|} + \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}_{\mathsf{e}}^{\mathsf{Lo}}}] + \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond E_{\mathsf{e}}].$$

Proof. As in Lemma 41, we construct a second MDP without ECs, but slightly modify the transition function. In particular, let $\mathcal{M}'_{\mathsf{e}} = (S_{\mathsf{e}}\cup\{s_+,s_-\}, Act_{\mathsf{e}}\cup\{a_+,a_-\}, \Delta'_{\mathsf{e}}, Av'_{\mathsf{e}})$ be defined as before. However, for $a \in E_{\mathsf{e}}$ and $s = \mathsf{state}(a,\mathcal{M}_{\mathsf{e}})$, we define

$$\Delta'_{\mathsf{e}}(s,a) = \{s_+ \mapsto \Delta_{\mathsf{e}}(s,a)\langle\pi_{\mathsf{e}}[\mathsf{Lo}_{\mathsf{e}}]\rangle, s_- \mapsto 1-\Delta_{\mathsf{e}}(s,a)\langle\pi_{\mathsf{e}}[\mathsf{Lo}_{\mathsf{e}}]\rangle\}.$$

Again, $\mathcal{M}'_{\mathsf{e}}$ has no ECs except in the two special states and thus Lemma 46 is applicable. We set $X(s,a) = \mathsf{Lo}_{\mathsf{e}}(a)$ for all states $s \in S_{\mathsf{e}}$, $X(s_+,a_+) = 1$, and $X(s_-,a_-) = 0$. As above, we have that $\pi'_{\mathsf{e}}[X](s) = \pi_{\mathsf{e}}[\mathsf{Lo}_{\mathsf{e}}](s)$ for all $s \in S_{\mathsf{e}}$. We apply the lemma with $\mathcal{M} = \mathcal{M}'_{\mathsf{e}}$, $\pi = \pi'_{\mathsf{e}}$, $\kappa_l = -3\overline{\varepsilon}$, and $\kappa_u = 1$. Thus, for each state $s \in S_{\mathsf{e}}$

$$\mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}',s}[\Diamond\{s_+\}] - \pi'_{\mathsf{e}}[X](s) \leq 3\overline{\varepsilon}\cdot|S|p_{\min}^{-|S|},$$

where $\mathcal{M}'$ is the MDP defined in the lemma. We again show that $\mathcal{K} = \mathcal{K}_{\mathsf{e}}^{\mathsf{Lo}}\cup\{a_+,a_-\}$ by case distinction.

- Trivially, $a_+, a_- \in \mathcal{K}$, $\pi'_{\mathsf{e}}[X](s_+) = 1$, and $\pi'_{\mathsf{e}}[X](s_-) = 0$.
- $s \in T_{\mathsf{e}}\cup\mathsf{Z}_{\mathsf{e}}$: The claims follow by an analogous argument. Recall that for these states we have $\mathsf{Up}_{\mathsf{e}}(a) = \mathsf{Lo}_{\mathsf{e}}(a)$ for all $a \in Av_{\mathsf{e}}(s)$.
- $a \in E$: Inserting the definitions, we get

$$\begin{aligned}
\Delta'_{\mathsf{e}}(s,a)\langle\pi'_{\mathsf{e}}[X]\rangle &= \Delta'_{\mathsf{e}}(s,a,s_+)\cdot\pi'_{\mathsf{e}}[X](s_+) + \Delta'_{\mathsf{e}}(s,a,s_-)\cdot\pi'_{\mathsf{e}}[X](s_-) \\
&= \Delta_{\mathsf{e}}(s,a)\langle\pi_{\mathsf{e}}[\mathsf{Lo}_{\mathsf{e}}]\rangle\cdot 1 + (1-\Delta_{\mathsf{e}}(s,a)\langle\pi_{\mathsf{e}}[\mathsf{Lo}_{\mathsf{e}}]\rangle)\cdot 0 \\
&= \Delta_{\mathsf{e}}(s,a)\langle\pi_{\mathsf{e}}[\mathsf{Lo}_{\mathsf{e}}]\rangle.
\end{aligned}$$

- $s \notin T_{\mathsf{e}}\cup\mathsf{Z}_{\mathsf{e}}$, $a \notin E$: Follows analogously.

As in Lemma 26, we also get for all states $s \in S_{\mathsf{e}}$ that

$$\mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\{s_+\}] \leq \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}',s}[\Diamond\{s_+\}] + \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{e}}}].$$

Similar to the above proof, we have $\pi'_{\mathsf{e}}[X](s) = \pi_{\mathsf{e}}[\mathsf{Lo}_{\mathsf{e}}](s) \leq \mathsf{Lo}_{\mathsf{e}}(s)$ by Lemma 40, Item 5. With completely analogous reasoning, we can show that $\mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{e}}}] \leq \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{e}}}]$. Putting all equations together, we get that

$$\mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\{s_+\}] \leq \mathsf{Lo}_{\mathsf{e}}(s) + 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} + \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{e}}}].$$

Now, it remains to show that $\mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond T_{\mathsf{e}}] - \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},s}[\Diamond E_{\mathsf{e}}] \leq \mathsf{Pr}^{\pi'_{\mathsf{e}}}_{\mathcal{M}'_{\mathsf{e}},s}[\Diamond\{s_+\}]$. This claim follows with the same reasoning as before, since we have that $\Delta_{\mathsf{e}}(s,a) = \Delta'_{\mathsf{e}}(s,a)$ for $a \notin E_{\mathsf{e}}, s = \mathsf{state}(a, \mathcal{M}_{\mathsf{e}})$. Thus, every path in $\mathcal{M}_{\mathsf{e}}$ which does not visit E has a corresponding, equally probable path in $\mathcal{M}'_{\mathsf{e}}$. The overall claim again follows by combining the above equations. □

Theorem 4. *Algorithm 5 terminates and yields a correct result with probability at least* $1-\delta$ *after at most* $\mathcal{O}(\mathrm{POLY}(|Act|, p_{\min}^{-|S|}, \varepsilon^{-1}, \ln\delta))$ *steps.*

Proof. This proof is largely analogous to the proof of Theorem 3, we shorten some of its parts. Again, we only consider executions where Assumptions 11, 10, and 9 hold. By Lemma 35 and Lemma 40, Items 7 and 3 together with the union bound, this happens with probability at least $1-\delta$. Correctness of the result upon termination follows from Lemma 40, Item 4.

We show by contradiction that the algorithm terminates for almost all considered executions. Thus, assume that the execution does not halt with non-zero probability. By Lemma 31, all of these executions experience an infinite number of episodes.

Due to Lemma 40, Item 2, there are only finitely many attempted updates on all considered executions and the algorithm eventually does not change Up, since no successful updates can occur from some step t onwards. Similarly, there are only finitely many EC collapses due to Lemma 30, and eventually the sampling MDP $\mathcal{M}_{\mathsf{e}}$ stabilizes. This means that all following samples are obtained by sampling according to the strategy π_{t} on the MDP $\mathcal{M}_{\mathsf{e}}$. Again, we employ Lemma 48 to continue the proof and we get $\mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Up}}_{\mathsf{t}}}] = 0$ and $\mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{t}}}] = 0$ on almost all considered executions. By an analogous argument, we can show that $\mathsf{Pr}^{\pi_{\mathsf{t}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond E_{\mathsf{e}}] = 0$, since otherwise by Lemma 27 (with $T = T_{\mathsf{e}} \cup \mathsf{Z}_{\mathsf{e}}$) we have a non-zero probability of detecting a new EC, contradicting our assumption.

Thus, by applying Lemma 41

$$\mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond T_{\mathsf{e}}] \geq \mathsf{Up}_{\mathsf{e}}(\hat{s}) - 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} - \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Up}}_{\mathsf{e}}}] - \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond E_{\mathsf{e}}] > \mathsf{Up}_{\mathsf{e}}(\hat{s}) - \frac{\varepsilon}{2}.$$

Dually, with Lemma 42 we get

$$\mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond T_{\mathsf{e}}] \leq \mathsf{Lo}_{\mathsf{e}}(\hat{s}) + 3\overline{\varepsilon} \cdot |S| p_{\min}^{-|S|} + \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond\overline{\mathcal{K}^{\mathsf{Lo}}_{\mathsf{e}}}] + \mathsf{Pr}^{\pi_{\mathsf{e}}}_{\mathcal{M}_{\mathsf{e}},\hat{s}}[\Diamond E_{\mathsf{e}}] < \mathsf{Lo}_{\mathsf{e}}(\hat{s}) + \frac{\varepsilon}{2}.$$

Together, $\mathsf{Up}_{\mathsf{e}}(\hat{s}) - \mathsf{Lo}_{\mathsf{e}}(\hat{s}) < \varepsilon$, contradicting the assumption.

For the step bound, we can mostly replicate the idea of the DQL variant without ECs. In particular, we can bound the number of paths by the same argument: The probability of reaching a non-Up- / non-Lo-converged action within $|S|$ steps is at least $p_{\min}^{|S|}$ (or 0). By Lemma 40, Item 9 we again get that the number of visits to such actions is bounded. Since $\mathsf{i} \geq |Act| \geq |S|$ and thus the sampling isn't stopped early due to that condition, we again can bound the maximal number of paths by the same n. For the length of the paths, observe that they are bounded by $2\mathsf{i}^3$ by construction of the algorithm. From the definition of i in Equation (6.1), we see that this bound is polynomial, too, by considering the Taylor expansion of the exponential. □

7 Experimental Evaluation

In this section, we provide a brief experimental evaluation of our BRTDP method. We rewrote the implementation of [Brá+14] from scratch. Instead of integrating with PRISM [KNP11], we only used it as a library for parsing the modelling language and instead use our own, tailored representations of MDP. Previously, several sampling heuristics were proposed. We implemented all of them and several more, however our experiments have shown that a weighted sampling heuristic (transition probability times upper bound of the successor) consistently performs among the best. Thus, we only report numbers for this single sampling heuristic. Our implementation is able to handle both bounded and unbounded reachability queries on MDP and Markov chains, as well as unbounded queries on continuous-time Markov chains via embedding. Additionally, the implementation supports simple 'until' properties, with support for full LTL [Pnu77] planned.

We also ran the current version of PRISM (v4.5) on the given models in its default configuration. In particular, it uses the 'hybrid' engine, profiting from symbolic computation. Moreover, PRISM uses standard value iteration without guarantees on the correctness of the result by default, saving a lot of computational resources. To obtain somewhat comparable values, we also ran PRISM with the `-explicit -intervaliteration` options.

Unfortunately, we could not obtain a working version of the original implementation of [Brá+14] (compilation of the provided source code failed). Hence, we copied the values reported in the original paper in Table 7.1. These results numbers were obtained on different hardware. However, since the execution times of PRISM are very similar, we conjecture that these numbers are comparable.

We omit evaluation of the DQL approach, since the associated constants are infeasible for practical application: Already for an MDP with 10 States, 20 actions and $p_{\min} = 0.1$, we obtain $\overline{m} \approx 10^{26}$ for $\varepsilon = 0.1$ and $\delta = 0.01$. A practically more feasible approach based on the ideas of this DQL algorithm can be found in [AKW19].

All experiments were run on a Ryzen 5 3600 Processor (6x3.60 GHz) and 16 GB RAM, using OpenJDK version `11.0.8` on a Ubuntu 18.04 VM inside WSL2. Each experiment was restricted to a single core (using `taskset`) and 2 GB RAM (using the `-Xmx2G` switch of the JVM) with a timeout of 10 minutes (using `timeout`). We always report the wall-clock time, i.e. the total elapsed time from start to finish of the overall process, including JVM startup etc.

7.1 Results

We compare our new implementation to the original one of [Brá+14], replicating their Table 1. In this comparison, four models with several parameter values are used. The first three, **zeroconf**, **wlan**, and **firewire** (implementation with deadline), are taken from the PRISM benchmark suite [KNP12]. The fourth, **mer**, is taken from [FKP11]. As in

Table 7.1: Comparison of the BRTDP algorithm to both PRISM's default reachability computation and the previous implementation of [Brá+14], replicating the structure of their Table 1. For each of the four models, we first give the values of all parameters, the size of the complete model as reported by PRISM, the average number of states explored by our method and then the average times required to obtain a result up to the specified precision (10^{-6} except for **zeroconf**, where 10^{-8} was chosen in [Brá+14]) for each of the implementations. For PRISM, we list the execution times for the 'hybrid' engine. We omit values for the explicit approach since it ran out of memory on every instance. For [Brá+14], we include the values reported in the original paper, taking the best solution time out of all the heuristics presented there.

Name [params]	Values	States		Time (s) PRISM	BRTDP	[Brá+14]
zeroconf [N, K]	20, 10	$3.0 \cdot 10^6$	529	124	< 1	1.5
	20, 14	$4.4 \cdot 10^6$	666	214	< 1	2.1
	20, 18	$5.5 \cdot 10^6$	814	294	< 1	3.7
wlan [BOFF]	4	$3.5 \cdot 10^5$	537	11	< 1	< 1
	5	$1.3 \cdot 10^6$	532	33	< 1	< 1
	6	$5.0 \cdot 10^6$	500	127	< 1	< 1
firewire [delay, dl]	36, 200	$6.7 \cdot 10^6$	17743	53	1.9	2.3
	36, 240	$1.3 \cdot 10^7$	31272	106	3.1	6.7
	36, 280	$1.9 \cdot 10^7$	48673	165	4.7	7.4
mer [N, q]	3000, 0.0001	$1.8 \cdot 10^7$	2589	145	1.0	2.4
	3000, 0.9999		4154	144	1.1	2.8
	4500, 0.0001	$2.7 \cdot 10^7$	2588	225	1.0	2.4
	4500, 0.9999		4156	225	1.1	2.8

[Brá+14], we use a precision requirement of $\varepsilon = 10^{-6}$ except for **zeroconf**, where we set $\varepsilon = 10^{-8}$. The given values for our BRTDP approach are averaged over 5 runs to account for the involved randomization. The results of Table 7.1 clearly show that on these particular models the BRTDP method vastly outperforms the standard approach of PRISM. Additionally, the explicit / interval iteration method of PRISM runs out of memory while constructing the model on every single instance where our methods converge within a few seconds. Moreover, we see that our new implementation is significantly faster than the previous one.

Furthermore, we extended the evaluation to all applicable models of the PRISM benchmark suite. The results are summarized in Table 7.2. Overall, the table shows that a significant portion of models is well suited for our analysis. However, as expected, our approach does not consistently outperform PRISM. This most likely is due to a particular structure of the models, for example, comprising a single end component. This makes them less suitable for the guided path sampling approach, since we then have to repeatedly discover and collapse end components instead of simply identifying and collapsing the whole component once. Recall that this can be overcome by adapting the SamplePairs and UpdateECs methods accordingly. Understanding the exact reasons why sometimes classical interval iteration performs better and designing heuristics to identify these cases

Table 7.2: Evaluation of our BRTDP algorithm on several models from the PRISM benchmark suite [KNP12]. For each model, we report both the total number of instances as well as the average size of all those instances where both PRISM and our method succeeded. Then, for each property associated with those models we list both the average ratio of explored states and time until convergence of our method compared to PRISM (both for the 'hybrid' and 'explicit' engine). For readability, we group all models where BRTDP performed faster in the upper part of the table.

Family	#	Avg. Size	Property	States	Time	
brp	12	2,302	`p1`	0.91	0.50	0.45
			`p2`	0.98	0.50	0.45
			`p4`	0.98	0.51	0.46
egl	16	95,230	`unfairA`	0.72	1.05	0.86
			`unfairB`	0.75	1.09	0.84
wlan	7	969,161	`collisions`	0.00	0.38	0.24
zeroconf	16	383,919	`correct_max`	0.12	0.23	0.22
			`correct_min`	0.10	0.23	0.24
consensus	6	7,819	`c2`	0.88	2.09	1.56
			`disagree`	0.98	6.27	4.36
crowds	16	411,042	`positive`	0.95	5.41	1.89
csma	9	389,136	`some_before`	0.89	5.01	2.02
firewire (dl)	8	217,286	`deadline`	0.74	11.56	5.88
firewire (impl,dl)	8	2,048,813	`deadline`	0.51	3.79	2.96
nand	10	2,491,977	`reliable`	0.50	2.86	1.86
wlan (dl)	7	4,507,799	`deadline`	0.55	16.52	6.07

could significantly improve performance.

8 Conclusion and Future Work

In this thesis, we improved and extended the ideas of [Brá+14], fixing several imprecisions and issues of the proofs. This results in a framework for verifying MDP, using learning algorithms. Building upon exiting methods, we thus provide novel techniques to analyse infinite-horizon reachability properties of arbitrary MDPs, yielding either exact bounds in the white-box scenario or probabilistically correct bounds in the black-box scenario. Moreover, we presented a generalization of the methods of [Brá+14], allowing for further, more sophisticated applications.

Given this framework, an interesting direction for future work would be to extend this approach with more sophisticated learning algorithms. Another, orthogonal direction is to explore whether our approach can be combined with symbolic methods. Finally, we plan to evaluate our algorithms on more real-world models.

A Auxiliary Statements

In this chapter we provide some general statements about Markov chains and decision processes which are used in various proofs for the DQL algorithms.

From Reachability to Step-bounded Reachability

In this section we prove several statements relating the infinite-horizon reachability with the reachability after a sufficiently large number of steps.

Lemma 43. *For any Markov chain* $\mathsf{M} = (S, \delta)$*, state* s*, and target set* T*, we have that either* $\Pr_{\mathsf{M},s}[\Diamond T] = 0$ *or* $\Pr_{\mathsf{M},s}[\Diamond^{\leq |S|} T] \geq \delta_{\min}^{|S|}$*, where* $\delta_{\min}$ *is the minimal transition probability, i.e.* $\delta_{\min} = \min\{\delta(s, s') \mid s \in S, s' \in \operatorname{supp} \delta(s)\}$*.*

Proof. Fix the Markov chain M, state s, and target set T as in the lemma. In the first case there is nothing to prove, thus assume that $\Pr_{\mathsf{M},s}[\Diamond T] > 0$. This means that there exists a finite path ϱ from s to some state in T. By the pigeon-hole principle, we can assume this path has length at most $|S|$. Clearly, the probability of any single transition on this path is at least $\delta_{\min}$ and thus the overall probability of this path is at least $\delta_{\min}^{|S|}$. □

Corollary 2. *For any MDP* $\mathcal{M} = (S, Act, Av, \Delta)$*, memoryless strategy* $\pi \in \Pi_{\mathcal{M}}^{\mathsf{MD}}$*, state* s*, and target set* T*, we have that either* $\Pr_{\mathcal{M},s}^{\pi}[\Diamond T] = 0$ *or* $\Pr_{\mathcal{M},s}^{\pi}[\Diamond^{\leq |S|} T] \geq \delta_{\min}(\pi)^{|S|}$*, where* $\delta_{\min}(\pi) = \min\{\pi(s, a) \cdot \Delta(s, a, s') \mid s \in S, a \in Av(s), \pi(s, a) > 0, s' \in \operatorname{supp} \Delta(s, a, s')\}$*.*

Proof. Follows directly from the above lemma by applying it to $\mathcal{M}^{\pi}$. □

The following lemma shows that by considering a large enough horizon, the step-bounded and unbounded reachability values coincide up to a small error, similar in spirit to [KS02, Lemma 2].

Lemma 44. *Given a Markov chain* $\mathsf{M} = (S, \delta)$*, a state* $s \in S$*, a constant* $\tau \in (0, 1]$*, and a target set* T*, for* $N \geq \ln(\frac{2}{\tau}) \cdot |S| \delta_{\min}^{-|S|}$ *we have*

$$\Pr_{\mathsf{M},s}[\Diamond T] - \Pr_{\mathsf{M},s}[\Diamond^{\leq N} T] \leq \tau.$$

Proof. We can express $\Pr_{\mathsf{M},s}[\Diamond T]$ as a sum of $\Pr_{\mathsf{M},s}[\Diamond^{\leq N} T]$ and $\Pr_{\mathsf{M},s}[\Diamond^{> N} T]$, where $\Diamond^{> N} T = \Diamond T \setminus \Diamond^{\leq N} T$ are all paths which reach the set T but only after at least $N + 1$ steps. Clearly,

$$\Pr_{\mathsf{M},s}[\Diamond T] - \Pr_{\mathsf{M},s}[\Diamond^{\leq N} T] = \Pr_{\mathsf{M},s}[\Diamond^{> N} T].$$

By [BKK14, Lemma 5.1] we have that $\Pr_{\mathsf{M},s}[\Diamond^{> N} T] \leq 2 \cdot c^N$, where $c = \exp(-|S|^{-1} \delta_{\min}^{|S|})$.

$$\begin{aligned} 2 \cdot c^N \leq \tau \quad &\Leftrightarrow \quad N \cdot \ln c \geq \ln \frac{\tau}{2} \quad \Leftrightarrow \quad N \geq \ln \frac{\tau}{2} \cdot (\ln c)^{-1} \\ &\Leftrightarrow \quad N \geq \ln \frac{\tau}{2} \cdot -|S| \delta_{\min}^{-|S|} \quad \Leftrightarrow \quad N \geq \ln \frac{2}{\tau} \cdot |S| \delta_{\min}^{-|S|} \end{aligned}$$ □

Unique Solution of Bellman Equations

Now, we prove that a particular class of Bellman equations has a unique solution by proving that the associated functor is a contraction.

Lemma 45. *Let $\mathcal{M}$ be an MDP, $Av_? : S \to Act$ a function mapping a state s to a subset of its available actions $Av_?(s) \subseteq Av(s)$, $c : S \to \mathbb{R}$ a cost function, and π a memoryless strategy on $\mathcal{M}$. Define $S_= = \{s \mid Av_?(s) = \emptyset\}$.*

If $\mathsf{Pr}^\pi_{\mathcal{M},s}[\Diamond S_=] > 0$ for all states $s \in S$, then the system of Bellman equations

$$f(s) = c(s) + \sum\nolimits_{a \in Av_?(s)} \pi(s,a) \cdot \Delta(s,a)\langle f \rangle$$

has a unique solution f.

Proof. Define the iteration operator F as

$$F(f)(s) = c(s) + \sum\nolimits_{a \in Av_?(s)} \pi(s,a) \cdot \Delta(s,a)\langle f \rangle.$$

Trivially, a function $f : S \to \mathbb{R}$ is a solution to the equation system if and only if it is a fixed point of F, i.e. $F(f)(s) = f(s)$ for all states $s \in S$.

We show that $F^{|S|}$, i.e. F applied $|S|$ times, is a contraction and thus has a unique fixed point, obtainable by iterating F. This means that there exists a contraction factor $0 \leq \gamma < 1$ such that for two arbitrary $f, g : S \to \mathbb{R}$, we have

$$\max_{s \in S} \left| F^{|S|}(f)(s) - F^{|S|}(g)(s) \right| \leq \gamma \cdot \max_{s \in S} |f(s) - g(s)|. \tag{A.1}$$

Let $P(s, s', k)$ be the probability of reaching state s' starting from s in exactly k steps using the strategy π by using only actions from $Av_?$. Note that for $s \in S_=$ this implies $P(s, s', k) = 0$ for any $s' \in S$ and any number k. For $s \in S_? := S \setminus S_=$, we have that

$$F^{|S|}(f)(s) = \sum\nolimits_{s' \in S} \left(\sum\nolimits_{i=0}^{|S|-1} P(s, s', i) \cdot c(s') \right) + \sum\nolimits_{s' \in S_?} P(s, s', |S|) \cdot f(s')$$

Observe that the first term is independent of f, hence for $s \in S_?$ we have

$$\begin{aligned}
&\left| F^{|S|}(f)(s) - F^{|S|}(g)(s) \right| \\
&\quad = \left| \sum\nolimits_{s' \in S_?} P(s, s', |S|) \cdot f(s') - \sum\nolimits_{s' \in S_?} P(s, s', |S|) \cdot g(s') \right| \\
&\quad \leq \sum\nolimits_{s' \in S_?} P(s, s', |S|) \cdot |f(s') - g(s')| \\
&\quad \leq \left(\sum\nolimits_{s' \in S_?} P(s, s', |S|) \right) \cdot \max_{s' \in S} |f(s') - g(s')|.
\end{aligned}$$

By assumption, we have that $\mathsf{Pr}^\pi_{\mathcal{M},s}[\Diamond S_=] > 0$. This implies that $\mathsf{Pr}^\pi_{\mathcal{M},s}[\Diamond^{\leq |S|} S_=] \geq \delta_{\min}(\pi) > 0$ by Corollary 2. For $s \in S_=$, observe that $F^{|S|}(f)(s) = f(s) = c(s)$ and hence $\left| F^{|S|}(f)(s) - F^{|S|}(g)(s) \right| = |f(s) - g(s)| = |c(s) - c(s)| = 0$. Consequently, $\gamma = \max_{s \in S_?} \sum_{s' \in S_?} P(s, s', |S|) \leq \delta_{\min}(\pi) < 1$ satisfies Inequality A.1 and we have that $F^{|S|}$ is a contraction. By the Banach fixed point theorem we get that $F^{|S|}$ has a unique fixed point and thus the equation system has a unique solution. □

Local Error Bounds to Global

The next lemma intuitively bounds the overall error of an approximation in an MDP given that the approximation is 'close' locally. Recall that, by definition

$$\Delta(s,a)\langle\pi[X]\rangle = \sum\nolimits_{s'\in S}\Delta(s,a,s')\cdot\sum\nolimits_{a'\in Av(s')}\pi(s',a')\cdot f(s',a').$$

Thus, the term $X(s,a) - \Delta(s,a)\langle\pi[X]\rangle$ in the lemma essentially denotes the difference between the state-action value $X(s,a)$ and the expected value obtained from X in the successors of (s,a) following π. Consequently, $\mathcal{K}$ contains those state-action pairs for which the value under X is consistent with the value of its successors up to some error.

Lemma 46. *Let $\mathcal{M} = (S, Act, Av, \Delta)$ be an MDP satisfying Assumption 1, $X : S \times Av \to [0,1]$ a function assigning a value between 0 and 1 to each state-action pair, π a memoryless strategy on $\mathcal{M}$, and $\kappa_l \leq \kappa_u$ two error bounds. Set*

$$\mathcal{K} := \{(s,a) \mid \kappa_l \leq X(s,a) - \Delta(s,a)\langle\pi[X]\rangle \leq \kappa_u\}.$$

Define a new MDP $\mathcal{M}' = (S, Act, Av, \Delta')$ where

$$\Delta'(s,a) = \begin{cases} \Delta(s,a) & \text{if } (s,a) \in \mathcal{K}, \text{ and} \\ \{s_+ \mapsto X(s,a), s_- \mapsto 1 - X(s,a)\} & \text{otherwise.} \end{cases}$$

Then, for each state $s \in S$ we have

$$\kappa_l \leq \frac{\delta_{\min}(\pi)^{|S|}}{|S|}\left(\pi[X](s) - \mathsf{Pr}^{\pi}_{\mathcal{M}',s}[\Diamond\{s_+\}]\right) \leq \kappa_u,$$

where $\delta_{\min}(\pi) = \min\{\pi(s,a)\cdot\Delta(s,a,s') \mid s \in S, a \in Av(s), \pi(s,a) > 0, s' \in \mathrm{supp}(\Delta(s,a))\}$ is the smallest transition probability in the Markov chain $\mathcal{M}^{\pi}$.

Proof. Define $v'(s) = \mathsf{Pr}^{\pi}_{\mathcal{M}',s}[\Diamond\{s_+\}]$. Furthermore, let $\mathcal{K}(s) = \{a \in Av(s) \mid (s,a) \in \mathcal{K}\}$ and $\neg\mathcal{K}(s) = \overline{\mathcal{K}(s)} \cap Av(s)$ the sets of all actions $a \in Av(s)$ such that $(s,a) \in \mathcal{K}$ and $(s,a) \notin \mathcal{K}$, respectively. Observe that v' is a solution to the following system of equations:

$$\begin{aligned} v'(s_+) &= 1 \\ v'(s_-) &= 0 \\ v'(s) &= \sum\nolimits_{a\in\mathcal{K}(s)}\pi(s,a)\cdot\Delta(s,a)\langle v'\rangle + \sum\nolimits_{a\in\neg\mathcal{K}(s)}\pi(s,a)\cdot X(s,a) \end{aligned}$$

We apply Lemma 45 to show that v' is the unique solution. Let $\varepsilon(s_+) = 1$, $\varepsilon(s_-) = 0$, and $\varepsilon(s) = \sum_{a\in\neg\mathcal{K}(s)}\pi(s,a)\cdot X(s,a)$ for all other $s \in S$. Further, set $Av_?(s_+) = Av_?(s_-) = \emptyset$ and $Av_?(s) = \mathcal{K}(s)$ for all other $s \in S$. Then, $\{s_+, s_-\} \subseteq S_=$. The MDP $\mathcal{M}'$ also satisfies Assumption 1, since no new ECs are introduced, and thus $\mathsf{Pr}^{\pi}_{\mathcal{M},s}[\Diamond S_=] = 1 > 0$ for all $s \in S$ by Lemma 2. Consequently, Lemma 45 is applicable and v' is the unique solution of the above equations.

$\pi[X]$ satisfies a similar set of equations:

$$\begin{aligned}
\pi[X](s_+) &= 1 \\
\pi[X](s_-) &= 0 \\
\pi[X](s) &= \sum\nolimits_{a \in Av(s)} \pi(s,a) \cdot X(s,a) \\
&= \sum\nolimits_{a \in \mathcal{K}(s)} \pi(s,a) \cdot X(s,a) + \sum\nolimits_{a \in \neg\mathcal{K}(s)} \pi(s,a) \cdot X(s,a) \\
&= \kappa(s) + \sum\nolimits_{a \in \mathcal{K}(s)} \pi(s,a) \cdot \Delta(s,a)\langle \pi[X] \rangle + \sum\nolimits_{a \in \neg\mathcal{K}(s)} \pi(s,a) \cdot X(s,a)
\end{aligned}$$

where $\kappa(s) = \sum_{a \in \mathcal{K}(s)} \pi(s,a) \cdot (X(s,a) - \Delta(s,a)\langle \pi[X] \rangle)$ is bounded by $\kappa_l \leq \kappa(s) \leq \kappa_u$. Again, by Lemma 45, these equations then have a unique fixed point, setting $\varepsilon(s) = \kappa(s) + \sum_{a \in \neg\mathcal{K}(s)} \pi(s,a) \cdot X(s,a)$.

Now, we prove a bound for the difference between X and v' using the above characterizations. Observe that the above equation systems only differ structurally by the error term $\kappa(s)$. Let thus $f(s) = \pi[X](s) - v'(s)$. This f is a fixed point of the following equation system:

$$\begin{aligned}
f(s_+) = f(s_-) &= 0 \\
f(s) &= \kappa(s) + \sum\nolimits_{a \in \mathcal{K}(s)} \pi(s,a) \cdot \Delta(s,a)\langle f \rangle
\end{aligned}$$

Clearly, f again is unique by Lemma 45.

Given a state s, the probability to reach the terminal states s_+ and s_- in $|S|$ steps following strategy π is bounded from below by $\delta_{\min}(\pi)^{|S|}$ due to Corollary 2. Consequently, the probability of not reaching these states in $|S|$ steps is bounded from above by $1 - \delta_{\min}(\pi)^{|S|} < 1$. Hence, we can bound the difference between $\pi[X]$ and v' by

$$\kappa(s) \cdot \sum\nolimits_{n=0}^{\infty} |S| \left(1 - \delta_{\min}(\pi)^{|S|}\right)^n = \kappa(s) \cdot |S| \delta_{\min}(\pi)^{-|S|}. \qquad \square$$

Bounding Reachability on Similar MDP

In this lemma, we show that MDP which are sufficiently 'similar' also have similar reachability values.

Lemma 47. *Let $\mathcal{M} = (S, Act, Av, \Delta)$ be an MDP, $T \subseteq S$ a set of target states, $\mathcal{K} \subseteq S \times Av$ a set of state-action pairs, and $\mathcal{M}' = (S', Act', Av', \Delta')$ an arbitrary MDP with $\mathcal{K} \subseteq S' \times Av'$ that coincides with $\mathcal{M}$ on $\mathcal{K}$ and T, i.e. (i) $Av(s) = Av'(s)$ for all $s \in \mathcal{K}$, (ii) $\Delta(s,a) = \Delta'(s,a)$ for all $(s,a) \in \mathcal{K}$, and (iii) $T \subseteq S'$. Moreover, let π be a strategy in $\mathcal{M}$, $s \in S \cap S'$ an arbitrary state in both MDP, and $N \in \mathbb{N}$ a natural number. Then,*

$$\mathsf{Pr}^{\pi}_{\mathcal{M},s}[\Diamond^{\leq N} T] \geq \mathsf{Pr}^{\pi'}_{\mathcal{M}',s}[\Diamond^{\leq N} T] - \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\Diamond^{\leq N} \overline{\mathcal{K}}],$$

where π' is an arbitrary strategy equal to π on all finite paths over $\mathcal{K}$, i.e. $\pi(\varrho) = \pi'(\varrho)$ for all $\varrho \in \mathcal{K}^\star \times S \cap \mathsf{FPaths}_{\mathcal{M}}$.

Proof. For a finite path $\varrho = s_1 a_1 \dots a_{n-1} s_n \in \mathsf{FPaths}_{\mathcal{M}}$, let $\mathsf{Pr}^{\pi}_{\mathcal{M},s}[\varrho]$ denote the probability of path ϱ occurring when following strategy π from state s. Let $\mathcal{K}_N$ denote the (finite)

set of all finite paths ϱ of length N starting in s such that all state-action pairs (s_i, a_i) in ϱ are in $\mathcal{K}$. Similarly, let $\neg\mathcal{K}_N$ denote the set of all such paths containing at least one state-action pair not in $\mathcal{K}$. Let $\mathcal{R}(\varrho)$ be a function which returns 1 if some target state of T is in path ϱ and 0 otherwise. Then, we have the following:

$$\mathsf{Pr}^{\pi'}_{\mathcal{M}',s}[\lozenge^{\leq N} T] - \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\lozenge^{\leq N} T] \tag{A.2}$$

$$= \sum_{\varrho \in \mathcal{K}_N} \Big(\mathsf{Pr}^{\pi'}_{\mathcal{M}',s}[\varrho] \cdot \mathcal{R}(\varrho) - \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\varrho] \cdot \mathcal{R}(\varrho)\Big) + \sum_{\varrho \in \neg\mathcal{K}_N} \Big(\mathsf{Pr}^{\pi'}_{\mathcal{M}',s}[\varrho] \cdot \mathcal{R}(\varrho) - \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\varrho] \cdot \mathcal{R}(\varrho)\Big) \tag{A.3}$$

$$= \sum_{\varrho \in \neg\mathcal{K}_N} \Big(\mathsf{Pr}^{\pi'}_{\mathcal{M}',s}[\varrho] \cdot \mathcal{R}(\varrho) - \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\varrho] \cdot \mathcal{R}(\varrho)\Big) \tag{A.4}$$

$$\leq \sum_{\varrho \in \neg\mathcal{K}_N} \mathsf{Pr}^{\pi'}_{\mathcal{M}',s}[\varrho] \cdot \mathcal{R}(\varrho) \tag{A.5}$$

$$\leq \sum_{\varrho \in \neg\mathcal{K}_N} \mathsf{Pr}^{\pi'}_{\mathcal{M}',s}[\varrho] \tag{A.6}$$

$$= \mathsf{Pr}^{\pi}_{\mathcal{M},s}[\lozenge^{\leq N} \overline{\mathcal{K}}] \tag{A.7}$$

In Equation A.3, we simply split the set of all paths of length N into $\mathcal{K}_N$ and $\neg\mathcal{K}_N$. For Equations A.4 and A.7, note that $\mathsf{Pr}^{\pi'}_{\mathcal{M}',s}$ and $\mathsf{Pr}^{\pi}_{\mathcal{M},s}$ agree on $\mathcal{K}_N$ by choice of $\mathcal{M}'$ and π'. □

Repeating Events in Markov Processes

Finally, we prove a general statement of Markov processes. The statement itself seems to be quite obvious, yet surprisingly tricky to prove. In essence, we want to show the following. Suppose that we are given a Markov process X_t on some probability space Ω together with a sequence of events A_t. Moreover, assume that for a significant set of atoms $\omega \in \Omega$ there is an infinite set of times T such that the *conditional* probability of A_t occurring is at least $\varepsilon > 0$, i.e. $\mathbb{P}[X_t \in A_t \mid X_{t-1}(\omega)] > \varepsilon$. Then, the set of atoms for which infinitely many A_t actually occur is also significant. The subtle difficulty of this statement arises from the fact that (i) conditional probabilities are considered, and (ii) the set T depends on the particular atom ω.

Lemma 48. *Fix some probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and a measure space $(S, \mathcal{S})$. Let $X_t : \Omega \to S$ be a Markov process on Ω and $A_t \in \mathcal{S}$ measurable events in S. Assume that the set $\Omega' = \{\omega \in \Omega \mid \exists T.\ |T| = \infty \wedge \forall t \in T.\ \mathbb{P}[X_t \in A_t \mid X_{t-1}](\omega) > \varepsilon\}$ has positive measure, i.e. $\mathbb{P}[\Omega'] > 0$, and that $\Omega'_t = \{\omega \in \Omega \mid \mathbb{P}[X_t \in A_t \mid X_{t-1}](\omega) > \varepsilon\}$ is measurable for all $t \in \mathbb{N}$. Then, $\mathbb{P}[\{\omega \in \Omega \mid \exists T.\ |T| = \infty \wedge \forall t \in T.\ X_{\mathsf{t}}(\omega) \in A_t\}] = \mathbb{P}[\Omega']$.*

Proof. Let $\omega \in \Omega'$. By assumption, for each such ω, there exists an infinite set of time-points $\mathsf{Tries}(\omega) = \{t_1, t_2, \cdots\}$ with $1 \leq t_1 < t_2 < \cdots$ where $\mathbb{P}[X_t \in A_t \mid X_{t-1}](\omega) > \varepsilon$. We call such an event a *try* of ω. Denote $\mathsf{Try}_i(\omega) = t_i$ or ∞ if no such t_i exists, e.g. for $\omega \notin \Omega'$. Informally, Try_i is the time of the i-th try of some outcome ω. Try_i is measurable by assumption, since its pre-images can be constructed using Ω'_t. Moreover,

let $\mathsf{Succs}(\omega) = \{s_1, s_2, \cdots\} \subseteq \mathsf{Tries}(\omega)$ be the times where $X_{s_j}(\omega) \in A_{s_j}$, called *$j$-th success(ful try)*. Note that $\mathsf{Succs}(\omega)$ possibly is finite or even empty for some outcomes ω, even for $\omega \in \Omega'$, since infinitely many tries may fail. Now, let $\mathsf{Succ}_j(\omega) = s_j \in \mathsf{Succs}(\omega)$ the time of the j-th success or ∞ if no such s_j exists, i.e. $j > |\mathsf{Succs}(\omega)|$. Succ_j is measurable since Try_i, X_t and A_t are measurable. To succinctly capture corner-cases, we further define $\mathsf{Succ}_0 = 0$. The successes $\mathsf{Succs}(\omega)$ naturally partition the set $\mathsf{Tries}(\omega)$ into $\mathsf{TriesJ}_j(\omega) = \{t \in \mathsf{Tries}(\omega) \mid \mathsf{Succ}_j(\omega) < t \leq \mathsf{Succ}_{j+1}(\omega)\}$. We use $\mathsf{TryJ}_{i,j}(\omega)$ to refer to the i-th element of $\mathsf{TriesJ}_j(\omega)$, or ∞ if no such element exists. $\mathsf{TryJ}_{i,j}$ is measurable due to Succ_j being measurable. Informally, $\mathsf{TryJ}_{i,j}(\omega)$ denotes the time of the i-th try since the j-th success.

We show that after a sufficient number of tries, there is a success with high probability. Repeating this argument inductively, we then show that there are infinitely many successes for almost all outcomes ω in Ω'.

Let thus $\mathsf{TryAtTJ}^t_{i,j}$ denote the set of runs which at time t have succeeded j times before and since the j-th success experienced i-th tries, where this i-th try happens exactly at time t. Formally,

$$\mathsf{TryAtTJ}^t_{i,j} := \{\omega \in \Omega' \mid \mathsf{TryJ}_{i,j}(\omega) = t\}.$$

Note that this definition implicitly includes the condition $\mathsf{Succ}_j(\omega) \leq t < \mathsf{Succ}_{j+1}(\omega)$ by definition of $\mathsf{TryJ}_{i,j}$. Thus, $\mathsf{TryAtTJ}^t_{i,j}$ are disjoint for fixed i and j.

We furthermore define $\mathsf{TriesJ}_{i,j} = \bigcup_{t=1}^{\infty} \mathsf{TryAtTJ}^t_{i,j} = \{\omega \in \Omega' \mid \mathsf{TryJ}_{i,j}(\omega) < \infty\}$ as the set of outcomes which after their j-th success experienced at least $i - 1$[1] unsuccessful tries. We have $\mathsf{TriesJ}_{i,j} = \mathsf{TriesJ}_{i+1,j} \cup \mathsf{TriesJ}_{1,j+1}$, since the i-th try either fails and the $i+1$-th try is experienced later (since $\mathsf{TriesJ}_{i,j} \subseteq \Omega'$, implying infinitely many tries) or the try succeeds. Observe that $\mathsf{TriesJ}_{i+1,j}$ and $\mathsf{TriesJ}_{1,j+1}$ are not disjoint, since, for example, the runs succeeding at the $i+1$-th try also are an element of $\mathsf{TriesJ}_{1,j+1}$. On the contrary, we show that $\mathbb{P}[\mathsf{TriesJ}_{i,j} \setminus \mathsf{TriesJ}_{1,j+1}] = 0$, i.e. almost all runs in $\mathsf{TriesJ}_{i,j}$ will eventually succeed again.

To this end, we show that for any fixed j we have that $\lim_{i\to\infty} \mathbb{P}[\mathsf{TriesJ}_{i,j}] = 0$. Fix some j and i with $\mathbb{P}[\mathsf{TriesJ}_{i,j}] > 0$ (otherwise there is nothing to prove, since $\mathsf{TriesJ}_{i,j}$ is monotonically decreasing in i). Let $\mathsf{TryTimesJ}_{i,j} = \{t \mid \mathbb{P}[\mathsf{TryAtTJ}^t_{i,j}] > 0\}$ which is non-empty by the previous condition. Clearly, $\mathbb{P}[\mathsf{TriesJ}_{i,j}] = \sum_{t=1}^{\infty} \mathbb{P}[\mathsf{TryAtTJ}^t_{i,j}] = \sum_{t\in\mathsf{TryTimesJ}_{i,j}} \mathbb{P}[\mathsf{TryAtTJ}^t_{i,j}]$, as $\mathsf{TryAtTJ}^t_{i,j}$ are disjoint. Observe that $\mathsf{TryAtTJ}^t_{i,j}$ is the intersection of several conditions on $X_{t'}$ for $t' < t$ and requiring that $\mathbb{P}[X_t \in A_t \mid X_{t-1}] > \varepsilon$. Hence, by the Markov property we have

$$\mathbb{P}[X_t \notin A_t \mid \mathsf{TryAtTJ}^t_{i,j}] = 1 - \mathbb{P}[X_t \in A_t \mid \mathsf{TryAtTJ}^t_{i,j}] = 1 - \mathbb{P}[X_t \in A_t \mid X_{t-1}] < 1 - \varepsilon.$$

Intuitively, this simply means that the probability of a try at time t succeeding does not depend on the number of previous tries and successes. Thus, for all $t \in \mathsf{TryTimesJ}_{i,j}$, we have $\mathbb{P}[X_t \notin A_t \cap \mathsf{TryAtTJ}^t_{i,j}] < (1-\varepsilon) \cdot \mathbb{P}[\mathsf{TryAtTJ}^t_{i,j}]$. Observe that $\bigcup_{t=1}^{\infty}(X_t \notin A_t \cap \mathsf{TryAtTJ}^t_{i,j}) = \mathsf{TriesJ}_{i+1,j}$ since the intersection implies that the i-th try at time t was

[1] $\mathsf{TryJ}_{i,j}(\omega) = t$ does not exclude that the try at time t is successful.

unsuccessful. Together, we get

$$\begin{aligned}\mathbb{P}[\mathsf{TriesJ}_{i+1,j}] &= \mathbb{P}[\textstyle\bigcup_{t=1}^{\infty} X_t \notin A_t \cap \mathsf{TryAtTJ}^t_{i,j}] = \sum_{t=1}^{\infty} \mathbb{P}[X_t \notin A_t \cap \mathsf{TryAtTJ}^t_{i,j}] \\ &= \sum\nolimits_{t \in \mathsf{TryTimesJ}_{i,j}} \mathbb{P}[X_t \notin A_t \cap \mathsf{TryAtTJ}^t_{i,j}] \\ &< \sum\nolimits_{t=1}^{\infty} (1-\varepsilon) \cdot \mathbb{P}[\mathsf{TryAtTJ}^t_{i,j}] = (1-\varepsilon) \cdot \mathbb{P}[\textstyle\bigcup_{t=1}^{\infty} \mathsf{TryAtTJ}^t_{i,j}] \\ &= (1-\varepsilon) \cdot \mathbb{P}[\mathsf{TriesJ}_{i,j}].\end{aligned}$$

Consequently, $\lim_{i\to\infty} \mathbb{P}[\mathsf{TriesJ}_{i,j}] = 0$ for any fixed j.

As argued before, we have $\mathsf{TriesJ}_{i,j} = \mathsf{TriesJ}_{i+1,j} \cup \mathsf{TriesJ}_{1,j+1}$. Iterating this equation yields $\mathsf{TriesJ}_{i,j} = \mathsf{TriesJ}_{i+k,j} \cup \mathsf{TriesJ}_{1,j+1}$ for any $k \geq 1$ and consequently $\mathsf{TriesJ}_{1,j} = \bigcap_{i=1}^{\infty} \mathsf{TriesJ}_{i,j} \cup \mathsf{TriesJ}_{1,j+1}$. Informally, this equation can be read as 'all outcomes which succeed at least j times either try infinitely often or succeed at least $j+1$ times.' Let $\mathsf{TriesJ}_{\infty,j} = \bigcap_{i=1}^{\infty} \mathsf{TriesJ}_{i,j} = \{\omega \in \Omega' \mid \mathsf{Succ}_j(\omega) < \infty = \mathsf{Succ}_{j+1}(\omega)\}$. Clearly, $\mathsf{TriesJ}_{\infty,j} \cap \mathsf{TriesJ}_{1,j+1} = \emptyset$, thus we have $\mathbb{P}[\mathsf{TriesJ}_{1,j+1} \setminus \mathsf{TriesJ}_{1,j}] = \mathbb{P}[\mathsf{TriesJ}_{\infty,j}]$. Additionally, we have $\mathbb{P}[\mathsf{TriesJ}_{\infty,j}] = \inf_{i\in\mathbb{N}} \mathbb{P}[\mathsf{TriesJ}_{i,j}] = 0$ by the above reasoning. Hence $\mathbb{P}[\mathsf{TriesJ}_{1,j+1} \setminus \mathsf{TriesJ}_{1,j}] = 0$. This implies that almost all runs in Ω' succeed infinitely often, concluding the proof. □

Bibliography

[AKW19] Pranav Ashok, Jan Kretínský and Maximilian Weininger. 'PAC Statistical Model Checking for Markov Decision Processes and Stochastic Games'. In: *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*. Ed. by Isil Dillig and Serdar Tasiran. Vol. 11561. Lecture Notes in Computer Science. Springer, 2019, pp. 497–519. DOI: 10.1007/978-3-030-25540-4_29. URL: https://doi.org/10.1007/978-3-030-25540-4%5C_29.

[AL09] Husain Aljazzar and Stefan Leue. 'Generation of Counterexamples for Model Checking of Markov Decision Processes'. In: *QEST 2009, Sixth International Conference on the Quantitative Evaluation of Systems, Budapest, Hungary, 13-16 September 2009*. IEEE Computer Society, 2009, pp. 197–206. DOI: 10.1109/QEST.2009.10. URL: https://doi.org/10.1109/QEST.2009.10.

[Ang88] Dana Angluin. 'Learning With Hints'. In: *Proceedings of the First Annual Workshop on Computational Learning Theory, COLT '88, Cambridge, MA, USA, August 3-5, 1988*. Ed. by David Haussler and Leonard Pitt. ACM/MIT, 1988, pp. 167–181. URL: http://dl.acm.org/citation.cfm?id=93075.

[Ash+17] Pranav Ashok, Krishnendu Chatterjee, Przemyslaw Daca, Jan Kretínský and Tobias Meggendorfer. 'Value Iteration for Long-Run Average Reward in Markov Decision Processes'. In: *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*. Ed. by Rupak Majumdar and Viktor Kuncak. Vol. 10426. Lecture Notes in Computer Science. Springer, 2017, pp. 201–221. DOI: 10.1007/978-3-319-63387-9_10. URL: https://doi.org/10.1007/978-3-319-63387-9%5C_10.

[Ash+18] Pranav Ashok, Yuliya Butkova, Holger Hermanns and Jan Kretínský. 'Continuous-Time Markov Decisions Based on Partial Exploration'. In: *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings*. Ed. by Shuvendu K. Lahiri and Chao Wang. Vol. 11138. Lecture Notes in Computer Science. Springer, 2018, pp. 317–334. DOI: 10.1007/978-3-030-01090-4_19. URL: https://doi.org/10.1007/978-3-030-01090-4%5C_19.

[Bah+97] R. Iris Bahar, Erica A. Frohm, Charles M. Gaona, Gary D. Hachtel, Enrico Macii, Abelardo Pardo and Fabio Somenzi. 'Algebraic Decision Diagrams and Their Applications'. In: *Formal Methods in System Design* 10.2/3 (1997), pp. 171–206. DOI: 10.1023/A:1008699807402. URL: https://doi.org/10.1023/A:1008699807402.

[Bai+17] Christel Baier, Joachim Klein, Linda Leuschner, David Parker and Sascha Wunderlich. 'Ensuring the Reliability of Your Model Checker: Interval Iteration for Markov Decision Processes'. In: *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I*. Ed. by Rupak Majumdar and Viktor Kuncak. Vol. 10426. Lecture Notes in Computer Science. Springer, 2017, pp. 160–180. DOI: 10.1007/978-3-319-63387-9_8. URL: https://doi.org/10.1007/978-3-319-63387-9%5C_8.

[Bar+08] Jiri Barnat, Lubos Brim, Ivana Černá, Milan Ceska and Jana Tumova. 'ProbDiVinE-MC: Multi-core LTL Model Checker for Probabilistic Systems'. In: *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008), 14-17 September 2008, Saint-Malo, France*. IEEE Computer Society, 2008, pp. 77–78. DOI: 10.1109/QEST.2008.29. URL: https://doi.org/10.1109/QEST.2008.29.

[BBR14] Aaron Bohy, Véronique Bruyère and Jean-François Raskin. 'Symblicit algorithms for optimal strategy synthesis in monotonic Markov decision processes'. In: *Proceedings 3rd Workshop on Synthesis, SYNT 2014, Vienna, Austria, July 23-24, 2014*. Ed. by Krishnendu Chatterjee, Rüdiger Ehlers and Susmit Jha. Vol. 157. EPTCS. 2014, pp. 51–67. DOI: 10.4204/EPTCS.157.8. URL: https://doi.org/10.4204/EPTCS.157.8.

[BBS95] Andrew G. Barto, Steven J. Bradtke and Satinder P. Singh. 'Learning to Act Using Real-Time Dynamic Programming'. In: *Artif. Intell.* 72.1-2 (1995), pp. 81–138. DOI: 10.1016/0004-3702(94)00011-O. URL: https://doi.org/10.1016/0004-3702(94)00011-O.

[BDG06] Christel Baier, Pedro R. D'Argenio and Marcus Größer. 'Partial Order Reduction for Probabilistic Branching Time'. In: *Electron. Notes Theor. Comput. Sci.* 153.2 (2006), pp. 97–116. DOI: 10.1016/j.entcs.2005.10.034. URL: https://doi.org/10.1016/j.entcs.2005.10.034.

[BDH17] Carlos E. Budde, Pedro R. D'Argenio and Arnd Hartmanns. 'Better Automated Importance Splitting for Transient Rare Events'. In: *Dependable Software Engineering. Theories, Tools, and Applications - Third International Symposium, SETTA 2017, Changsha, China, October 23-25, 2017, Proceedings*. Ed. by Kim Guldstrand Larsen, Oleg Sokolsky and Ji Wang. Vol. 10606. Lecture Notes in Computer Science. Springer, 2017, pp. 42–58. DOI: 10.1007/978-3-319-69483-2_3. URL: https://doi.org/10.1007/978-3-319-69483-2%5C_3.

[Bel57] Richard Bellman. 'A Markovian decision process'. In: *Journal of mathematics and mechanics* (1957), pp. 679–684.

[Bel66] Richard Bellman. 'Dynamic programming'. In: *Science* 153.3731 (1966), pp. 34–37.

[Ber17] Dimitri P. Bertsekas. 'Value and Policy Iterations in Optimal Control and Adaptive Dynamic Programming'. In: *IEEE Trans. Neural Netw. Learning Syst.* 28.3 (2017), pp. 500–509. DOI: 10.1109/TNNLS.2015.2503980. URL: https://doi.org/10.1109/TNNLS.2015.2503980.

[BGC04] Christel Baier, Marcus Größer and Frank Ciesinski. 'Partial Order Reduction for Probabilistic Systems'. In: *1st International Conference on Quantitative Evaluation of Systems (QEST 2004), 27-30 September 2004, Enschede, The Netherlands*. IEEE Computer Society, 2004, pp. 230–239. DOI: 10.1109/QEST.2004.1348037. URL: https://doi.org/10.1109/QEST.2004.1348037.

[BHH12] Jonathan Bogdoll, Arnd Hartmanns and Holger Hermanns. 'Simulation and Statistical Model Checking for Modestly Nondeterministic Models'. In: *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance - 16th International GI/ITG Conference, MMB & DFT 2012, Kaiserslautern, Germany, March 19-21, 2012. Proceedings*. Ed. by Jens B. Schmitt. Vol. 7201. Lecture Notes in Computer Science. Springer, 2012, pp. 249–252. DOI: 10.1007/978-3-642-28540-0_20. URL: https://doi.org/10.1007/978-3-642-28540-0%5C_20.

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008. ISBN: 978-0-262-02649-9.

[BKH99] Christel Baier, Joost-Pieter Katoen and Holger Hermanns. 'Approximate Symbolic Model Checking of Continuous-Time Markov Chains'. In: *CONCUR '99: Concurrency Theory, 10th International Conference, Eindhoven, The Netherlands, August 24-27, 1999, Proceedings*. Ed. by Jos C. M. Baeten and Sjouke Mauw. Vol. 1664. Lecture Notes in Computer Science. Springer, 1999, pp. 146–161. DOI: 10.1007/3-540-48320-9_12. URL: https://doi.org/10.1007/3-540-48320-9%5C_12.

[BKK14] Tomás Brázdil, Stefan Kiefer and Antonín Kucera. 'Efficient Analysis of Probabilistic Programs with an Unbounded Counter'. In: *J. ACM* 61.6 (2014), 41:1–41:35. DOI: 10.1145/2629599. URL: https://doi.org/10.1145/2629599.

[BKW14] Nicolas Basset, Marta Z. Kwiatkowska and Clemens Wiltsche. 'Compositional Controller Synthesis for Stochastic Games'. In: *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*. Ed. by Paolo Baldan and Daniele Gorla. Vol. 8704. Lecture Notes in Computer Science. Springer, 2014, pp. 173–187. DOI: 10.1007/978-3-662-44584-6_13. URL: https://doi.org/10.1007/978-3-662-44584-6%5C_13.

[BKW18] Nicolas Basset, Marta Z. Kwiatkowska and Clemens Wiltsche. 'Compositional strategy synthesis for stochastic games with multiple objectives'. In: *Inf. Comput.* 261.Part (2018), pp. 536–587. DOI: 10.1016/j.ic.2017.09.010. URL: https://doi.org/10.1016/j.ic.2017.09.010.

[Bog+11] Jonathan Bogdoll, Luis María Ferrer Fioriti, Arnd Hartmanns and Holger Hermanns. 'Partial Order Methods for Statistical Model Checking and Simulation'. In: *Formal Techniques for Distributed Systems - Joint 13th IFIP WG 6.1 International Conference, FMOODS 2011, and 31st IFIP WG 6.1 International Conference, FORTE 2011, Reykjavik, Iceland, June 6-9, 2011. Proceedings*. Ed. by Roberto Bruni and Jürgen Dingel. Vol. 6722. Lecture Notes in Computer Science. Springer, 2011, pp. 59–74. DOI: 10.1007/978-3-642-21461-5_4. URL: https://doi.org/10.1007/978-3-642-21461-5%5C_4.

[Boh+14] Dimitri Bohlender, Harold Bruintjes, Sebastian Junges, Jens Katelaan, Viet Yen Nguyen and Thomas Noll. 'A Review of Statistical Model Checking Pitfalls on Real-Time Stochastic Models'. In: *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications - 6th International Symposium, ISoLA 2014, Imperial, Corfu, Greece, October 8-11, 2014, Proceedings, Part II*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 8803. Lecture Notes in Computer Science. Springer, 2014, pp. 177–192. DOI: 10.1007/978-3-662-45231-8_13. URL: https://doi.org/10.1007/978-3-662-45231-8%5C_13.

[Bøn+19] Frederik M. Bønneland, Peter Gjøl Jensen, Kim G. Larsen, Marco Muñiz and Jirí Srba. 'Partial Order Reduction for Reachability Games'. In: *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*. Ed. by Wan Fokkink and Rob van Glabbeek. Vol. 140. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 23:1–23:15. DOI: 10.4230/LIPIcs.CONCUR.2019.23. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2019.23.

[Boy+13] Benoît Boyer, Kevin Corre, Axel Legay and Sean Sedwards. 'PLASMA-lab: A Flexible, Distributable Statistical Model Checking Library'. In: *Quantitative Evaluation of Systems - 10th International Conference, QEST 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*. Ed. by Kaustubh R. Joshi, Markus Siegle, Mariëlle Stoelinga and Pedro R. D'Argenio. Vol. 8054. Lecture Notes in Computer Science. Springer, 2013, pp. 160–164. DOI: 10.1007/978-3-642-40196-1_12. URL: https://doi.org/10.1007/978-3-642-40196-1%5C_12.

[Boz+19] Alper Kamil Bozkurt, Yu Wang, Michael M. Zavlanos and Miroslav Pajic. 'Control Synthesis from Linear Temporal Logic Specifications using Model-Free Reinforcement Learning'. In: *CoRR* abs/1909.07299 (2019). arXiv: 1909.07299. URL: http://arxiv.org/abs/1909.07299.

[Brá+14] Tomás Brázdil, Krishnendu Chatterjee, Martin Chmelik, Vojtech Forejt, Jan Kretínský, Marta Z. Kwiatkowska, David Parker and Mateusz Ujma. 'Verification of Markov Decision Processes Using Learning Algorithms'. In: *Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014,*

Proceedings. Ed. by Franck Cassez and Jean-François Raskin. Vol. 8837. Lecture Notes in Computer Science. Springer, 2014, pp. 98–114. DOI: 10.1007/978-3-319-11936-6_8. URL: https://doi.org/10.1007/978-3-319-11936-6%5C_8.

[Bry86] Randal E. Bryant. 'Graph-Based Algorithms for Boolean Function Manipulation'. In: *IEEE Trans. Computers* 35.8 (1986), pp. 677–691. DOI: 10.1109/TC.1986.1676819. URL: https://doi.org/10.1109/TC.1986.1676819.

[Bud+17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges and Andrea Turrini. 'JANI: Quantitative Model and Tool Interaction'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*. Ed. by Axel Legay and Tiziana Margaria. Vol. 10206. Lecture Notes in Computer Science. 2017, pp. 151–168. DOI: 10.1007/978-3-662-54580-5_9. URL: https://doi.org/10.1007/978-3-662-54580-5%5C_9.

[Bud+18] Carlos E. Budde, Pedro R. D'Argenio, Arnd Hartmanns and Sean Sedwards. 'A Statistical Model Checker for Nondeterminism and Rare Events'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II*. Ed. by Dirk Beyer and Marieke Huisman. Vol. 10806. Lecture Notes in Computer Science. Springer, 2018, pp. 340–358. DOI: 10.1007/978-3-319-89963-3_20. URL: https://doi.org/10.1007/978-3-319-89963-3%5C_20.

[Bul+12] Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikucionis, Danny Bøgsted Poulsen, Axel Legay and Zheng Wang. 'UPPAAL-SMC: Statistical Model Checking for Priced Timed Automata'. In: *Proceedings 10th Workshop on Quantitative Aspects of Programming Languages and Systems, QAPL 2012, Tallinn, Estonia, 31 March and 1 April 2012*. Ed. by Herbert Wiklicky and Mieke Massink. Vol. 85. EPTCS. 2012, pp. 1–16. DOI: 10.4204/EPTCS.85.1. URL: https://doi.org/10.4204/EPTCS.85.1.

[Cai+10] Benoît Caillaud, Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen and Andrzej Wasowski. 'Compositional Design Methodology with Constraint Markov Chains'. In: *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Virginia, USA, 15-18 September 2010*. IEEE Computer Society, 2010, pp. 123–132. DOI: 10.1109/QEST.2010.23. URL: https://doi.org/10.1109/QEST.2010.23.

[CCD15] Krishnendu Chatterjee, Martin Chmelik and Przemyslaw Daca. 'CEGAR for compositional analysis of qualitative properties in Markov decision processes'. In: *Formal Methods Syst. Des.* 47.2 (2015), pp. 230–264. DOI: 10.1007/

s10703-015-0235-2. URL: https://doi.org/10.1007/s10703-015-0235-2.

[CH11] Krishnendu Chatterjee and Monika Henzinger. 'Faster and Dynamic Algorithms for Maximal End-Component Decomposition and Related Graph Problems in Probabilistic Verification'. In: *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, San Francisco, California, USA, January 23-25, 2011*. Ed. by Dana Randall. SIAM, 2011, pp. 1318–1336. DOI: 10.1137/1.9781611973082.101. URL: https://doi.org/10.1137/1.9781611973082.101.

[CH12] Krishnendu Chatterjee and Monika Henzinger. 'An $O(n^2)$ time algorithm for alternating Büchi games'. In: *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*. Ed. by Yuval Rabani. SIAM, 2012, pp. 1386–1399. DOI: 10.1137/1.9781611973099.109. URL: https://doi.org/10.1137/1.9781611973099.109.

[CH14] Krishnendu Chatterjee and Monika Henzinger. 'Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-Component Decomposition'. In: *J. ACM* 61.3 (2014), 15:1–15:40. DOI: 10.1145/2597631. URL: https://doi.org/10.1145/2597631.

[Cha+13] Hyeong Soo Chang, Jiaqiao Hu, Michael C Fu and Steven I Marcus. *Simulation-based algorithms for Markov decision processes*. Springer Science & Business Media, 2013.

[Cie+08] Frank Ciesinski, Christel Baier, Marcus Größer and Joachim Klein. 'Reduction Techniques for Model Checking Markov Decision Processes'. In: *Fifth International Conference on the Quantitative Evaluaiton of Systems (QEST 2008), 14-17 September 2008, Saint-Malo, France*. IEEE Computer Society, 2008, pp. 45–54. DOI: 10.1109/QEST.2008.45. URL: https://doi.org/10.1109/QEST.2008.45.

[CY90] Costas Courcoubetis and Mihalis Yannakakis. 'Markov decision processes and regular events'. In: *Automata, Languages and Programming*. Ed. by Michael S. Paterson. Berlin, Heidelberg: Springer Berlin Heidelberg, 1990, pp. 336–349. ISBN: 978-3-540-47159-2.

[CY95] Costas Courcoubetis and Mihalis Yannakakis. 'The Complexity of Probabilistic Verification'. In: *J. ACM* 42.4 (1995), pp. 857–907. DOI: 10.1145/210332.210339. URL: https://doi.org/10.1145/210332.210339.

[DAr+02] Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen and Kim Guldstrand Larsen. 'Reduction and Refinement Strategies for Probabilistic Analysis'. In: *Process Algebra and Probabilistic Methods, Performance Modeling and Verification, Second Joint International Workshop PAPM-PROBMIV 2002, Copenhagen, Denmark, July 25-26, 2002, Proceedings*. Ed. by Holger

Hermanns and Roberto Segala. Vol. 2399. Lecture Notes in Computer Science. Springer, 2002, pp. 57–76. DOI: 10.1007/3-540-45605-8_5. URL: https://doi.org/10.1007/3-540-45605-8%5C_5.

[DAr+15] Pedro D'Argenio, Axel Legay, Sean Sedwards and Louis-Marie Traonouez. 'Smart sampling for lightweight verification of Markov decision processes'. In: *Int. J. Softw. Tools Technol. Transf.* 17.4 (2015), pp. 469–484. DOI: 10.1007/s10009-015-0383-0. URL: https://doi.org/10.1007/s10009-015-0383-0.

[Dav+11a] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, Danny Bøgsted Poulsen, Jonas van Vliet and Zheng Wang. 'Statistical Model Checking for Networks of Priced Timed Automata'. In: *Formal Modeling and Analysis of Timed Systems - 9th International Conference, FORMATS 2011, Aalborg, Denmark, September 21-23, 2011. Proceedings*. Ed. by Uli Fahrenberg and Stavros Tripakis. Vol. 6919. Lecture Notes in Computer Science. Springer, 2011, pp. 80–96. DOI: 10.1007/978-3-642-24310-3_7. URL: https://doi.org/10.1007/978-3-642-24310-3%5C_7.

[Dav+11b] Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis and Zheng Wang. 'Time for Statistical Model Checking of Real-Time Systems'. In: *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 349–355. DOI: 10.1007/978-3-642-22110-1_27. URL: https://doi.org/10.1007/978-3-642-22110-1%5C_27.

[Dav+15] Alexandre David, Peter Gjøl Jensen, Kim Guldstrand Larsen, Marius Mikucionis and Jakob Haahr Taankvist. 'Uppaal Stratego'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*. Ed. by Christel Baier and Cesare Tinelli. Vol. 9035. Lecture Notes in Computer Science. Springer, 2015, pp. 206–211. DOI: 10.1007/978-3-662-46681-0_16. URL: https://doi.org/10.1007/978-3-662-46681-0%5C_16.

[De 97] Luca De Alfaro. *Formal verification of probabilistic systems*. 1601. Citeseer, 1997.

[Deh+17] Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen and Matthias Volk. 'A Storm is Coming: A Modern Probabilistic Model Checker'. In: *Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II*. Ed. by Rupak Majumdar and Viktor Kuncak. Vol. 10427. Lecture Notes in Computer Science. Springer, 2017, pp. 592–600. DOI: 10.1007/978-3-319-63390-9_31. URL: https://doi.org/10.1007/978-3-319-63390-9%5C_31.

[DH13] Yuxin Deng and Matthew Hennessy. 'Compositional reasoning for weighted Markov decision processes'. In: *Sci. Comput. Program.* 78.12 (2013), pp. 2537–2579. DOI: 10.1016/j.scico.2013.02.009. URL: https://doi.org/10.1016/j.scico.2013.02.009.

[DHS18] Pedro R. D'Argenio, Arnd Hartmanns and Sean Sedwards. 'Lightweight Statistical Model Checking in Nondeterministic Continuous Time'. In: *Leveraging Applications of Formal Methods, Verification and Validation. Verification - 8th International Symposium, ISoLA 2018, Limassol, Cyprus, November 5-9, 2018, Proceedings, Part II*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 11245. Lecture Notes in Computer Science. Springer, 2018, pp. 336–353. DOI: 10.1007/978-3-030-03421-4_22. URL: https://doi.org/10.1007/978-3-030-03421-4%5C_22.

[Día+12] Álvaro Fernández Díaz, Christel Baier, Clara Benac Earle and Lars-Åke Fredlund. 'Static Partial Order Reduction for Probabilistic Concurrent Systems'. In: *Ninth International Conference on Quantitative Evaluation of Systems, QEST 2012, London, United Kingdom, September 17-20, 2012*. IEEE Computer Society, 2012, pp. 104–113. DOI: 10.1109/QEST.2012.22. URL: https://doi.org/10.1109/QEST.2012.22.

[FHM18] Chuchu Fan, Zhenqi Huang and Sayan Mitra. 'Approximate Partial Order Reduction'. In: *Formal Methods - 22nd International Symposium, FM 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 15-17, 2018, Proceedings*. Ed. by Klaus Havelund, Jan Peleska, Bill Roscoe and Erik P. de Vink. Vol. 10951. Lecture Notes in Computer Science. Springer, 2018, pp. 588–607. DOI: 10.1007/978-3-319-95582-7_35. URL: https://doi.org/10.1007/978-3-319-95582-7%5C_35.

[FKP11] Lu Feng, Marta Z. Kwiatkowska and David Parker. 'Automated Learning of Probabilistic Assumptions for Compositional Reasoning'. In: *Fundamental Approaches to Software Engineering - 14th International Conference, FASE 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*. Ed. by Dimitra Giannakopoulou and Fernando Orejas. Vol. 6603. Lecture Notes in Computer Science. Springer, 2011, pp. 2–17. DOI: 10.1007/978-3-642-19811-3_2. URL: https://doi.org/10.1007/978-3-642-19811-3%5C_2.

[FMY97] Masahiro Fujita, Patrick C. McGeer and Jerry Chih-Yuan Yang. 'Multi-Terminal Binary Decision Diagrams: An Efficient Data Structure for Matrix Representation'. In: *Formal Methods in System Design* 10.2/3 (1997), pp. 149–169. DOI: 10.1023/A:1008647823331. URL: https://doi.org/10.1023/A:1008647823331.

[For+11] Vojtech Forejt, Marta Z. Kwiatkowska, Gethin Norman and David Parker. 'Automated Verification Techniques for Probabilistic Systems'. In: *Formal*

Methods for Eternal Networked Software Systems - 11th International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM 2011, Bertinoro, Italy, June 13-18, 2011. Advanced Lectures. Ed. by Marco Bernardo and Valérie Issarny. Vol. 6659. Lecture Notes in Computer Science. Springer, 2011, pp. 53–113. DOI: 10.1007/978-3-642-21455-4_3. URL: https://doi.org/10.1007/978-3-642-21455-4%5C_3.

[FT14] Jie Fu and Ufuk Topcu. 'Probably Approximately Correct MDP Learning and Control With Temporal Logic Constraints'. In: *Robotics: Science and Systems X, University of California, Berkeley, USA, July 12-16, 2014*. Ed. by Dieter Fox, Lydia E. Kavraki and Hanna Kurniawati. 2014. DOI: 10.15607/RSS.2014.X.039. URL: http://www.roboticsproceedings.org/rss10/p39.html.

[FV96] Jerzy Filar and Koos Vrieze. 'Competitive Markov decision processes'. In: (1996).

[Hah+10] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter and Lijun Zhang. 'PASS: Abstraction Refinement for Infinite Probabilistic Models'. In: *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*. Ed. by Javier Esparza and Rupak Majumdar. Vol. 6015. Lecture Notes in Computer Science. Springer, 2010, pp. 353–357. DOI: 10.1007/978-3-642-12002-2_30. URL: https://doi.org/10.1007/978-3-642-12002-2%5C_30.

[He+10] Ru He, Paul Jennings, Samik Basu, Arka P. Ghosh and Huaiqing Wu. 'A bounded statistical approach for model checking of unbounded until properties'. In: *ASE 2010, 25th IEEE/ACM International Conference on Automated Software Engineering, Antwerp, Belgium, September 20-24, 2010*. Ed. by Charles Pecheur, Jamie Andrews and Elisabetta Di Nitto. ACM, 2010, pp. 225–234. DOI: 10.1145/1858996.1859043. URL: https://doi.org/10.1145/1858996.1859043.

[Hen+12] David Henriques, João G. Martins, Paolo Zuliani, André Platzer and Edmund M. Clarke. 'Statistical Model Checking for Markov Decision Processes'. In: *Ninth International Conference on Quantitative Evaluation of Systems, QEST 2012, London, United Kingdom, September 17-20, 2012*. IEEE Computer Society, 2012, pp. 84–93. DOI: 10.1109/QEST.2012.19. URL: https://doi.org/10.1109/QEST.2012.19.

[Hér+04] Thomas Hérault, Richard Lassaigne, Frédéric Magniette and Sylvain Peyronnet. 'Approximate Probabilistic Model Checking'. In: *Verification, Model Checking, and Abstract Interpretation, 5th International Conference, VMCAI 2004, Venice, Italy, January 11-13, 2004, Proceedings*. Ed. by Bernhard Steffen and Giorgio Levi. Vol. 2937. Lecture Notes in Computer Science.

Springer, 2004, pp. 73–84. DOI: 10.1007/978-3-540-24622-0_8. URL: https://doi.org/10.1007/978-3-540-24622-0%5C_8.

[HH14] Arnd Hartmanns and Holger Hermanns. 'The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings*. Ed. by Erika Ábrahám and Klaus Havelund. Vol. 8413. Lecture Notes in Computer Science. Springer, 2014, pp. 593–598. DOI: 10.1007/978-3-642-54862-8_51. URL: https://doi.org/10.1007/978-3-642-54862-8%5C_51.

[HKK13] Holger Hermanns, Jan Krcál and Jan Kretínský. 'Compositional Verification and Optimization of Interactive Markov Chains'. In: *CONCUR 2013 - Concurrency Theory - 24th International Conference, CONCUR 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*. Ed. by Pedro R. D'Argenio and Hernán C. Melgratti. Vol. 8052. Lecture Notes in Computer Science. Springer, 2013, pp. 364–379. DOI: 10.1007/978-3-642-40184-8_26. URL: https://doi.org/10.1007/978-3-642-40184-8%5C_26.

[HM14] Serge Haddad and Benjamin Monmege. 'Reachability in MDPs: Refining Convergence of Value Iteration'. In: *Reachability Problems - 8th International Workshop, RP 2014, Oxford, UK, September 22-24, 2014. Proceedings*. Ed. by Joël Ouaknine, Igor Potapov and James Worrell. Vol. 8762. Lecture Notes in Computer Science. Springer, 2014, pp. 125–137. DOI: 10.1007/978-3-319-11439-2_10. URL: https://doi.org/10.1007/978-3-319-11439-2%5C_10.

[HM18] Serge Haddad and Benjamin Monmege. 'Interval iteration algorithm for MDPs and IMDPs'. In: *Theor. Comput. Sci.* 735 (2018), pp. 111–131. DOI: 10.1016/j.tcs.2016.12.003. URL: https://doi.org/10.1016/j.tcs.2016.12.003.

[Hoe94] Wassily Hoeffding. 'Probability inequalities for sums of bounded random variables'. In: *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[How60] Ronald A Howard. 'Dynamic programming and markov processes.' In: (1960).

[HWZ08] Holger Hermanns, Björn Wachter and Lijun Zhang. 'Probabilistic CEGAR'. In: *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*. Ed. by Aarti Gupta and Sharad Malik. Vol. 5123. Lecture Notes in Computer Science. Springer, 2008, pp. 162–175. DOI: 10.1007/978-3-540-70545-1_16. URL: https://doi.org/10.1007/978-3-540-70545-1%5C_16.

[JLS12] Cyrille Jégourel, Axel Legay and Sean Sedwards. 'A Platform for High Performance Statistical Model Checking - PLASMA'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 18th International Conference, TACAS 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012. Proceedings*. Ed. by Cormac Flanagan and Barbara König. Vol. 7214. Lecture Notes in Computer Science. Springer, 2012, pp. 498–503. DOI: 10.1007/978-3-642-28756-5_37. URL: https://doi.org/10.1007/978-3-642-28756-5%5C_37.

[JLS13] Cyrille Jégourel, Axel Legay and Sean Sedwards. 'Importance Splitting for Statistical Model Checking Rare Properties'. In: *Computer Aided Verification - 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings*. Ed. by Natasha Sharygina and Helmut Veith. Vol. 8044. Lecture Notes in Computer Science. Springer, 2013, pp. 576–591. DOI: 10.1007/978-3-642-39799-8_38. URL: https://doi.org/10.1007/978-3-642-39799-8%5C_38.

[Jon+15] Austin Jones, Derya Aksaray, Zhaodan Kong, Mac Schwager and Calin Belta. 'Robust Satisfaction of Temporal Logic Specifications via Reinforcement Learning'. In: *CoRR* abs/1510.06460 (2015). arXiv: 1510.06460. URL: http://arxiv.org/abs/1510.06460.

[Kar84] Narendra Karmarkar. 'A new polynomial-time algorithm for linear programming'. In: *Combinatorica* 4.4 (1984), pp. 373–396. DOI: 10.1007/BF02579150. URL: https://doi.org/10.1007/BF02579150.

[Kat+10] Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman and David Parker. 'A game-based abstraction-refinement framework for Markov decision processes'. In: *Formal Methods Syst. Des.* 36.3 (2010), pp. 246–280. DOI: 10.1007/s10703-010-0097-6. URL: https://doi.org/10.1007/s10703-010-0097-6.

[KCC05] Stuart Kurkowski, Tracy Camp and Michael Colagrosso. 'MANET simulation studies: the incredibles'. In: *Mobile Computing and Communications Review* 9.4 (2005), pp. 50–61. DOI: 10.1145/1096166.1096174. URL: https://doi.org/10.1145/1096166.1096174.

[Kel+18] Edon Kelmendi, Julia Krämer, Jan Kretínský and Maximilian Weininger. 'Value Iteration for Simple Stochastic Games: Stopping Criterion and Learning Algorithm'. In: *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*. Ed. by Hana Chockler and Georg Weissenbacher. Vol. 10981. Lecture Notes in Computer Science. Springer, 2018, pp. 623–642. DOI: 10.1007/978-3-319-96145-3_36. URL: https://doi.org/10.1007/978-3-319-96145-3%5C_36.

[Kha79] Leonid G Khachiyan. 'A polynomial algorithm in linear programming'. In: *Doklady Academii Nauk SSSR*. Vol. 244. 1979, pp. 1093–1096.

[Kle+16] Joachim Klein, Christel Baier, Philipp Chrszon, Marcus Daum, Clemens Dubslaff, Sascha Klüppelholz, Steffen Märcker and David Müller. 'Advances in Symbolic Probabilistic Model Checking with PRISM'. In: *Tools and Algorithms for the Construction and Analysis of Systems - 22nd International Conference, TACAS 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*. Ed. by Marsha Chechik and Jean-François Raskin. Vol. 9636. Lecture Notes in Computer Science. Springer, 2016, pp. 349–366. DOI: 10.1007/978-3-662-49674-9_20. URL: https://doi.org/10.1007/978-3-662-49674-9%5C_20.

[KM17] Jan Kretínský and Tobias Meggendorfer. 'Efficient Strategy Iteration for Mean Payoff in Markov Decision Processes'. In: *Automated Technology for Verification and Analysis - 15th International Symposium, ATVA 2017, Pune, India, October 3-6, 2017, Proceedings*. Ed. by Deepak D'Souza and K. Narayan Kumar. Vol. 10482. Lecture Notes in Computer Science. Springer, 2017, pp. 380–399. DOI: 10.1007/978-3-319-68167-2_25. URL: https://doi.org/10.1007/978-3-319-68167-2%5C_25.

[KM19] Jan Kretínský and Tobias Meggendorfer. 'Of Cores: A Partial-Exploration Framework for Markov Decision Processes'. In: *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands.* Ed. by Wan Fokkink and Rob van Glabbeek. Vol. 140. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 5:1–5:17. DOI: 10.4230/LIPIcs.CONCUR.2019.5. URL: https://doi.org/10.4230/LIPIcs.CONCUR.2019.5.

[KMN02] Michael J. Kearns, Yishay Mansour and Andrew Y. Ng. 'A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes'. In: *Machine Learning* 49.2-3 (2002), pp. 193–208. DOI: 10.1023/A:1017932429737. URL: https://doi.org/10.1023/A:1017932429737.

[KNP04] Marta Z. Kwiatkowska, Gethin Norman and David Parker. 'Probabilistic symbolic model checking with PRISM: a hybrid approach'. In: *STTT* 6.2 (2004), pp. 128–142. DOI: 10.1007/s10009-004-0140-2. URL: https://doi.org/10.1007/s10009-004-0140-2.

[KNP11] Marta Z. Kwiatkowska, Gethin Norman and David Parker. 'PRISM 4.0: Verification of Probabilistic Real-Time Systems'. In: *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*. Ed. by Ganesh Gopalakrishnan and Shaz Qadeer. Vol. 6806. Lecture Notes in Computer Science. Springer, 2011, pp. 585–591. DOI: 10.1007/978-3-642-22110-1_47. URL: https://doi.org/10.1007/978-3-642-22110-1%5C_47.

[KNP12] Marta Z. Kwiatkowska, Gethin Norman and David Parker. 'The PRISM Benchmark Suite'. In: *Ninth International Conference on Quantitative Evaluation of Systems, QEST 2012, London, United Kingdom, September 17-20, 2012*. IEEE Computer Society, 2012, pp. 203–204. DOI: 10.1109/QEST.2012.14. URL: https://doi.org/10.1109/QEST.2012.14.

[Kol+11] Andrey Kolobov, Mausam, Daniel S. Weld and Hector Geffner. 'Heuristic Search for Generalized Stochastic Shortest Path MDPs'. In: *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*. Ed. by Fahiem Bacchus, Carmel Domshlak, Stefan Edelkamp and Malte Helmert. AAAI, 2011. URL: http://aaai.org/ocs/index.php/ICAPS/ICAPS11/paper/view/2682.

[KS02] Michael J. Kearns and Satinder P. Singh. 'Near-Optimal Reinforcement Learning in Polynomial Time'. In: *Machine Learning* 49.2-3 (2002), pp. 209–232. DOI: 10.1023/A:1017984413808. URL: https://doi.org/10.1023/A:1017984413808.

[Lar13] Kim Guldstrand Larsen. 'Priced Timed Automata and Statistical Model Checking'. In: *Integrated Formal Methods, 10th International Conference, IFM 2013, Turku, Finland, June 10-14, 2013. Proceedings*. Ed. by Einar Broch Johnsen and Luigia Petre. Vol. 7940. Lecture Notes in Computer Science. Springer, 2013, pp. 154–161. DOI: 10.1007/978-3-642-38613-8_11. URL: https://doi.org/10.1007/978-3-642-38613-8%5C_11.

[LP15] Richard Lassaigne and Sylvain Peyronnet. 'Approximate planning and verification for large Markov decision processes'. In: *STTT* 17.4 (2015), pp. 457–467. DOI: 10.1007/s10009-014-0344-z. URL: https://doi.org/10.1007/s10009-014-0344-z.

[LR85] Tze Leung Lai and Herbert Robbins. 'Asymptotically efficient adaptive allocation rules'. In: *Advances in applied mathematics* 6.1 (1985), pp. 4–22.

[LST14] Axel Legay, Sean Sedwards and Louis-Marie Traonouez. 'Scalable Verification of Markov Decision Processes'. In: *Software Engineering and Formal Methods - SEFM 2014 Collocated Workshops: HOFM, SAFOME, OpenCert, MoKMaSD, WS-FMDS, Grenoble, France, September 1-2, 2014, Revised Selected Papers*. Ed. by Carlos Canal and Akram Idani. Vol. 8938. Lecture Notes in Computer Science. Springer, 2014, pp. 350–362. DOI: 10.1007/978-3-319-15201-1_23. URL: https://doi.org/10.1007/978-3-319-15201-1%5C_23.

[MLG05] H. Brendan McMahan, Maxim Likhachev and Geoffrey J. Gordon. 'Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees'. In: *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*. Ed. by Luc De Raedt and Stefan Wrobel. Vol. 119. ACM International Conference Proceeding Series. ACM, 2005, pp. 569–576. DOI: 10.1145/1102351.1102423. URL: https://doi.org/10.1145/1102351.1102423.

[PGT03] Joelle Pineau, Geoffrey J. Gordon and Sebastian Thrun. 'Point-based value iteration: An anytime algorithm for POMDPs'. In: *IJCAI-03, Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, Acapulco, Mexico, August 9-15, 2003*. Ed. by Georg Gottlob and Toby Walsh. Morgan Kaufmann, 2003, pp. 1025–1032. URL: http://ijcai.org/Proceedings/03/Papers/147.pdf.

[Pnu77] Amir Pnueli. 'The Temporal Logic of Programs'. In: *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*. IEEE Computer Society, 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32. URL: https://doi.org/10.1109/SFCS.1977.32.

[Put94] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994. ISBN: 978-0-47161977-2. DOI: 10.1002/9780470316887. URL: https://doi.org/10.1002/9780470316887.

[QK18] Tim Quatmann and Joost-Pieter Katoen. 'Sound Value Iteration'. In: *Computer Aided Verification - 30th International Conference, CAV 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings, Part I*. Ed. by Hana Chockler and Georg Weissenbacher. Vol. 10981. Lecture Notes in Computer Science. Springer, 2018, pp. 643–661. DOI: 10.1007/978-3-319-96145-3_37. URL: https://doi.org/10.1007/978-3-319-96145-3%5C_37.

[Roo+17] Nima Roohi, Yu Wang, Matthew West, Geir E. Dullerud and Mahesh Viswanathan. 'Statistical Verification of the Toyota Powertrain Control Verification Benchmark'. In: *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC 2017, Pittsburgh, PA, USA, April 18-20, 2017*. Ed. by Goran Frehse and Sayan Mitra. ACM, 2017, pp. 65–70. DOI: 10.1145/3049797.3049804. URL: https://doi.org/10.1145/3049797.3049804.

[RP09] Diana El Rabih and Nihal Pekergin. 'Statistical Model Checking Using Perfect Simulation'. In: *Automated Technology for Verification and Analysis, 7th International Symposium, ATVA 2009, Macao, China, October 14-16, 2009. Proceedings*. Ed. by Zhiming Liu and Anders P. Ravn. Vol. 5799. Lecture Notes in Computer Science. Springer, 2009, pp. 120–134. DOI: 10.1007/978-3-642-04761-9_11. URL: https://doi.org/10.1007/978-3-642-04761-9%5C_11.

[SB98] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press, 1998. ISBN: 978-0-262-19398-6. URL: http://www.worldcat.org/oclc/37293240.

[Sch99] Alexander Schrijver. *Theory of linear and integer programming*. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 1999. ISBN: 978-0-471-98232-6.

[Seg96] Roberto Segala. 'Modeling and verification of randomized distributed real-time systems'. In: (1996).

[SLL09] Alexander L. Strehl, Lihong Li and Michael L. Littman. 'Reinforcement Learning in Finite MDPs: PAC Analysis'. In: *J. Mach. Learn. Res.* 10 (2009), pp. 2413–2444. URL: https://dl.acm.org/citation.cfm?id=1755867.

[Str+06] Alexander L. Strehl, Lihong Li, Eric Wiewiora, John Langford and Michael L. Littman. 'PAC model-free reinforcement learning'. In: *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*. Ed. by William W. Cohen and Andrew Moore. Vol. 148. ACM International Conference Proceeding Series. ACM, 2006, pp. 881–888. DOI: 10.1145/1143844.1143955. URL: https://doi.org/10.1145/1143844.1143955.

[Str08] Alexander L. Strehl. 'Probably Approximately Correct (PAC) Exploration in Reinforcement Learning'. In: *International Symposium on Artificial Intelligence and Mathematics, ISAIM 2008, Fort Lauderdale, Florida, USA, January 2-4, 2008*. 2008. URL: http://isaim2008.unl.edu/PAPERS/SS3-ActiveLearning/isaim08-alex-strehl.pdf.

[SVA04] Koushik Sen, Mahesh Viswanathan and Gul Agha. 'Statistical Model Checking of Black-Box Probabilistic Systems'. In: *Computer Aided Verification, 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004, Proceedings*. Ed. by Rajeev Alur and Doron A. Peled. Vol. 3114. Lecture Notes in Computer Science. Springer, 2004, pp. 202–215. DOI: 10.1007/978-3-540-27813-9_16. URL: https://doi.org/10.1007/978-3-540-27813-9%5C_16.

[SVA05a] Koushik Sen, Mahesh Viswanathan and Gul Agha. 'On Statistical Model Checking of Stochastic Systems'. In: *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*. Ed. by Kousha Etessami and Sriram K. Rajamani. Vol. 3576. Lecture Notes in Computer Science. Springer, 2005, pp. 266–280. DOI: 10.1007/11513988_26. URL: https://doi.org/10.1007/11513988%5C_26.

[SVA05b] Koushik Sen, Mahesh Viswanathan and Gul A. Agha. 'VESTA: A Statistical Model-checker and Analyzer for Probabilistic Systems'. In: *Second International Conference on the Quantitative Evaluaiton of Systems (QEST 2005), 19-22 September 2005, Torino, Italy*. IEEE Computer Society, 2005, pp. 251–252. DOI: 10.1109/QEST.2005.42. URL: https://doi.org/10.1109/QEST.2005.42.

[Sze10] Csaba Szepesvári. *Algorithms for Reinforcement Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2010. DOI: 10.2200/S00268ED1V01Y201005AIM009. URL: https://doi.org/10.2200/S00268ED1V01Y201005AIM009.

[Tar72] Robert Endre Tarjan. 'Depth-First Search and Linear Graph Algorithms'. In: *SIAM J. Comput.* 1.2 (1972), pp. 146–160. DOI: 10.1137/0201010. URL: https://doi.org/10.1137/0201010.

[Val84] Leslie G. Valiant. 'A Theory of the Learnable'. In: *Commun. ACM* 27.11 (1984), pp. 1134–1142. DOI: 10.1145/1968.1972. URL: https://doi.org/10.1145/1968.1972.

[WD92] Christopher JCH Watkins and Peter Dayan. 'Q-learning'. In: *Machine learning* 8.3-4 (1992), pp. 279–292.

[Whi85] Douglas J White. 'Real applications of Markov decision processes'. In: *Interfaces* 15.6 (1985), pp. 73–83.

[Whi88] Douglas J White. 'Further real applications of Markov decision processes'. In: *Interfaces* 18.5 (1988), pp. 55–61.

[Whi93] Douglas J White. 'A survey of applications of Markov decision processes'. In: *Journal of the operational research society* 44.11 (1993), pp. 1073–1096.

[Wim+10] Ralf Wimmer, Bettina Braitling, Bernd Becker, Ernst Moritz Hahn, Pepijn Crouzen, Holger Hermanns, Abhishek Dhama and Oliver E. Theel. 'Symblicit Calculation of Long-Run Averages for Concurrent Probabilistic Systems'. In: *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Virginia, USA, 15-18 September 2010*. IEEE Computer Society, 2010, pp. 27–36. DOI: 10.1109/QEST.2010.12. URL: https://doi.org/10.1109/QEST.2010.12.

[YCZ10] Håkan L. S. Younes, Edmund M. Clarke and Paolo Zuliani. 'Statistical Verification of Probabilistic Properties with Unbounded Until'. In: *Formal Methods: Foundations and Applications - 13th Brazilian Symposium on Formal Methods, SBMF 2010, Natal, Brazil, November 8-11, 2010, Revised Selected Papers*. Ed. by Jim Davies, Leila Silva and Adenilso da Silva Simão. Vol. 6527. Lecture Notes in Computer Science. Springer, 2010, pp. 144–160. DOI: 10.1007/978-3-642-19829-8_10. URL: https://doi.org/10.1007/978-3-642-19829-8%5C_10.

[You05] Håkan L. S. Younes. 'Ymer: A Statistical Model Checker'. In: *Computer Aided Verification, 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005, Proceedings*. Ed. by Kousha Etessami and Sriram K. Rajamani. Vol. 3576. Lecture Notes in Computer Science. Springer, 2005, pp. 429–433. DOI: 10.1007/11513988_43. URL: https://doi.org/10.1007/11513988%5C_43.

[YS02] Håkan L. S. Younes and Reid G. Simmons. 'Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling'. In: *Computer Aided Verification, 14th International Conference, CAV 2002,Copenhagen, Denmark, July 27-31, 2002, Proceedings*. Ed. by Ed Brinksma and Kim Guldstrand Larsen. Vol. 2404. Lecture Notes in Computer Science. Springer, 2002, pp. 223–

235. DOI: 10.1007/3-540-45657-0_17. URL: https://doi.org/10.1007/3-540-45657-0%5C_17.

[ZSF12] Zahra Zamani, Scott Sanner and Cheng Fang. 'Symbolic Dynamic Programming for Continuous State and Action MDPs'. In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. Ed. by Jörg Hoffmann and Bart Selman. AAAI Press, 2012. URL: http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5186.

List of Figures

List of Tables