A Implementation details

Algorithm 3 shows the extension of tabular Recursive Q-learning to the neural network setting by using the additional techniques introduced by DQN. The algorithm performs two steps per sample: the first computes v and pushes the required values to the replay buffer, and the second samples the replay buffer and updates the neural network via the mean squared error.

For sampling trajectories, we sampled actions with the standard ε-greedy policy. We matched the hyperparameters for (deep) Recursive Q-learning and (deep) Q-learning on each example, with the discount factor set to $\lambda = 1$ for the latter. The experiments were run on a server with 12 CPU cores and no GPU.

Hyperparameters.

Cloud

Parameter	Value
Test samples	100
Learning rate α	0.02
Exploration rate ε	0.1
Quantize amount	0.001

Spelunking

Parameter	Value
Test samples	100
Learning rate α	0.2
Exploration rate ε	0.1

Palindrome

Parameter	Value
Test samples	100
Initial ε	1
Final ε	0.1
Final ε timestep	$30k$
Buffer size	$20k$
Buffer warm up	$1k$
Batch size N	256
Update frequency C	500
Hidden layers	2
Hidden dimension	128
Activation function	tanh
Learning rate	0.0005
Optimizer	Adam

B Discussion on discounting

There are multiple choices for discounting in RMDPs. The most straightforward choice is that discounting by λ is equivalent to stopping the entire process with probability $1 - \lambda$. We call this step-wise discounting. One can transform this type of discounting into a total reward model satisfying Assumption 1 by adding a special exit (the *exit-lane*) which leads to the special exit in the box above with no reward. Unfortunately, step-wise discounting induces a multi-exit RMDP and policies may depend on the stack. This type of discounting is further discussed in Appendix F. An alternative choice for discounting in single-exit RMDPs is that discounting by λ is equivalent to stopping the current box—by leaving out of its only exit—with probability $1 - \lambda$. We call this box-wise discounting, which does not add any exits. With this type of discounting, the discount factor can be incorporated into Recursive Q-learning by simply multiplying the terms $\max_{a' \in A(S')} Q(s', a')$ and $\max_{a' \in A(s_{\text{exit}})} Q(s_{\text{exit}}, a')$ in Algorithm 2 by λ. However, box-wise discounting does not necessarily ensure properness for all single-exit RMDPs and discount factors $\lambda < 1$. Instead, the discount factor must be sufficiently small. Note that box-wise and step-wise discounting schemes are equivalent in MDPs (RMDPs with no recursive calls). Our results subsume these settings by considering total reward under the properness assumption.

C Exponential Succinctness of Hierchical MDPs (Proof from Section 1)

Lemma 1. *Hierarchical MDPs are exponentially more succinct than finite-state MDPs.*

Proof. This proof is adapted from a similar result on the (non-stochastic) recursive state machines from [2]. Consider a collection of hierarchical, deterministic MDPs $M_1, M_2, \ldots, M_n$. The MDP M_1 upon taking an action a gives a reward of 1 and terminates, while it gives a reward of 0 for any other action and moves to a sink. Each MDP M_i (for $i > 1$) upon an action a calls the MDP M_{i-1} twice in sequence and then accepts with a reward of 1, and for other actions it makes a transition to a sink

Algorithm 3: Deep Recursive Q-learning (DQN-style)

Buffer size N, update frequency C
Initialize network parameters θ
Set target network parameters $\theta^- \leftarrow \theta$
Initialize empty replay buffer
while *not converged* **do**
  $v \leftarrow \mathbf{0}$
  stack $\leftarrow \emptyset$
  Sample trajectory $\tau \sim \{(s, a, r, s'), ...\}$
  for s, a, r, s' *in* τ **do**
    `// Push update to replay buffer`
    if *entered box* **then**
      $\{s_{\text{exit}_1}, \ldots, s_{\text{exit}_n}\} \leftarrow \text{getExits}(s')$
      $v' \leftarrow [\max_{a' \in A(s_{\text{exit}_1})} Q(s_{\text{exit}_1}, v, a'; \theta), \ldots, \max_{a' \in A(s_{\text{exit}_n})} Q(s_{\text{exit}_n}, v, a'; \theta)]$
      $v'_{\min} \leftarrow \min(v')$
      $v' \leftarrow v' - v'_{\min}$
      buffer.add(*entered box*, (s, v, a, r, s', v'), $v'_{\min}$)
      stack.push(v)
      $v \leftarrow v'$
    else if *exited box* **then**
      $\{s_{\text{exit}_1}, \ldots, s_{\text{exit}_n}\} \leftarrow \text{getExits}(s)$
      Set k such that $s' = s_{\text{exit}_k}$
      buffer.add(*exited box*, (s, v, a, r, s', v), $v(k)$)
      $v \leftarrow$ stack.pop()
    else
      buffer.add(*normal*, (s, v a, r, s', v), $\perp$)
    end
    `// Update network`
    Sample minibatch $\{\textit{type}_j, (s_j, v_j, a_j, r_j, s'_j, v'_j), \textit{aux}_j\}_{j=1}^N$ of size N from replay buffer
    for $j = 1, \ldots, N$ **do**
      $$\text{targ}_j \leftarrow \begin{cases} r_j + \max_{a' \in A(s'_j)} Q(s'_j, v'_j, a'_j; \theta^-) + \textit{aux}_j & \textit{type}_j = \textit{entered box} \\ r_j + \textit{aux}_j & \textit{type}_j = \textit{exited box} \\ r_j + \max_{a \in A(s'_j)} Q(s'_j, v'_j, a'_j; \theta^-) & \textit{type}_j = \textit{normal} \end{cases}$$
    end
    $\mathcal{L} = \frac{1}{N} \sum_{j=1}^N (Q(s_j, v_j, a_j) - \text{targ}_j)^2$
    Update θ with respect to loss $\mathcal{L}$ with gradient descent
    Set $\theta^- \leftarrow \theta$ every C steps
  end
end
return θ

without providing any reward. This MDP M_n has the property that the optimal value is $2^n - 1$ and the optimal policy corresponds to choosing $2^n - 1$ a's in succession. This environment can only be expressed by a finite-state MDP with at least $2^n - 1$ states. □

D Proof of Theorem 1

Theorem 1 (Undecidability of the Strategy Existence Problem)**.** *Given a proper RMDP and a threshold D, deciding whether there exists a strategy with expected value greater than D is undecidable.*

Proof. We make use of the existing results for the *probabilistic finite automata* (PFAs) model. PFAs are essentially finite automata where nondeterminism is replaced by probabilistic transitions. Specially, when a letter is read, the next state is selected by chance with a fixed probability distribution

(that depends on the current state and letter only) instead of the controller selecting the new state. A word is accepted by a PFA if the probability of reaching the special accept state after reading this word is higher than a given fixed threshold $\lambda \in [0, 1]$. Madani, Hanks, and Condon [27] showed undecidability of checking the non-emptiness for $\lambda = 1/2$ and a *leaky PFA* that is a PFA in which at each step we stop the run (or, equivalently, move to a non-accepting state with a self loop) with a fixed probability γ.

As shown in [15], RMDPs can simulate probabilistic finite automata. We adapt this to show how proper RMDPs can simulate leaky PFAs. Such a RMDP has a single component consisting of a box corresponding to each input letter. The number of exits of this component is the number of states in the PFA plus one additional exit-lane exit. One of these exits corresponds to the special accepting state of the PFA. There is a single entry of this component which is the only non-trivial choice point for the controller. Each input letter has a corresponding action that leads to the box corresponding to this letter with probability $1 - \gamma$ and with probability γ to the exit-lane exit. There is a special start action that leads to the exit corresponding to the initial state of the PFA. Once a box corresponding to a letter, a, is exited the probabilistic transitions corresponding to the effect of reading a in the PFA takes place by transitioning to the exit of the component corresponding to the new state of the simulated PFA. An exit-lane exit of any box is connected directly to the exit-lane exit of the component it is in and the reward for such transitions is 0. Note that all that that happens once exit-lane is reached is that the whole content of the stack is popped which results in terminating without modifying the accumulated reward so far. Note that this RMDP is proper, because in each step there is a fixed γ chance of entering the exit-lane and terminating (once the whole stack is popped), so the expected number of steps taken by any strategy is finite.

We claim that the controller has a strategy that terminates at the accepting exit (with empty stack content) with probability $\geq \frac{1}{2}$ iff there exists a word accepted by a leaky PFA (and so its language is non-empty) with the same probability $\geq \frac{1}{2}$. The strategy would pick the letter of the input word in reverse by calling the corresponding box and once done select the special start action. It is easy to see that the behavior of the leaky PFA (including stopping with probability γ) is mimicked precisely.

Now, one can easily encode termination objective using rewards: assign reward of 1 to the final single transition just before the RMDP reaches the accepting exit with an empty stack content and reward of 0 to all other transitions. This shows that the the strategy existence problem for the expected total reward objectives is at least as hard as the strategy existence problem for termination objective which we showed to be as hard as non-emptiness of the language accepted by a leaky PFA; an undecidable problem. □

Note that the above proof carries over to a discounted setting, so the strategy existence problem in such a setting is also undecidable.

E Proof of Theorem 3

Theorem 3 (Fixed Point)**.** *If y is a fixed point of $OPT_{\texttt{recur}}$ and x is a fixed point of $OPT_{\texttt{cont}}$, then $y(\langle\emptyset\rangle, q) = x(\mathbf{0}, q)$. Moreover, any policy optimal from $(\mathbf{0}, q)$ is also optimal from $(\langle\emptyset\rangle, q)$.*

Proof. We first show that the fixed point y of $\text{OPT}_{\texttt{recur}}$ is unique and $y(\langle\kappa\rangle, q) = \text{ETotal}^M(\langle\kappa\rangle, q)$.

The values $\text{ETotal}^M(\langle\kappa\rangle, q)$ are indeed a fixed point of $\text{OPT}_{\texttt{recur}}$. Consider any state $\langle\kappa\rangle, q$. We proceed over all possible cases for the type of q.

If $q{=}(b, en) \in$ Call then $\text{ETotal}^M(\langle\kappa\rangle, q) = \text{ETotal}^M(\langle\kappa, b\rangle, en)$, and if $q \in Ex, (b, q) \in \text{Ret}(b), \kappa{=}\langle\kappa', b\rangle$ then $\text{ETotal}^M(\langle\kappa\rangle, q) = \text{ETotal}^M(\langle\kappa'\rangle, (b, q))$, both by definition of MDP $[\![M]\!]$. Similarly, if $q \in Ex$ then $\text{ETotal}^M(\langle\emptyset\rangle, q) = 0$ as we immediately terminate. Finally, if q is any other type of vertex, then

$$\text{ETotal}^M(\langle\kappa\rangle, q) = \max_{a \in A(q)} \Big\{ r(q, a) + \sum_{q' \in Q} p(q'|q, a) \text{ETotal}^M(\langle\kappa\rangle, q') \Big\},$$

because the best one can do while at $(\langle\kappa\rangle, q)$ is to pick an action that maximizes the one-step reward plus the weighted average of the expected total reward from the successor state.

On the other hand, any strategy that picks any such an action achieves $\text{ETotal}^M(\langle\kappa\rangle, q)$. Let us denote such a strategy by σ. This works because σ is guaranteed to terminate within a finite number of steps, K, when starting at any $(\langle\emptyset\rangle, q)$ due to Assumption 1. Note that the time to exit the current box when at any $(\langle\kappa\rangle, q)$ is also at most K. This is because when starting at $(\langle\kappa\rangle, q)$ and at $(\langle\emptyset\rangle, q)$ the transition structure of the model looks the same until an exit of the current box is reached. (Specifically, a mapping of $(\langle\kappa\kappa'\rangle, q)$ to $(\langle\kappa'\rangle, q)$ for every κ' and q is an isomorphism.) As a result, we get an upper bound of $K \cdot |\kappa|$ for the expected time to terminate from $(\langle\kappa\rangle, q)$ for any κ and q.

Note now that, for any $\epsilon > 0$, the probability of terminating when starting at $(\langle\kappa\rangle, q)$ after $K \cdot |\kappa|/\epsilon$ steps is at most ϵ, because otherwise the expected termination time would not $\leq K \cdot |\kappa|$. This gives us a bound of $K \cdot |\kappa| \cdot r_{max}$ for the expected total reward from $(\langle\kappa\rangle, q)$, because in each transition we can get at most r_{max}. In such a setting, the results from [32] imply that σ is optimal.

If there was another fixed point such that at least one coordinate is higher than ETotal^M, then a strategy constructed as above would obtain more than the supremum over all possible strategies of the expected total reward when starting at that state; a contradiction. On the other hand, if there was any other smaller fixed point than ETotal^M, then by using repeatedly using choices made by the strategy σ defined above we would converge at ETotal^M, but on the other we should never be able to improve based on the assumption that this is a fixed point; a contradiction. (For more details see the end of the proof of Theorem 5 in Appendix G).

Now suppose that there exists a fixed point point y of $\text{OPT}_{\texttt{recur}}$ and a fixed point x of $\text{OPT}_{\texttt{cont}}$ such that $y(\langle\emptyset\rangle, q) \neq x(\mathbf{0}, q)$. We now show how to reconstruct a fixed point $y'(\langle\kappa\rangle, q)$ of $\text{OPT}_{\texttt{recur}}$ that corresponds exactly to the fixed point $x(\mathbf{v}, q)$. This would lead to a contradiction as we just showed that $\text{OPT}_{\texttt{recur}}$ has a unique fixed point.

This will be done by a recursive parallel fixed point reconstruction process below. Let the current call stack be $\langle\kappa\rangle$ and valuation of the exits be $\mathbf{v}$. We start the process at $\kappa = \emptyset$ and $\mathbf{v} = \mathbf{0}$. We assign the values as follows.

$$y'(\langle\kappa\rangle, q) \quad = \quad \begin{cases} \text{make recursive call for } \kappa := \kappa b \text{ and } \mathbf{v} := (x(\mathbf{v}, q'))_{q' \in \text{Ret}_b} & q = (b, en) \in \text{Call} \\ \mathbf{v}(q) & q \in \textit{Ex} \\ \max\limits_{a \in A(q)} \left\{ r(q,a) + \sum\limits_{q' \in Q} p(q'|q,a) x(\mathbf{v}, q') \right\} & \text{otherwise.} \end{cases}$$

It is clear that the value of all the states are the same in y' and x, so the optimal strategies would be the same as well. □

F PAC Learnability for RMDPs with Discounting and the Milder Restrictions from Section 2

In this section, we start with defining discounted rewards with the usual semantics. We then provide PAC learning results for RMDPs with discounted rewards (instead of the requirements on the expected decline of the stack size and the expected rewards) (Section F.2), and then extend these results to the undiscounted case (Section F.3).

F.1 Discounted Rewards

In the discounted setting, the objective in a RMDP M is to find a strategy $\sigma \in \Sigma_M$ that maximizes the *discounted reward* $\text{EDisct}(\lambda)^M_\sigma(s)$, which is defined as

$$\text{EDisct}(\lambda)^M_\sigma(s) = \lim_{N \to \infty} \mathbb{E}^M_\sigma(s) \left\{ \sum_{1 \leq i \leq N} \lambda^{i-1} r(X_{i-1}, Y_i) \right\},$$

for some discount factor $0 \leq \lambda < 1$. The optimal (discounted) value $\text{EDisct}(\lambda)^M_*(s)$ is then defined as

$$\text{EDisct}(\lambda)^M_*(s) = \sup_{\sigma \in \Sigma_M} \text{EDisct}(\lambda)^M_\sigma(s).$$

We say that σ is discounted optimal if $\text{EDisct}(\lambda)^M_*(s) = \text{EDisct}(\lambda)^M_\sigma(s)$ for every $s \in S$.

F.2 PAC Learning for RMDPs with Discounted Rewards

Lemma 2. *Let M be an RMDP with diameter d, M' be an RMDP that differs from M only by using a different transition function $\delta_{M'}$ ε-close to δ_M, and let σ an ε-proper strategy. Then* $|\mathrm{EDisct}(\lambda)^M_\sigma(s) - \mathrm{EDisct}(\lambda)^{M'}_\sigma(s)| \leq \frac{\varepsilon d}{(1-\lambda)^2}$.

Proof. We can simply estimate

$$\begin{aligned} |\mathrm{EDisct}(\lambda)^M_\sigma(s) - \mathrm{EDisct}(\lambda)^{M'}_\sigma(s)| &\leq \textstyle\sum_{i=1}^\infty \lambda^{i-1} |\mathbb{E}^M_\sigma(s) r(X_{i-1}, Y_i) - \mathbb{E}^{M'}_\sigma(s) r(X_{i-1}, Y_i)| \\ &\leq \textstyle\sum_{i=1}^\infty d\lambda^{i-1}(1-(1-\varepsilon)^i) = \frac{d}{1-\lambda} - \frac{(1-\varepsilon)d}{1-\lambda+\lambda\varepsilon} \\ &= \frac{\varepsilon d}{(1-\lambda)(1-\lambda+\lambda\varepsilon)} \leq \frac{\varepsilon d}{(1-\lambda)^2}\ , \end{aligned}$$

where the first inequality is by triangulation, while the second is estimating the chance, that the difference between δ and δ' has been realized within the first i steps by $1-(1-\varepsilon)^i$. This bounds the sum of the different probabilities of taking a transition; the difference in the rewards is in this case bounded by the diameter d. □

As the estimation from the previous lemma survives a supremum operation over all ε-proper strategies we get the following corollary.

Corollary 1. *For an ε'-proper RMDP M with diameter d, and an M' that differs from M only by using a different transition function $\delta_{M'}$ ε-close to δ_M for $\varepsilon \leq \varepsilon'$, then* $|\mathrm{EDisct}(\lambda)^M(s) - \mathrm{EDisct}(\lambda)^{M'}(s)| \leq \frac{\varepsilon d}{(1-\lambda)^2}$.

Corollary 2. *Consequently, it suffices to learn the transition function δ_M of an ε'-proper RMDP*

- *with precision $\varepsilon'' = \min\left\{\varepsilon', \frac{\varepsilon(1-\lambda)^2}{d}\right\}$ to obtain (efficient) PAC learnability if we can (efficiently) evaluate the approximate RMDP M' obtained, or*
- *with precision $\varepsilon'' = \frac{1}{2}\min\left\{\varepsilon', \frac{\varepsilon(1-\lambda)^2}{d}\right\}$ if we can (efficiently) approximately evaluate M' with precision ε''*

with probability δ.

But this is a standard condition that does not differ much from MDPs and provides the following theorem.

Theorem 8. *Taking as input $\frac{1}{\varepsilon'}, \frac{1}{\varepsilon}, \frac{1}{\delta}, d, n$, we can construct with probability $\geq \delta$, for an ε'-proper RMDP M with n states and diameter d,* $\mathrm{EDisct}(\lambda)^M(s)$ *with precision ε. If the evaluation (approximation) of an RMDP M' that differs from M only in using $\delta_{M'}$ instead of δ_M with precision* $\min\left\{\varepsilon', \frac{\varepsilon(1-\lambda)^2}{d}\right\}$ *(*$\frac{1}{2}\min\left\{\varepsilon', \frac{\varepsilon(1-\lambda)^2}{d}\right\}$*) is tractable in the parameters above, then we can efficiently PAC learn* $\frac{1}{2}\min\left\{\varepsilon', \frac{\varepsilon(1-\lambda)^2}{d}\right\}$.

Naturally, the discounted value for a given RMDP can be approximated by unraveling sufficiently deeply.

Corollary 3. *For every ε'-proper RMDP,* $\mathrm{EDisct}(\lambda)^M(s)$ *is PAC-learnable.*

F.3 PAC Learning for RMDPs with Undiscounted Rewards

We now generalize this result to the undiscounted case we refer to in the paper. The first step is a reduction, which shows that the discounted case can be viewed as a special case of the undiscounted case by encoding the discounted payoffs by using an adjusted RMDP. We then continue to adjust the resulting RMDP further to add costs to "stopping" the RMDP in accordance with the level we are in.

This will then turn into a cost estimation, whereby the exit lane is entered to overestimate the effect of changing the strategy. For this to be estimated with a low cost, we use the knowledge that the expected number of steps till termination is finite.

F.3.1 From Discounting to Stopping

In MDPs, discounting can be simulated by stopping in each step with a chance of $1 - \lambda$, while continuing with a chance of λ. In RMDPs, there is no equivalent of stopping immediately, as we will first have to move down the stack. However, this can be simulated by adding a special exit to each component and block: from this exit of a block, we always continue with a reward of 0 to the special exit of the component, and thus have a cost-free direct way to termination, albeit it takes as many steps as the stack is high.

Once equipped with such an *exit lane* without rewards, we can use a reduction similar to the standard stopping reduction for MDPs: we simply go with probability $1-\lambda$ to the special exit of the component, and continue with a probability of λ, i.e., all former probabilities are multiplied by λ.

F.3.2 Adding Rewards to the Exit Lane

In order to connect the construction with the parameters from Section 2, we first consider the effect of changing the rewards on the exit lane.

If we change the rewards of all transitions that lead to the special exit from 0 to an upper bound $b \geq 0$ or a lower bound $-b \leq 0$ for the maximal or minimal expected reward for *any* strategy for *any* state with empty stack for *any* RMDP M' ε-close to M, then the collected reward when moving down the *exit lane* with rewards can serve as an over- or underrepresentation of the remaining reward for any strategy.

In turn the over- and underestimations for a stack of hight h are over- and underestimations for the remaining rewards after h steps.

F.3.3 Connecting our Parameters with Discounting

To see that the parameters we have provided generalize the case of discounting, note that, for an RMDP with discounting, one can choose $b = \frac{d}{1-\lambda}$, $c_o = 1 + \frac{1}{1-\lambda}$, and $\mu = 1 - \lambda$.

To justify the choice of $b = \frac{d}{1-\lambda}$, we observe that the expected sum of the weight-discounted number of steps is at most $\frac{1}{1-\lambda}$, such that the absolute value of the rewards collected is at most the diameter times this number.

To justify the choice for c_o and μ, we first define and estimate a different value for M^+ (and M^-): we define the value of a state as

1. the height of the stack plus $\frac{1}{1-\lambda}$ if the RMDP is still running and we are not on the exit lane,
2. the height of the stack if we are on the exit lane, and
3. 0 if the RMDP has terminated.

In the first case, the expected value of the state will drop by at least $1 - \lambda$: this is because it can only increase by 1 when the exit lane is not entered (which happens with a likelihood of λ), and goes down by $1 + \frac{1}{1-\lambda}$ if the exit lane is entered (with a likelihood of $1 - \lambda$), which leads to an expected value that is at least $2(1 - \lambda)$ smaller.

In the second case, the value is always reduced by 1.

Thus, whenever the RMDP has not terminated after k steps, the expected value of this sum is reduced by (at least) $\min\{1, 2(1 - \lambda)\}$, and therefore by at least $1 - \lambda$.

Consequently, the expected value of this sum is bounded by $c_o - \mu \cdot \sum_{i=1}^{k} p_{\mathsf{run}}^{M'_\sigma}(k)$, and it in turn bounds the stack height.

F.3.4 Wrapping up the Proof

Theorem 2. *For every ε-proper RMDP with parameters c_o, μ, and b,* $\mathrm{ETotal}^{\mathcal{M}}(s)$ *is PAC-learnable.*

To wrap this up to show Theorem 2, we can estimate the effect of moving to a different position due to a change of a strategy by moving to the exit lane with either reward b (for overestimation) or $-b$ (for underestimation).

Thus, if we refer to an (arbitrary but fixed) RMDP M' ε-close to M, and to the over- and under approximation of with exit lanes with rewards b and $-b$ and a likelihood to transition to the exit lane of ε as M^+ and M^-, respectively, we have

$$\text{ETotal}_\sigma^{M^+}\left((\langle\emptyset\rangle, q)\right) \geq \text{ETotal}_\sigma^{M'}\left((\langle\emptyset\rangle, q)\right) \geq \text{ETotal}_\sigma^{M^-}\left((\langle\emptyset\rangle, q)\right)$$

for all strategies σ, and thus

$$\text{ETotal}^{M^+}\left((\langle\emptyset\rangle, q)\right) \geq \text{ETotal}^{M'}\left((\langle\emptyset\rangle, q)\right) \geq \text{ETotal}^{M^-}\left((\langle\emptyset\rangle, q)\right) .$$

In order to estimate $\text{ETotal}_\sigma^{M^+}(s) - \text{ETotal}_\sigma^{M^-}(s)$, we first observe that the expected stack height must be converging to 0 for M_σ by our assumption that it is bounded by $c_o - \mu \sum_{i-1}^{k} p_{\mathsf{run}}^{M_\sigma}(k)$ by our assumption. (Note that, for convergence, it is enough to assume termination in expected finite time. The stronger assumption is used for to establish PAC learnability.)

For a given $\varepsilon' > 0$ we therefore pick a k such that, after $k' \geq k$ steps, the expected stack height is lower than $\frac{\varepsilon'}{b}$. E.g., we can choose k such that $k\mu\frac{\varepsilon'}{b} \geq c_o$ holds, such that the falling sequence $p_{\mathsf{run}}^{M_\sigma}(k)$ cannot be above $\frac{\varepsilon'}{b}$ for k steps. This is satisfied for $k = \left\lceil \frac{c_o b}{\mu\varepsilon'} \right\rceil$.

We then pick $\varepsilon = \frac{\varepsilon'}{k^2 b}$.

In this case, the likelihood of moving onto the exit lane before step k is lower than $\frac{\varepsilon'}{kc}$, and the difference in rewards along the exit lane in this case below kc, leaving a contribution to the difference of below ε'.

When entering the exit lane on or after step k, the expected stack height is below $\frac{\varepsilon'}{b}$, leading to a contribution to the expected distance between the rewards for M^+ and M^- below ε'.

Together, this provides

$$\text{ETotal}_\sigma^{M^+}\left((\langle\emptyset\rangle, q)\right) - \text{ETotal}_\sigma^{M^-}\left((\langle\emptyset\rangle, q)\right) \leq 2\varepsilon' .$$

This shows that the expected value converges. We still need to show that we can approximate it to up to $4\varepsilon'$ with a likelihood of at least $1 - \delta$.

For this, we can estimate $\text{ETotal}_\sigma^{M^+}\left((\langle\emptyset\rangle, q)\right)$ and $\text{ETotal}_\sigma^{M^-}\left((\langle\emptyset\rangle, q)\right)$ by cutting the runs after step k, then the error in the expected value is at most ε'.

Estimating $\text{ETotal}_\sigma^{M}\left((\langle\emptyset\rangle, q)\right)$ by cutting the runs after k steps leads to a value in between. Using triangulation, this entails that it estimates the value of $\text{ETotal}_\sigma^{M}\left((\langle\emptyset\rangle, q)\right)$.

But this can be unravelled into a finite MDP, and hence be PAC learned up to precision ε' with likelihood at least $1 - \delta$ with standard techniques.

G Reinforcement Learning for Single-Exit RMDPs

Theorem 5 (Unique Fixed Point)**.** *The vector consisting of* $\text{ETotal}^M(q)$ *values is the unique fixed point of* F*. Moreover, a solution of these equations provide optimal stackless strategies.*

Proof of Theorem 5. The proof proceeds over several lemmas.

Lemma 3. $\text{ETotal}^M(q)$ *is a fixed point of* F*.*

Proof. Let us consider any vertex v.

If v is an entry port, i.e., $q = (b, en) \in \text{Call}, ex = Ex_{Y(b)}$ then once the box b is entered, by Assumption 1, we will be almost surely reach the unique exit ex' of b no matter which action will be picked inside of b. Therefore, to maximize the total reward it suffices to focus solely first on maximizing the reward while inside b before ex' is reached no matter what the current call stack

is. The maximal reward possible to obtain is equal to $\text{ETotal}^M(en)$ by definition. Once the exit is reached, it suffices to maximize the reward from the exit port (b, ex'), which is at most equal to $\text{ETotal}^M((b, ex'))$. It is easy to see that these two strategies can be combined into a single strategy from q that can get arbitrarily close to the maximum possible value of $\text{ETotal}^M(en) + \text{ETotal}^M((b, ex'))$. So this all together shows that $\text{ETotal}^M(q) = \text{ETotal}^M(en) + \text{ETotal}^M((b, ex'))$ holds.

If v is any other non-exit vertex, then the best we can do is to pick any action available at v that maximizes expected reward from the successor vertex plus the one step reward for picking this action at v. Once the successor node is reached, we can then switch to the best possible strategy from that node. This shows that the value of $\max_{a \in A(q)} \left\{ r(q,a) + \sum_{q' \in Q} p(q'|q,a)x(q') \right\}$ is not only achievable but also the most one can expect to get when starting at v. □

Lemma 4. *For any fixed point $\bar{x}$ of F, there exists a stackless strategy $\sigma(\bar{x})$ such that $\text{ETotal}^M_{\sigma(\bar{x})}(q) = \bar{x}(q)$ for every q.*

Proof. This strategy $\sigma(x)$ will simply pick

$$\sigma(\bar{x})(\langle \kappa \rangle, q) = \arg\max_{a \in A(q)} \left\{ r(q,a) + \sum_{q' \in Q} p(q'|q,a)x(q') \right\}$$

from any vertex q and the call stack κ. This works, because as argued in Theorem 3 a mapping of $(\langle \kappa\kappa' \rangle, q)$ to $(\langle \kappa' \rangle, q)$ for all κ' and q gives us two isomorphic models until the exit of the top box in κ is reached and so the optimal strategy for $(\langle \emptyset \rangle, q)$ works for $(\langle \kappa \rangle, q)$ as well. In other words, the best thing to do at $(\langle \kappa \rangle, q)$ is to try to maximize the reward before exiting the current box which happens with probability 1 and the same holds for $(\langle \emptyset \rangle, q)$. In order to maximize this reward, the call stack being κ or $\emptyset$ makes no difference as the transitions only depend on the current component. □

Now, we can observe that $\bar{x}^*(q) = \text{ETotal}^M(q)$ is the largest fixed point of F. If there was $\bar{x}$ such that $\bar{x}^*(q) < \bar{x}(q)$ for some q then we could pick $\sigma(\bar{x})$ as our strategy and get the reward of $\bar{x}(q)$ from q due to Lemma 4. However, just by definition, $\bar{x}^*(q) = \text{ETotal}^M(q) = \sup_\sigma \text{ETotal}^M(q)$ is the largest possible reward obtainable from q as all strategies σ are considered; a contradiction.

Note that once a stackless strategy σ is fixed then $F(\bar{x})$ becomes equal to $A_\sigma \bar{x} + \bar{b}_\sigma$, where $\bar{b}_\sigma$ is the one-step transitions rewards obtained using σ and A_σ is a transition matrix derived from σ. Note that all entries of A_σ are non-negative. We will refer to F as F_σ in such a case and exploit this fact later in various proofs. Note that we can interpret $F_\sigma(\bar{x})$ as the expected total reward of using σ for each vertex once and then obtaining reward $\bar{x}$.

Lemma 5. *For any proper 1-exit RMDP and stackless strategy σ, we have that $\text{ETotal}^M_\sigma = \sum_{i=1}^{\infty} A^i_\sigma b_\sigma$.*

Proof. Notice that

$$\begin{aligned}
F_\sigma(\bar{x}) &= A_\sigma \bar{x} + b_\sigma \\
F^2_\sigma(\bar{x}) &= A_\sigma(A_\sigma \bar{x} + b_\sigma) + b_\sigma = A^2_\sigma \bar{x} + (A_\sigma + I) b_\sigma \\
F^3_\sigma(\bar{x}) &= A_\sigma F^2_\sigma(\bar{x}) + b_\sigma = A^3_\sigma \bar{x} + (A^2_\sigma + A_\sigma + I) b_\sigma \\
&\vdots \\
F^k_\sigma(\bar{x}) &= A^k_\sigma \bar{x} + \Big(\sum_{i=0}^{k} A^i_\sigma \Big) b_\sigma .
\end{aligned}$$

Let $\mathbf{1}$ and $\mathbf{0}$ be vectors consisting only of 1s and 0s, respectively. As in [15], we can show that the expected number of steps taken before termination while using σ can be computed by iterating $\bar{y}_{i+1} = A_\sigma(\bar{y}_i) + \mathbf{1}$ when starting at $\bar{y}_0 = \mathbf{0}$. Intuitively, $\bar{y}_{i+1}$ will correspond to the expected number of steps taken before termination when we assume the call and return from a box to be executed in

parallel in a single step. The crucial assumption made in [15] that all rewards are positive (and equal to 1) in true in this case.

This shows that $\lim_{k\to\infty} F^k_\sigma(\mathbf{0}) = A^k_\sigma \mathbf{0} + \left(\sum_{i=0}^k A^i_\sigma\right)\mathbf{1}$ is finite. Therefore, we have that $\lim_{i\to\infty} A^i_\sigma$ is an 0 matrix, because otherwise that sum $\sum_{i=0}^k A^i_\sigma$ would not be finite as a sum of all non-negative matrices.

Now, one can see that $\lim_{k\to\infty} F^k_\sigma(\bar{x}) = \lim_{k\to\infty} A^k_\sigma \bar{x} + \left(\sum_{i=0}^k A^i_\sigma\right) b_\sigma = \lim_{k\to\infty}\left(\sum_{i=0}^k A^i_\sigma\right) b_\sigma$ converges absolutely, because $\left(\sum_{i=0}^k A^i_\sigma\right)|b_\sigma| \leq \left(\sum_{i=0}^k A^i_\sigma\right) r_{max} \leq K r_{max}$, where K is the expected number of steps taken by any strategy as guaranteed by Assumption 1. It is clear that this limit $\lim_{k\to\infty} F^k_\sigma(\bar{x})$ is a fixed point of F_σ, because $F_\sigma(\lim_{k\to\infty} F^k_\sigma(\bar{x})) = \lim_{k\to\infty} F^{k+1}_\sigma(\bar{x}) = \lim_{k\to\infty} F^k_\sigma(\bar{x})$. As we already showed ETotal^M_σ is a fixed point of F_σ. It is clear that F_σ cannot have any other fixed point because, for arbitrary $\bar{x}$, we have $\lim_{k\to\infty} F^k_\sigma(\bar{x}) = A^i_\sigma b_\sigma$ which is fixed in terms of $\bar{x}$. This shows that $\text{ETotal}^M_\sigma = \sum_{i=1}^\infty A^i_\sigma b_\sigma$ has to hold. □

We are ready to show that no other than $\bar{x}^* = \text{ETotal}^M$ fixed point of F exists. Suppose there is one and let us denote it by $\bar{x}$. As we just showed $\bar{x} \leq \bar{x}^*$ and $\bar{x}(q) < \bar{x}^*(q)$ for some q (because otherwise $\bar{x}^* = \bar{x}$).

Let us denote $\sigma(\bar{x}^*)$ by $\sigma*$ and consider $F_{\sigma^*}(\bar{x})$. If any coordinates of $F_{\sigma^*}(\bar{x})$ is larger than $\bar{x}$ then we can improve the expected total reward by using σ^* once and then follow the strategy that gives us expected total reward of $\bar{x}$. A contradiction with the assumption that $\bar{x}$ is a fixed point of F.

So we have $F_{\sigma^*}(\bar{x}) \leq \bar{x}$ and by iterating this we get that $F^k_{\sigma^*}(\bar{x}) \leq \bar{x}$ for every k. In the previous Lemma we showed that $\text{ETotal}^M = \lim_{k\to\infty} F^k_{\sigma^*}(\bar{x}) \leq \bar{x} < \text{ETotal}^M$; a contraction. □

Theorem 7 (Efficient PAC Learning for 1-Exit RMDPs)**.** *For every ϵ-proper 1-exit RMDP with diameter r_{max} and the expected time to terminate $\leq K$, $\text{ETotal}^{\mathcal{M}}(s)$ is efficiently PAC-learnable.*

Proof. Let us denote the ϵ-proper 1-exit RMDP by M and its optimal total reward stackless strategy by σ^*. As we know that every strategy is ϵ-proper then so is σ^*. This means that for all M' that that are ε-close to $\mathcal{M}$ the expected time to terminate when using σ^* is also $\leq K$. Recall that ε-closeness means that $\sum_{q\in S, a\in A, r\in S} |\delta_{\mathcal{M}}(q,a)(r) - \delta_{\mathcal{M}'}(q,a)(r)| \leq \varepsilon$, and where the support of $\delta_{M'}(q,a)$ is a subset of the support of $\delta_M(q,a)$ for all $q \in S$ and $a \in A$.

Lemma 6. *If stackless σ is ε-proper, then $\|\text{ETotal}^M_\sigma - \text{ETotal}^{M'}_\sigma\|_\infty \leq 2\epsilon K^2 r_{\max}$.*

Proof. As shown in Lemma 5, the optimal total rewards in M satisfy $\bar{x} = A_\sigma \bar{x} + b_\sigma$ and in M' statisfy $\bar{x}' = A'_\sigma \bar{x}' + b_\sigma$. Moreover, $\bar{x}$ and $\bar{x}'$ are the sole fixed points of these equations. To avoid clutter, we will write A, A' and $\bar{b}$ instead of A_σ, A'_σ and $\bar{b}_\sigma$. We know that $\|A - A'\|_1 \leq 2\varepsilon$ and let $\mathcal{E} = A' - A$.

When we try to converge at $\bar{x}'$ by iterating $\bar{x}'_{i+1} = A'\bar{x}'_i + b'$ when starting at $\bar{x}'_0 = \bar{x}$, we obtain the following:

$$\begin{aligned}
\bar{x}'_1 &= A'\bar{x}_0 + \bar{b} = (A' - A)\bar{x} + A\bar{x} + \bar{b} = (A' - A)\bar{x} + \bar{x} = \mathcal{E}\bar{x} + \bar{x}_0 \\
\bar{x}'_2 &= A'\bar{x}_1 + \bar{b} = A'(\mathcal{E}\bar{x} + \bar{x}_0) + \bar{b} = A'\mathcal{E}\bar{x} + A'\bar{x}_0 + \bar{b} = A'\mathcal{E}\bar{x} + \bar{x}_1 \\
\bar{x}'_3 &= A'\bar{x}_2 + \bar{b} = A'(A'\mathcal{E}\bar{x} + \bar{x}_1) + \bar{b} = (A')^2\mathcal{E}\bar{x} + A'\bar{x}_1 + \bar{b} = (A')^2\mathcal{E}\bar{x} + \bar{x}_2 \\
&\vdots \\
\bar{x}'_{k+1} &= (A')^k\mathcal{E}\bar{x} + \bar{x}_k
\end{aligned}$$

and thus $\bar{x}' - \bar{x} = (\lim_{k\to\infty} \bar{x}'_{k+1}) - \bar{x}_0 = (\lim_{k\to\infty} \sum_{i=0}^k A'^i \mathcal{E}\bar{x} + \bar{x}_0) - \bar{x}_0 = (\sum_{i=0}^\infty A'^i)\mathcal{E}\bar{x}$.

Note that the time to terminate in M and M' from each vertex is equal to $(\sum_{i=0}^\infty A^i)\mathbf{1}$ and $(\sum_{i=0}^\infty A'^i)\mathbf{1}$, respectively. We know that all of these values are $\leq K$. We know that the absolute value of each entry of $\bar{x}$ is at most Kr_{max}, as the most we can get in each step is r_{max}

during the expected number of $\leq K$ steps. So $\|\mathcal{E}\bar{x}\|_\infty \leq 2\varepsilon K r_{max}\mathbf{1}$, and so $\|(\sum_{i=0}^{\infty} {A'}^i)\mathcal{E}\bar{x}\|_\infty \leq \|2\varepsilon K r_{max}(\sum_{i=0}^{\infty} {A'}^i)\mathbf{1}\|_\infty \leq 2\varepsilon K^2 r_{max}$ as required. □

We observe that the estimate in Lemma 6 holds when we take the supremum over all stackless ε-proper strategies. This is because for any two functions f and g such that $f(\sigma) - g(\sigma) \leq \varepsilon$ for every stackless σ and $\sigma' = \arg\max_\sigma f(\sigma)$ we get $\sup_\sigma f(\sigma) - \sup_\sigma g(\sigma) \leq f(\sigma') - g(\sigma') \leq \varepsilon$. In other words, Lemma 6 implies that $\|\text{ETotal}^M - \text{ETotal}^{M'}\|_\infty \leq 2\epsilon K^2 r_{\max}$.

We can now choose $\varepsilon' = \frac{\varepsilon}{2\epsilon K^2 r_{\max}}$, and learn δ_M up to precision ε' with probability $\geq \delta$ by the usual sampling techniques as done for finite MDPs.

We then obtain 1-exit RMDP M', which we can solve efficiently and exactly using linear programming similarly to what was done in [15]. First, let us create a variable t_q for every $q \in Q$. The linear program is then as follows.

$$\text{Minimize} \sum_q t_q \text{ subject to:}$$

$$t_q \geq t_{en} + t_{(b,ex')} \qquad \text{if } q = (b, en) \in \text{Call}, ex = Ex_{Y(b)}$$

$$t_q \geq r(q,a) + \sum_{q' \in Q} p(q'|q,a) t_{q'} \qquad \text{otherwise, for every possible } a \in A$$

One can easily show that any optimal solution gives us a fixed point of F which we know has to be equal to $\text{ETotal}^{M'}$. This is because if all inequalities for a variable t_q are non-strict then we can decrease the value of t_q to the maximum of their right hand sides and still get a valid solution. As linear programs can be solved in polynomial time, and $\text{ETotal}^{M'}$ is ϵ-close to ETotal^M, we obtain an efficient PAC learning algorithm. □