A New Sample-Efficient PAC Reinforcement Learning Algorithm

Ashkan Zehfroosh and Herbert G. Tanner

Abstract— **This paper introduces a new hybrid PAC RL algorithm for MDPs, which intelligently maintains favorable features of its parents. The DDQ algorithm, integrates model-free and model-based learning approaches, preserving some advantages from both. A PAC analysis of the DDQ algorithm is presented and its sample complexity is explicitly bounded. Numerical results from a small-scale example motivated by work on human-robot interaction models corroborates the theoretical predictions on sample complexity.**

I. Introduction

A probabilistic, discrete-time and space dynamical system can be mathematically modelled as a Markov decision process (MDP), and controllers for this type of systems can be designed by reinforcement learning (RL) algorithms. Reinforcement learning is a procedure to obtain an optimal policy in an MDP, when the actual transition probabilities and/or reward function are not known. The procedure involves exploration of the MDP model. Out of the available RL algorithms, few are probably approximately correct (PAC)—namely, they can guarantee how soon the algorithm will converge to the near-optimal policy. Existing PAC RL algorithms for MDPs can be classified into two categories: model-based [1]–[4], and model-free [5]. This paper reports on an effort to capture the "best of both worlds" in a single PAC algorithm.

The motivation for this new algorithm comes from application problems in the area of early pediatric motor rehabilitation, where robots can be used as smart toys to interact with infants who have special needs, and engage with them in play-based activity that involves gross motion. There, MDP models capture the dynamics of social interaction between infant and robot [6]. Similar models have been utilized in other human-robot interaction (HRI) applications [7]–[9].

The idea is that RL algorithms can guide the behavior of the robot as it interacts with the infant in order to achieve the maximum possible rehabilitation outcome—the latter possibly quantified by the overall length of infant displacement, or the frequency of infant motor transitions. Some early attempts at modeling such instances of HRI did not result in models of particularly large state and action spaces, but were particularly complicated by the absence of sufficient data sets for learning [6], [10], [11]. There is a need for RL approaches that maintain efficiency and accuracy even when the learning set is small.

Model-free RL is thought of requiring relatively large bodies of training data—what is more formally referred to as a PAC algorithm's *sample complexity*. This is mainly because model-free RL tends to ignore information from state transitions, relying exclusively on observed rewards [12]. As a result of the large training set requirement, the applicability of model-free RL in some real-world learning scenarios may be limited. In contrast, model-based RL learns a model of the interaction between the system and its environment first, using all information from state transitions, and then computes an optimal policy based on that model. Because of that, the sample complexity of model-based RL algorithms is typically lower than comparable model-free algorithms [13]. The price is computational effort and possible bias [12].

A popular model-free PAC RL algorithm applicable on MDPs is *Delayed Q-learning* [5], enjoying an upper-bound on sample complexity of that is smaller than model-based counterparts in cases of MDPs with large state-spaces [14]—the size of the state space is taxing when it comes to model-based algorithms building a model. One such model-based PAC algorithm is R-max [2]. The upper-bound that is known for its sample complexity [15] confirms the algorithm's superiority in terms of training data needs (sample efficiency) when state and action spaces are relatively small.

In general, R-max and Delayed Q-learning are incomparable in terms of sample efficiency, because there can be additional considerations that may skew preferences depending on the application. For a given sample size, R-max is likely to return more accurate policies than those of Delayed Q-learning, but the latter will converge much faster on problems with large state spaces. While model-free algorithms circumvent the model learning stage of the solution process, reducing complexity in problems of large size as a result, the process of learning a model is often not the key limitation. Some neurophysiologically-inspired hypotheses [16] promote the view that the brain learns complex tasks by a *combination* of approaches, which one could classify as either model-free (trial and error) or model-based (deliberate planning and computation), depending on the amount and reliability of the available information. Motivated by these hypotheses, this paper introduces a hybrid PAC MDP RL design, that concurrently blends model-free processes (borrowed from Delayed Q-learning) with model-based processes (taken from R-max). This new algorithm is referred to as Dyna-Delayed Q-learning (DDQ).

The results reported in this paper are along an established line of research that seeks connections between model-free and model-based RL algorithms, and which has begun with the the Dyna-Q algorithm [17]. This algorithm utilized synthetic experiences produced by a learned model in order to expedite Q-learning. Along this track of research one can

Ashkan Zehfroosh, and Bert Tanner are with the Department of Mechanical Engineering, University of Delaware. {ashkanz, btanner}@udel.edu

This work was supported by NIH under grant # R01HD87133.

also find partial model back propagation [18], approaches that build a goal condition Q function [19]–[22], and more recent methods that integrate model-based linear quadratic regulator schemes into model-free algorithms for path integral policy improvement [23]. Along the later research thread, in particular, a recent approach that utilizes Temporal Difference Models (TDM) [12] attempts a smooth(er) transition from model-free to model-based.

This paper leverages the classic Dyna-Q concept to blend the processes of a model-free and a model-based PAC algorithm, yielding a new one which is not only PAC like its predecessors, but can outperform them both in terms of sample complexity. This new algorithm is called DDQ, and is designed to follow a model-free approach when attacking large problems, and a model-based approach when faced with problems that require high accuracy. In fact, the sample complexity of DDQ, in the worst case equal to the minimum between the bounds of R-max and Delayed Q-learning, and often lower than both.

II. Preliminaries

A finite MDP M is a tuple $\{S, A, R, T, \gamma\}$ with elements

S	a finite set of states
A	a finite set of actions
$R : S \times A \to [0,1]$	the *reward* from executing a at s
$T : S \times A \times S \to [0,1]$	the *transition probabilities*
$\gamma \in [0,1)$	the *discount factor*

A *policy* π is a mapping $\pi : S \to A$ that selects an action a to be executed at state s. By $|S|$ we denote the cardinality of a set S. A policy is *optimal* if it maximizes the expected sum of discounted rewards; if t indexes the current time step and a_t, s_t denote current action and state, respectively, then this expected sum is written $\mathbb{E}_M\{\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)\}$. The discount factor γ here reflects the preference of immediate rewards over future ones. The *value* of state s under policy π in MDP M is defined as $v_M^{\pi}(s) = \mathbb{E}_M\left\{R\big(s,\pi(s)\big) + \sum_{t=1}^{\infty}\gamma^t R\big(s_t,\pi(s_t)\big)\right\}$. Note that an upper bound for the value at any state is $v_{\max} = \frac{1}{1-\gamma}$. Similarly defined is the value of *state-action pair* (s,a) under policy π: $Q_M^{\pi}(s,a) = \mathbb{E}_M\left\{R(s,a) + \sum_{t=1}^{\infty}\gamma^t R\big(s_t,\pi(s_t)\big)\right\}$. Every MDP M has at least one optimal policy π^* that results in an optimal value assignment at all states; the latter is denoted $v_M^*(s)$ (or $Q_M^*(s,a)$, respectively). Optimal values result from a search for fixed points of the Bellman equation $v_M^*(s) = \max_a\{R(s,a) + \gamma\sum_{s'}T(s,s',a)V_M^*(s')\}$ where, after substituting $v_M^*(s') = \max_a Q_M^*(s',a)$, can equivalently be written in terms of state-action values $Q_M^*(s,a) = R(s,a) + \gamma\sum_{s'}T(s,s',a)v_M^*(s')$.

An RL algorithm usually maintains a table of state-action pair value estimates $Q(s,a)$ that are updated based on the exploration data. Denote $Q_t(s,a)$ the currently stored value for state-action pair (s,a) at timestep t of an RL algorithm; consequently, $v_t(s) = \max_a Q_t(s,a)$. An RL algorithm is *greedy* if at any timestep t, it always executes action $a_t = \arg\max_{a\in A} Q_t(s_t,a)$. The policy in force at t is denoted π_t.

An RL algorithm is expected to converge to the optimal policy, practically reporting a near-optimal one at termination. Probably approximately correct (PAC) analysis of RL algorithms deals with the question of how fast an RL algorithm converges to a near-optimal policy. An RL algorithm is PAC if there exists a probabilistic bound on the number of exploration steps that the algorithm can take before converging to a near-optimal policy.

Definition 1: Consider that an RL algorithm $\mathcal{A}$ is executing on MDP M. Let s_t be the visited state at time step t and $\mathcal{A}_t$ denotes the (non-stationary) policy that the $\mathcal{A}$ executes at t. For a given $\epsilon > 0$ and $\delta > 0$, $\mathcal{A}$ is a PAC RL *algorithm* if there is an $N > 0$ such that with probability at least $1-\delta$ and for all but N time steps,

$$v_M^{\mathcal{A}_t}(s_t) \geq v_M^*(s_t) - \epsilon \ . \tag{1}$$

Equation (1) is known as the ϵ-optimality condition and N as the *sample complexity* of $\mathcal{A}$, which is a function of $\big(|S|, |A|, \frac{1}{\epsilon}, \frac{1}{\delta}, \frac{1}{1-\gamma}\big)$.

Definition 2: Consider MDP $M = \{S, A, R, T, \gamma\}$ which at time t has a set of state-action value estimates $Q_t(s,a)$, and let $K_t \subseteq S \times A$ be a set of state-action pairs labeled *known*. The *known state-action* MDP, $M_{K_t} = \big\{S \cup \{z_{s,a} | (s,a) \notin K_t\}, A, T_{K_t}, R_{K_t}, \gamma\big\}$, is an MDP derived from M and K_t by defining new states $z_{s,a}$ for each unknown state-action pair $(s,a) \notin K_t$, with self-loops for all actions, i.e., $T_{K_t}(z_{s,a},\cdot,z_{s,a}) = 1$. For all $(s,a) \in K_t$, it is $R_{K_t}(s,a) = R(s,a)$ and $T_{K_t}(s,a,\cdot) = T(s,a,\cdot)$. When an unknown state-action pair $(s,a) \notin K_t$ is experienced, $R_{K_t}(s,a) = Q_t(s,a)(1-\gamma)$ and the model jumps to $z_{s,a}$ with $T_{K_t}(s,a,z_{s,a}) = 1$, and $R_{K_t}(z_{s,a},\cdot) = Q_t(s,a)(1-\gamma)$.

Let K_t be set of current known state-action pairs of an RL algorithm $\mathcal{A}$ at time t. Set K_t can be arbitrarily constructed as long as it depends only on the history of exploration data up to t. Any $(s,a) \notin K_t$ experienced at t is an *escape event*.

Theorem 1 ([14]): Let $\mathcal{A}$ be a greedy RL algorithm for an arbitrary MDP M, and let K_t be the set of current known state-action pairs, defined based only on the history of the exploration data up to timestep t. Assume that $K_t = K_{t+1}$ unless an update to some state-action value occurs or an escape event occurs at timestep t, and that $Q_t(s,a) \leq v_{\max}$ for all (s,a) and t. Let M_{K_t} be the known state-action MDP at timestep t and $\pi_t(s) = \arg\max_a Q_t(s,a)$ denote the greedy policy that $\mathcal{A}$ executes. Suppose now that for any positive constant ϵ and δ, the following conditions hold with probability at least $1-\delta$ for all s, a and t:

optimism: $v_t(s) \geq v_M^*(s) - \epsilon$

accuracy: $v_t(s) - v_{M_{K_t}}^{\pi_t}(s) \leq \epsilon$

complexity: sum of number of timesteps with Q-value updates plus number of timesteps with escape events is bounded by $\zeta(\epsilon,\delta) > 0$.

Then, executing algorithm $\mathcal{A}$ on any MDP M will result in following a 4ϵ-optimal policy on all but

$$\mathcal{O}\left(\frac{\zeta(\epsilon,\delta)}{\epsilon(1-\gamma)^2}\ln\big(\tfrac{1}{\delta}\big)\ln\big(\tfrac{1}{\epsilon(1-\gamma)}\big)\right) \simeq \mathcal{O}\left(\frac{\zeta(\epsilon,\delta)}{\epsilon(1-\gamma)^2}\right) \tag{2}$$

timesteps, with probability at least $1-2\delta$.

III. DDQ Algorithm

This section presents Algorithm 1, the one we call DDQ and the main contribution of this paper. DDQ integrates elements of R-max and Delayed Q-learning, while preserving the implementation advantages of both. We refer to the assignment in line 30 of Algorithm 1 as a *type*-1 *update*, and to the one on line 51 as a *type*-2 *update*. Type-1 updates use the m_1 most recent experiences (occurances) of a state-action pair (s, a) to update that pair's value, while a type-2 update is realized through a value iteration algorithm (lines $42-53$) and applies to state-action pairs experienced at least m_2 times. The outcome at timestep t of the value iteration for a type-2 update is denoted $Q_t^{\mathrm{vl}}(s, a)$. The value iteration is set to run for $\frac{\ln(1/(\epsilon_2(1-\gamma)))}{(1-\gamma)}$ iterations; parameter ϵ_2 regulates the desired accuracy on the resulting estimate (Lemma 3). A type-1 update is successful only if the condition on line 29 of the algorithm is satisfied, and this condition ensures that the type-1 update necessarily decreases the value estimate by at least $\epsilon_1 = 3\epsilon_2$. Similarly, a type-2 update is successful only if the condition on line 50 of the algorithm holds. The DDQ algorithm maintains the following internal variables:

- $l(s, a)$: the number of samples gathered for the update type-1 of $Q(s, a)$ once $l(s, a) = m_1$.
- $U(s, a)$: the running sum of target values used for a type-1 update of $Q(s, a)$, once enough samples have been gathered.
- $b(s, a)$: the timestep at which the most recent or ongoing collection of m_1 (s, a) experiences has started.
- $\mathsf{learn}(s, a)$: a Boolean flag that indicates whether or not samples are being gathered for type-1 update of $Q(s, a)$. The flag is set to true initially, and is reset to true whenever some Q-value is updated. It flips to false when no updates to any Q-values occurs within a time window of m_1 experiences of (s, a) in which attempted updates type-1 of $Q^i(s, a)$ fail.
- $n(s, a)$: variable that keeps track of the number of times (s, a) is experienced.
- $n(s, a, s')$: variable that keeps track of the number of transitions to s' on action a at state s.
- $r(s, a)$: the accumulated rewards by doing a in s.

The execution of the DDQ algorithm is tuned via the m_1 and m_2 parameters. One can practically reduce it to Delayed Q-learning by setting m_2 very large, and to R-max by setting m_1 large. The next section provides a proof sketch (due to lack of space) that DDQ is not only PAC but also *possesses the minimum sample complexity* between R-max and Delayed Q-learning in the worst case —often, it outperforms both. A complete formal proof of the properties will be provided in a separate paper.

IV. PAC Analysis of DDQ Algorithm

It has been noted that the sample complexity bounds known for R-max and Delayed Q-learning are incomparable directly [14]. The sample complexity of R-max algorithm is $\frac{|S|^2|A|}{\epsilon^3(1-\gamma)^8}$ —note the power on ϵ; the sample complexity of Delayed Q-learning algorithm is $\frac{|S||A|}{\epsilon^4(1-\gamma)^8}$ —note the

Algorithm 1 The DDQ algorithm

```
1: Inputs: $S, A, \gamma, m_1, m_2, \epsilon_1, \epsilon_2$
2: for all $s, a, s'$ do
3:     $Q(s,a) \leftarrow v_{\max}$        ▷ initialize $Q$ values to its maximum
4:     $U(s,a) \leftarrow 0$        ▷ used for attempted updates of type-1
5:     $l(s,a) \leftarrow 0$        ▷ counters
6:     $b(s,a) \leftarrow 0$  ▷ beginning timestep of attempted update type-1
7:     $\mathsf{learn}(s,a) \leftarrow \mathrm{true}$        ▷ learn flags
8:     $n(s,a) \leftarrow 0$        ▷ number of times $(s,a)$ is tried
9:     $n(s,a,s') \leftarrow 0$     ▷ number of transitions to $s'$ by $a$ in $s$
10:    $r(s,a) \leftarrow 0$    ▷ accumulated reward by execution of $a$ in $s$
11: end for
12: $t^* \leftarrow 0$        ▷ time of the most recent successful timestep
13: for $t = 1, 2, 3, \ldots$ do
14:     let $s$ denotes the state at time $t$
15:     choose action $a = \arg\max_{a' \in A} Q(s, a')$
16:     observe immediate reward $r$ and next state $s'$
17:     $n(s,a) = n(s,a) + 1$ and $n(s,a,s') = n(s,a,s') + 1$
18:     $r(s,a) = r(s,a) + r$
19:     if $b(s,a) \leq t^*$ then
20:         $\mathsf{learn}(s,a) \leftarrow \mathrm{true}$
21:     end if
22:     if $\mathsf{learn}(s,a) = \mathrm{true}$ then
23:         if $l(s,a) = 0$ then
24:             $b(s,a) \leftarrow t$
25:         end if
26:         $l(s,a) \leftarrow l(s,a) + 1$
27:         $U(s,a) \leftarrow U(s,a) + r + \gamma \max_{a'} Q(s', a')$
28:         if $l(s,a) = m_1$ then
29:             if $Q(s,a) - U(s,a)/m_1 \geq 2\epsilon_1$ then
30:                 $Q(s,a) \leftarrow U(s,a)/m_1 + \epsilon_1$ and $t^* \leftarrow t$
31:             else if $b(s,a) > t^*$ then
32:                 $\mathsf{learn}(s,a) \leftarrow \mathrm{false}$
33:             end if
34:             $U(s,a) \leftarrow 0$ and $l(s,a) \leftarrow 0$
35:         end if
36:     end if
37:     if $n(s,a) = m_2$ or $t = t^*$ then
38:         $t^* \leftarrow t$
39:         for all $(\bar{s}, \bar{a})$ do
40:             $Q_{\mathrm{vl}}(\bar{s}, \bar{a}) \leftarrow Q(\bar{s}, \bar{a})$
41:         end for
42:         for $i = 1, 2, 3, \ldots, (\frac{\ln(1/(\epsilon_2(1-\gamma)))}{(1-\gamma)})$ do
43:             for all $(\bar{s}, \bar{a})$ do
44:                 if $n(\bar{s}, \bar{a}) \geq m_2$ then
45:                     $Q_{\mathrm{vl}}(\bar{s}, \bar{a}) \leftarrow \frac{r(\bar{s},\bar{a})}{n(\bar{s},\bar{a})} +$
                            $\gamma \sum_{s''} \frac{n(\bar{s},\bar{a},s'')}{n(\bar{s},\bar{a})} \max_{a'} Q_{\mathrm{vl}}(s'', a')$
46:                 end if
47:             end for
48:         end for
49:         for all $(\bar{s}, \bar{a})$ do
50:             if $Q_{\mathrm{vl}}(\bar{s}, \bar{a}) \leq Q(\bar{s}, \bar{a})$ then
51:                 $Q(\bar{s}, \bar{a}) \leftarrow Q_{\mathrm{vl}}(\bar{s}, \bar{a})$
52:             end if
53:         end for
54:     end if
55: end for
```

linear scaling with $|S|$. These bounds show R-max to be advantageous when the accuracy of the resulting policy is important, while Delayed Q-learning looks as if it scales better with the size of the state space.

It appears that Algorithm can bring together the best of both worlds; we bound its sample complexity by $\mathcal{O}\Big(\min\Big\{\mathcal{O}\big(\frac{|S|^2|A|}{\epsilon^3(1-\gamma)^8}\big), \mathcal{O}\big(\frac{|S||A|}{\epsilon^4(1-\gamma)^8}\big)\Big\}\Big)$. To present the proof sketch for the PAC properties of the DDQ algorithm and its sample complexity analysis, we need to introduce some technical definitions and results first. The results (lemmas) are provided here without a proof due to lack of space—their detailed proofs will be offered in a subsequent publication, and they basically follow the pattern of comparable known results [14]. For simplicity, let $\kappa \triangleq |S||A|(1 + \frac{1}{(1-\gamma)\epsilon_1})$.

Definition 3: An event when $\mathsf{learn}(s,a) = \mathrm{true}$ and at the same time $l(s,a) = m_1$ or $n(s,a) = m_2$, is called an *attempted update*.

Definition 4: At any timestep t in the execution of DDQ algorithm the set of *known state-action pairs* is defined as $K_t = \big\{(s,a) \mid n(s,a) \geq m_2$ or $Q_t(s,a) - \big(R(s,a) + \gamma\sum_{s'} T(s,a,s')v_t(s')\big) \leq 3\epsilon_1\big\}$.

In subsequent analysis, and to distinguish between the conditions that make a state-action pair (s,a) known, the set K_t will be partitioned into two subsets: $K_t^1 = \big\{(s,a) \mid Q_t(s,a) - \big(R(s,a) + \gamma\sum_{s'} T(s,a,s')v_t(s')\big) \leq 3\epsilon_1\big\}$, and $K_t^2 = \big\{(s,a) \mid n(s,a) \geq m_2\big\}$.

Definition 5: In the execution of DDQ algorithm a timestep t is called a *successful timestep* if at that step any state-action value is updated or the number of times that a state-action pair is visited reaches m_2.

Recall that a type-1 update necessarily decreases the Q-value by at least ϵ_1. Defining rewards as positive quantities prevents the Q-values from becoming negative. At the same time, state-action pairs can initiate update type-2 only once they are experienced m_2 times. Together, these conditions facilitate the establishment of an upper-bound on the total number of successful timesteps during the execution of DDQ:

Lemma 1: The total number of successful timesteps in the DDQ algorithm is bounded by κ.

Choosing m_1 big enough and applying Hoefding's inequality allows following conclusion (Lemma 2) for all type-1 updates, and paves the way for establishing the optimism condition of Theorem 1.

Lemma 2: Suppose that at time t during the execution of DDQ a state-action pair (s,a) experiences a successful update of type-1 with its value changing from $Q(s,a)$ to $Q'(s,a)$, and that there exists $\exists\epsilon_2 \in (0, \frac{\epsilon_1}{2})$ such that $\forall s \in S$ and $\forall t' < t$, $v_{t'}(s) \geq v_M^*(s) - 2\epsilon_2$. If

$$m_1 \geq \frac{\ln\big(\frac{8|S||A|(1+\kappa)}{\delta}\big)}{2(\epsilon_1 - 2\epsilon_2)^2(1-\gamma)^2} \simeq \mathcal{O}\left(\frac{\ln\big(\frac{|S|^2|A|^2}{\delta}\big)}{\epsilon_1^2(1-\gamma)^2}\right) \quad (3)$$

for $\kappa = |S||A|(1 + 1/(1-\gamma)\epsilon_1)$, then $Q'(s,a) \geq Q_M^*(s,a)$ with probability at least $1 - \frac{\delta}{8}$.

The following two lemmas are borrowed from [14] with very minor modifications, and inform on how to choose parameter m_2, and the number of iterations for the value iteration part of the DDQ algorithm in order to obtain a desired accuracy.

Lemma 3: (cf. [14, Proposition 4]) Suppose the value-iteration algorithm runs on MDP M for $\frac{\ln(1/\epsilon_2(1-\gamma))}{1-\gamma}$ iterations, and each state-action value estimate $Q(s,a)$ is initialized to some value between 0 and $v_{\max}$ for all states and actions. Let $Q'(s,a)$ be the state-action value estimate the algorithm yields. Then $\max_{s,a}\big\{|Q'(s,a) - Q_M^*(s,a)|\big\} \leq \epsilon_2$.

Lemma 4: (cf. [14]) Consider an MDP M with reward function R and transition probabilities T. Suppose another MDP $\hat{M}$ has the same state and action set as M, but maintains an maximum likelihood (ML) estimate of R and T, with $n(s,a) \geq m_2$, in the form of $\hat{R}$ and $\hat{T}$ respectively. With C a constant and for all state-action pairs, choosing

$$m_2 \geq C\left(\frac{|S| + \ln\big(8|S||A|/\delta\big)}{\epsilon_2^2(1-\gamma)^4}\right) \simeq O\left(\frac{|S| + \ln\big(|S||A|/\delta\big)}{\epsilon_2^2(1-\gamma)^4}\right)$$

guarantees that $|R(s,a) - \hat{R}(s,a)| \leq C\epsilon_2(1-\gamma)^2$ and $\|T(s,a,\cdot) - \hat{T}(s,a,\cdot)\|_1 \leq C\epsilon_2(1-\gamma)^2$, with probability at least $1-\frac{\delta}{8}$. Moreover, for any policy π and for all state-action pairs, $|Q_M^\pi(s,a) - Q_{\hat{M}}^\pi(s,a)| \leq \epsilon_2$ and $|v_M^\pi(s) - v_{\hat{M}}^\pi(s)| \leq \epsilon_2$ with probability at least $1 - \frac{\delta}{8}$.

Lemmas 3 and 4 together have as a consequence the following Lemma, which contributes to establishing the accuracy condition of Theorem 1 for the DDQ algorithm.

Lemma 5: During the execution of DDQ, for all t and $(s,a) \in S \times A$, we have:

$$Q_{M_{K_t^2}}^*(s,a) - 2\epsilon_2 \leq Q_t(s,a) \leq Q_{M_{K_t^2}}^*(s,a) + 2\epsilon_2 \quad (4)$$

with probability at least $1 - \frac{3\delta}{8}$.

Lemma 1 has already offered a bound on the number of updates in DDQ; however, for the complexity condition of Theorem 1 to be satisfied, one needs to show that during the execution of Algorithm IV the number of escape events is also bounded. The following Lemma is the first step: it states that by picking m_1 as in (3), and under specific conditions, an escape event necessarily results in a successful type-1 update. With the number of updates bounded, Lemma 6 can be utilized to derive a bound on the number of escape events.

Lemma 6: With the choice of m_1 as in (3), and assuming the DDQ algorithm at timestep t with $(s,a) \notin K_t$, $l(s,a) = 0$ and $\mathsf{learn}(s,a) = \mathrm{true}$, we know that an attempted type-1 update of $Q(s,a)$ will necessarily occur within m_1 occurrences of (s,a) after t, say at time step t_{m_1}. If (s,a) is visited fewer than m_2 till t_{m_1}, the attempted type-1 update at t_{m_1} is successful with probability at least $1 - \frac{\delta}{8}$.

A bound on the number the escape events of DDQ algorithm can be derived in a straightforward way. Note that a state-action pair that is visited m_2 times becomes a permanent member of set K_t. Therefore, the number of escape events is bounded by $|S||A|m_2$. On the other hand, Lemma 6 and the learn flag mechanism suggest another upper bound on escape events. The following Lemma simply states an upper bound for escape events in DDQ as the minimum among the two bounds.

Lemma 7: During the execution of DDQ, if the assumption of Lemma 6 holds, the total number of timesteps with $(s_t, a_t) \notin K_t$ (i.e., escape events) is at most $\min\{2m_1\kappa, |S||A|m_2)\}$.

The theorem that follows is the main result of this paper. The statement establishes the PAC properties of the DDQ algorithm and provides a bound on its sample complexity.

Theorem 2: Consider an MDP $M = \{S, A, T, R, \gamma\}$, and let $\epsilon \in (0, \frac{1}{1-\gamma})$, and $\delta \in (0, 1)$. There exist $m_1 = \mathcal{O}\left(\ln(|S|^2|A|^2/\delta)/\epsilon_1^2(1-\gamma)^2\right)$ and $m_2 = \mathcal{O}\left(|S| + \ln(|S||A|/\delta)/\epsilon_2^2(1-\gamma)^4\right)$ with $\frac{1}{\epsilon_1} = \frac{3}{(1-\gamma)\epsilon} = \mathcal{O}\left(1/\epsilon(1-\gamma)\right)$ and $\epsilon_2 = \frac{\epsilon_1}{3}$, such that if DDQ algorithm is executed, M follows a 4ϵ-optimal policy with probability at least $1 - 2\delta$ on all but $\mathcal{O}\left(\min\left\{\mathcal{O}(|S|^2|A|/\epsilon^3(1-\gamma)^8), \mathcal{O}(|S||A|/\epsilon^4(1-\gamma)^8)\right\}\right)$ time steps (logarithmic factors ignored).

Proof: We intend to apply Theorem 1. To satisfy the *optimism* condition, we start by proving that $Q_t(s,a) \geq Q^*_M(s,a) - 2\epsilon_2$ by strong induction for all state-action pairs: (i) At $t = 1$, the value of all state-action pairs are set to the maximum possible value in MDP M. This implies that $Q_1(s,a) \geq Q^*_M(s,a) \geq Q^*_M(s,a) - 2\epsilon_2$, therefore $v_t(s) \geq v^*_M(s) - 2\epsilon_2$. (ii) Assume that $Q_t(s,a) \geq Q^*_M(s,a) - 2\epsilon_2$ holds for all timesteps before or equal to $t = n - 1$. (iii) At timestep $t = n$, all $(s,a) \notin K^2_n$ can only be updated by a type-1 update before or at $t = n$. For these state-action pairs, Lemma 2 implies that it will be $Q_n(s,a) \geq Q^*_M(s,a)$ with probability $1 - \frac{\delta}{8}$.

For all $(s,a) \in K^2_n$, on the other hand, by Lemma 5 and with probability $1 - \frac{3\delta}{8}$,

$$Q_n(s,a) \geq Q^*_{M_{K^2_n}}(s,a) - 2\epsilon_2 \geq Q^*_M(s,a) - 2\epsilon_2$$

Note that $Q^*_{M_{K^2_n}}(s,a) \geq Q^*_M(s,a)$ since $M_{K^2_n}$ is similar to M exept for $(s,a) \notin K^2_n$ which their values are set to be more than or equal to $Q^*_M(s,a)$. Therefore, $Q_t(s,a) \geq Q^*_M(s,a) - 2\epsilon_2$ holds for all timesteps t and all state-action pairs, which directly implies $v_t(s) \geq v^*_M(s) - 2\epsilon_2 \geq v^*_M(s) - \epsilon$.

To establish the *accuracy* condition, first write

$$Q_t(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q_t(s',a') + \beta(s,a) \quad (5)$$

If $(s,a) \in K_t$, there can be two cases: either $(s,a) \in K^1_t$ or $(s,a) \in K^2_t$. If $(s,a) \in K^1_t$, then by Definition 4 $\beta(s,a) \leq 3\epsilon_1$. If $(s,a) \in K^2_t$, then Lemma 5 (right-hand side inequality) implies that with probability at least $1 - \frac{3\delta}{8}$

$$2\epsilon_2 \geq Q_t(s,a) - Q^*_{M_{K^2_t}}(s,a) \quad (6)$$

Meanwhile,

$$Q^*_{M_{K^2_t}}(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a'} Q^*_{M_{K^2_t}}(s',a') \quad (7)$$

and substituting from (7) and (5) into (6) yields

$$\gamma \sum_{s'} T(s,a,s') \left(\max_{a'} Q_t(s',a') - \max_{a'} Q^*_{M_{K^2_t}}(s',a') \right) + \beta(s,a) \leq 2\epsilon_2 \quad (8)$$

Let $a_1 := \arg\max_{a'} Q_{M_{K^2_t}}(s',a')$ and bound the difference

$$\begin{aligned} \max_{a'} Q_t(s',a') - \max_{a'} Q^*_{M_{K^2_t}}(s',a') &= \max_{a'} Q_t(s',a') - Q^*_{M_{K^2_t}}(s',a_1) \\ &\geq Q_t(s',a_1) - Q^*_{M_{K^2_t}}(s',a_1) \end{aligned}$$

Apply Lemma 5 (left-hand side inequality) to the latter expression to get $\max_{a'} Q_t(s',a') - \max_{a'} Q^*_{M_{K^2_t}}(s',a') \geq -2\epsilon_2$, which implies for (8) that $2\epsilon_2 \geq \beta(s,a) - 2\gamma\epsilon_2 \implies \beta(s,a) \leq 2(1+\gamma)\epsilon_2 \leq 3\epsilon_2$. Thus in any case when $(s,a) \in K_t$, $\beta(s,a) \leq 3\epsilon_1$ with probability at least $1 - \frac{3\delta}{8}$. In light of this, we write for the values of states in which $(s, \pi_t(s)) \in K_t$: $v^{\pi_t}_{M_{K_t}}(s) = R(s,\pi_t(s)) + \gamma \sum_{s'} T(s,\pi_t(s),s') v^{\pi_t}_{M_{K_t}}(s')$ and $v_t(s) = R(s,\pi_t(s)) + \gamma \sum_{s'} T(s,\pi_t(s),s') v_t(s') + \beta(s,a)$, while for $(s,\pi_t(s)) \notin K_t$, we already know that $v^{\pi_t}_{M_{K_t}}(s) = Q_t(s,\pi_t(s))$ and $v_t(s) = Q_t(s,\pi_t(s))$. So now if one denotes $\alpha := \max_s (v_t(s) - v^{\pi_t}_{M_{K_t}}(s)) = v_t(s^*) - v^{\pi_t}_{M_{K_t}}(s^*)$ then either $\alpha = 0$ (when $(s,\pi_t(s)) \notin K_t$) or it affords an upper bound

$$\begin{aligned} &\gamma \sum_{s'} T(s^*,\pi_t(s^*),s') \left(v_t(s') - v^{\pi_t}_{M_{K_t}}(s') \right) + \beta(s^*,\pi_t(s^*)) \\ &\leq \gamma \sum_{s'} T(s^*,\pi_t(s^*),s') \left(v_t(s') - v^{\pi_t}_{M_{K_t}}(s') \right) + 3\epsilon_1 \leq \gamma\alpha + 3\epsilon_1 \end{aligned}$$

from which it follows that $\alpha \leq \gamma\alpha + 3\epsilon_1 \implies \alpha \leq \frac{3\epsilon}{1-\gamma} = \epsilon$.

Finally, to analyze *complexity* invoke Lemmas 1 and 7 to see that the learning complexity $\zeta(\epsilon,\delta)$ is bounded by $\kappa + \min(2m_1\kappa, |S||A|m_2)$ with probability $1 - \frac{\delta}{8}$.

In conclusion, the conditions of Theorem 2 are satisfied with probability $1 - \delta$ and therefore the DDQ algorithm is PAC. Substituting $\zeta(\epsilon,\delta)$ into (2) completes the proof. ■

V. NUMERICAL RESULTS

In this section, R-max, Delayed Q-learning and DDQ algorithms are compared on an MDP model of the infant-robot interaction during the the pediatric motor rehabilitation sessions referred to in Section I. In the pediatric rehabilitation example that motivated the approach outlined in this paper, infants with motor disabilities are socially interacting with robots in play-based scenarios that involve physical activity. The play-based intervention is intended to incite infant mobility. If this form of HRI is to be automated, the robot has to have a way of deciding what is the intervention-appropriate response to child behavior in order to keep her engaged in play-based physical activity.

In this MDP, the action set (robot's actions) is defined as $A = \{f, s, b\}$, with f indicating the robot moving "forward" and toward the child, s associated with keeping the same distance to the child, and b representing retreating facing the child. The states of MDP are $S = \{NL, L, T/A, M\}$; with NL representing that the child is not looking at the robot, L expressing the child looking at the robot but not following it, T/A indicates that the robot has attracted the attention of the child, and M stands for the case where the child is following the robot. When the child makes transition to

T/A or M, this is considered a positive development and the system accrues a reward. Thus positive utility is assigned to these favorable states, and negative or zero utility is assigned to the remaining ones. We assign $R = \{0, 0, 0.5, 1\}$ utility for these states respectively. The robot employs the DDQ (Algorithm 1) in this MDP model, where it does not know the transition probabilities. We use data from 6 subjects to estimate the transition probabilities of the MDP and use them as the ground truth. These are as follows (rows are the beginning states and columns are the final states in the order $NL = 1, L = 2, T/A = 3, M = 4$):

$$T_f = \begin{bmatrix} 0.20 & 0.69 & 0.08 & 0.03 \\ 0.21 & 0.33 & 0.35 & 0.21 \\ 0.10 & 0.21 & 0.41 & 0.28 \\ 0.04 & 0.18 & 0.41 & 0.37 \end{bmatrix}, \; T_s = \begin{bmatrix} 0.45 & 0.45 & 0.07 & 0.03 \\ 0.09 & 0.44 & 0.28 & 0.19 \\ 0.29 & 0.04 & 0.46 & 0.21 \\ 0.29 & 0.08 & 0.50 & 0.13 \end{bmatrix}$$

$$T_b = \begin{bmatrix} 0.26 & 0.65 & 0.05 & 0.04 \\ 0.23 & 0.48 & 0.09 & 0.20 \\ 0.06 & 0.22 & 0.22 & 0.50 \\ 0.06 & 0.10 & 0.48 & 0.36 \end{bmatrix}$$

TABLE I

AVERAGE NUMBER OF SAMPLES FOR 4ε OPTIMALITY

	Delayed Q	R-max	DDQ
# of samples	1597	1526	1194

Initializing the three PAC algorithms with the following parameters: $m_1 = 50$, $m_2 = 100$, $\gamma = 0.8$ and $\varepsilon = 0.06$, yields performance metrics shown in Table I, in terms of the number of samples needed to reach at 4ε optimality, averaged over 10 algorithm runs. Parameters m_1 and m_2 are intentionally chosen in a way that the sample complexity of the model-free Delayed Q-learning, and the model-based R-max algorithms are almost identical. In this case, and with these same tuning parameters, the sample complexity improvement that DDQ offers becomes apparent.

VI. CONCLUSION

By leveraging the Dyna-Q idea, a PAC RL algorithm can capture the best of both model-free and model-based worlds, and exhibit the best (sample complexity) performance that is appropriate for a given application. The algorithm introduced by this paper is called DDQ, and it integrates processes from the model-based R-max and the model-free Delayed Q-learning algorithms. The paper theoretically establishes the PAC properties of DDQ, proves that none of its constituent methodologies can outperform it, and in fact demonstrates in numerical examples that DDQ itself can outperform them.

REFERENCES

[1] Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49(2-3):209–232, 2002.

[2] Ronen I Brafman and Moshe Tennenholtz. R-max a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3(Oct):213–231, 2002.

[3] Alexander L Strehl and Michael L Littman. A theoretical analysis of model-based interval estimation. In *Proceedings of the 22nd International Conference on Machine learning*, pages 856–863. ACM, 2005.

[4] Alexander L Strehl, Lihong Li, and Michael L Littman. Incremental model-based learners with formal learning-time guarantees. *arXiv preprint arXiv:1206.6870*, 2012.

[5] Alexander L Strehl, Lihong Li, Eric Wiewiora, John Langford, and Michael L Littman. PAC model-free reinforcement learning. In *Proceedings of the 23rd International Conference on Machine learning*, pages 881–888. ACM, 2006.

[6] A. Zehfroosh, E. Kokkoni, H. G. Tanner, and J. Heinz. Learning models of human-robot interaction from small data. In *2017 25th IEEE Mediterranean Conference on Control and Automation*, pages 223–228, July 2017.

[7] Frank Broz, Illah Nourbakhsh, and Reid Simmons. Planning for Human-Robot Interaction in Socially Situated Tasks: The Impact of Representing Time and Intention. *International Journal of Social Robotics*, 5(2):193–214, 2013.

[8] Tirthankar Bandyopadhyay, Kok Sung Won, Emilio Frazzoli, David Hsu, Wee Sun Lee, and Daniela Rus. Intention-Aware Motion Planning. In E. Frazzoli et al., editor, *Algorithmic Foundations of Robotics X*, volume 86, pages 475–491. Springer-Verlag, 2013.

[9] Catharine LR McGhan, Ali Nasir, and Ella M Atkins. Human intent prediction using markov decision processes. *Journal of Aerospace Information Systems*, 5(12):393–397, 2015.

[10] A. Zehfroosh, H. G. Tanner, and J. Heinz. Learning option mdps from small data. In *2018 IEEE American Control Conference*, pages 252–257, 2018.

[11] Ashkan Zehfroosh and Herbert G Tanner. Reactive motion planning for temporal logic tasks without workspace discretization. In *2019 American Control Conference (ACC)*, pages 4872–4877. IEEE, 2019.

[12] Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. Temporal difference models: Model-free deep RL for model-based control. *arXiv preprint arXiv:1802.09081*, 2018.

[13] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation*, pages 7559–7566, 2018.

[14] Alexander L Strehl, Lihong Li, and Michael L Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10(Nov):2413–2444, 2009.

[15] Sham Machandranath Kakade et al. *On the sample complexity of reinforcement learning*. PhD thesis, University of London London, England, 2003.

[16] Sang Wan Lee, Shinsuke Shimojo, and John P O'Doherty. Neural computations underlying arbitration between model-based and model-free learning. *Neuron*, 81(3):687–699, 2014.

[17] Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

[18] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning continuous control policies by stochastic value gradients. In *Advances in Neural Information Processing Systems*, pages 2944–2952, 2015.

[19] Ronald Parr, Lihong Li, Gavin Taylor, Christopher Painter-Wakefield, and Michael L Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, pages 752–759. ACM, 2008.

[20] Richard S Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M Pilarski, Adam White, and Doina Precup. Horde: A scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 761–768. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[21] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In *International Conference on Machine Learning*, pages 1312–1320, 2015.

[22] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[23] Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 703–711. JMLR. org, 2017.