The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)

How to Combine Tree-Search Methods in Reinforcement Learning

Yonathan Efroni
Technion, Israel

Gal Dalal
Technion, Israel

Bruno Scherrer
INRIA, Villers les Nancy, France

Shie Mannor
Technion, Israel

Abstract

Finite-horizon lookahead policies are abundantly used in Reinforcement Learning and demonstrate impressive empirical success. Usually, the lookahead policies are implemented with specific planning methods such as Monte Carlo Tree Search (e.g. in AlphaZero (Silver et al. 2017b)). Referring to the planning problem as tree search, a reasonable practice in these implementations is to back up the value only at the leaves while the information obtained at the root is not leveraged other than for updating the policy. Here, we question the potency of this approach. Namely, the latter procedure is non-contractive in general, and its convergence is not guaranteed. Our proposed enhancement is straightforward and simple: use the return from the optimal tree path to back up the values at the descendants of the root. This leads to a γ^h-contracting procedure, where γ is the discount factor and h is the tree depth. To establish our results, we first introduce a notion called *multiple-step greedy consistency*. We then provide convergence rates for two algorithmic instantiations of the above enhancement in the presence of noise injected to both the tree search stage and value estimation stage.

1 Introduction

A significant portion of the Reinforcement Learning (RL) literature regards Policy Iteration (PI) methods. This family of algorithms contains numerous variants which were thoroughly analyzed (Puterman 1994; Bertsekas and Tsitsiklis 1995) and constitute the foundation of sophisticated state-of-the-art implementations (Mnih et al. 2016; Silver et al. 2017b). The principal mechanism of PI is to alternate between policy evaluation and policy improvement. Various well-studied approaches exist for the policy evaluation stages; these may rely on single-step bootstrap, multi-step Monte-Carlo return, or parameter-controlled interpolation of the former two. For the policy improvement stage, theoretical analysis was mostly reserved for policies that are 1-step greedy, while recent prominent implementations of multiple-step greedy policies exhibited promising empirical behavior (Silver et al. 2017b; 2017a).

Relying on recent advances in the analysis of multiple-step lookahead policies (Efroni et al. 2018a; 2018c), we study the convergence of a PI scheme whose improvement stage is h-step greedy with respect to (w.r.t.) the value function, for $h > 1$. Calculating such policies can be done via Dynamic Programming (DP) or other planning methods such as tree search. Combined with sampling, the latter corresponds to the famous Monte Carlo Tree Search (MCTS) algorithm employed in (Silver et al. 2017b; 2017a). In this work, we show that even when partial (inexact) policy evaluation is performed and noise is added to it, along with a noisy policy improvement stage, the above PI scheme converges with a γ^h contraction coefficient. While doing so, we also isolate a sufficient convergence condition which we refer to as *h-greedy consistency* and relate it to previous 1-step greedy relevant literature.

A straightforward 'naive' implementation of the PI scheme described above would perform an h-step greedy policy improvement and then evaluate that policy by bootstrapping the 'usual' value function. Surprisingly, we find that this procedure does not necessarily contracts toward the optimal value, and give an example where it is indeed non-contractive. This contraction coefficient depends both on h and on the partial evaluation parameter: m in the case of m-step return, and λ when eligibility trace is used. The non-contraction occurs even when the h-greedy consistency condition is satisfied.

To solve this issue, we propose an easy fix which we employ in all our algorithms, that relieves the convergence rate from the dependence of m and λ, and allows the γ^h contraction mentioned earlier in this section. Let us treat each state as a root of a tree of depth h; then our proposed fix is the following. Instead of backing up the value only at the leaves and ridding of all non-root related tree-search outputs, we reuse the tree-search byproducts and back up the optimal value of the root node children. Hence, instead of bootstrapping the 'usual' value function in the evaluation stage, we bootstrap the optimal value obtained from the $h-1$ horizon optimal planning problem.

The contribution of this work is primarily theoretical, but in Section 8 we also present experimental results on a toy domain. The experiments support our analysis by exhibiting better performance of our enhancement above compared to the 'naive' algorithm. Additionally, we identified previous practical usages of this enhancement in literature. In (Baxter, Tridgell, and Weaver 1999), the authors proposed backing up the optimal tree search value as a heuristic. They named the algorithm TDLeaf(λ) and showcase its outperformance

over the alternative 'naive' approach. A more recent work (Lai 2015) introduced a deep learning implementation of TDLeaf(λ) called Giraffe. Testing it on the game of Chess, the authors claim (during publication) it is "the most successful attempt thus far at using end-to-end machine learning to play chess". In light of our theoretical results and empirical success described above, we argue that backing up the optimal value from a tree search should be considered as a 'best practice' among RL practitioners.

2 Preliminaries

Our framework is the infinite-horizon discounted Markov Decision Process (MDP). An MDP is defined as the 5-tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ (Puterman 1994), where $\mathcal{S}$ is a finite state space, $\mathcal{A}$ is a finite action space, $P \equiv P(s'|s,a)$ is a transition kernel, $R \equiv r(s,a) \in [R_{\min}, R_{\max}]$ is a reward function, and $\gamma \in (0,1)$ is a discount factor. Let $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ be a stationary policy, where $\mathcal{P}(\mathcal{A})$ is a probability distribution on $\mathcal{A}$. Let $v^\pi \in \mathbb{R}^{|\mathcal{S}|}$ be the value of a policy π, defined in state s as $v^\pi(s) \equiv \mathbb{E}^\pi_{|s}[\sum_{t=0}^\infty \gamma^t r(s_t, \pi(s_t))]$, where $\mathbb{E}^\pi_{|s}$ denotes expectation w.r.t. the distribution induced by π and conditioned on the event $\{s_0 = s\}$. For brevity, we respectively denote the reward and value at time t by $r_t \equiv r(s_t, \pi_t(s_t))$ and $v_t \equiv v(s_t)$. It is known that $v^\pi = \sum_{t=0}^\infty \gamma^t (P^\pi)^t r^\pi = (I - \gamma P^\pi)^{-1} r^\pi$, with the component-wise values $[P^\pi]_{s,s'} \triangleq P(s' \mid s, \pi(s))$ and $[r^\pi]_s \triangleq r(s, \pi(s))$. Our goal is to find a policy π^* yielding the optimal value v^* such that $v^* = \max_\pi (I - \gamma P^\pi)^{-1} r^\pi$. This goal can be achieved using the three classical operators (with equalities holding component-wise):

$$\forall v, \pi,\ T^\pi v = r^\pi + \gamma P^\pi v, \tag{1}$$

$$\forall v,\ Tv = \max_\pi T^\pi v, \tag{2}$$

$$\forall v,\ \mathcal{G}(v) = \{\pi : T^\pi v = Tv\}, \tag{3}$$

where T^π is a linear operator, T is the optimal Bellman operator and both T^π and T are γ-contraction mappings w.r.t. the max norm. It is known that the unique fixed points of T^π and T are v^π and v^*, respectively. The set $\mathcal{G}(v)$ is the standard set of 1-step greedy policies w.r.t. v. Furthermore, given v^*, the set $\mathcal{G}(v^*)$ coincides with that of stationary optimal policies. In other words, every policy that is 1-step greedy w.r.t. v^* is optimal and vice versa.

The most known variants of PI are Modified-PI (Puterman and Shin 1978) and λ-PI (Bertsekas and Ioffe 1996). In both, the evaluation stage of PI is relaxed by performing partial-evaluation, instead of the full policy evaluation. In this work, we will generalize algorithms using both of these approaches. Modified PI performs partial evaluation using the m-return, $(T^\pi)^m v$, where λ-PI uses the λ-return, $T^\pi_\lambda v$, with $\lambda \in [0,1]$. This operator has the following equivalent forms (see e.g. (Scherrer 2013), p.1182),

$$T^\pi_\lambda v \stackrel{\text{def}}{=} (1-\lambda) \sum_{j=0}^\infty \lambda^j (T^\pi)^{j+1} v \tag{4}$$

$$= v + (I - \gamma\lambda P^\pi)^{-1} (T^\pi v - v).$$

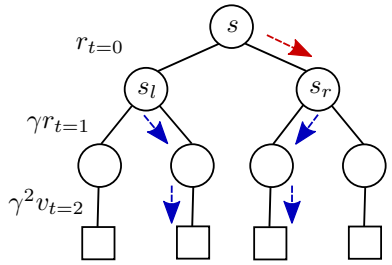

Figure 1: Obtaining the h-greedy policy with a tree-search also outputs $T^{\pi_h} T^{h-1} v$ and $T^{h-1} v$. In this example, the red arrow depicts the h-greedy policy. The value at the root's child node s_l is $T^{h-1} v(s_l)$, which corresponds to the optimal blue trajectory starting at s_l. The same holds for s_r.

These operators correspond to the ones used in the famous TD(n) and TD(λ) (Sutton, Barto, and others 1998),

$$(T^\pi)^m v = \mathbb{E}^\pi_{|\bullet} \left[\sum_{t=0}^{m-1} \gamma^t r(s_t, \pi_t(s_t)) + \gamma^m v(s_h) \right],$$

$$T^\pi_\lambda v = v + \mathbb{E}^\pi_{|\bullet} \left[\sum_{t=0}^{\infty} (\gamma\lambda)^t (r_t + \gamma v_{t+1} - v_t) \right].$$

3 The h-Greedy Policy and h-PI

Let $h \in \mathbb{N} \backslash \{0\}$. An h-greedy policy (Bertsekas and Tsitsiklis 1995; Efroni et al. 2018a) π_h outputs the first optimal action out of the sequence of actions solving a non-stationary, h-horizon control problem as follows:

$$\arg\max_{\pi_0} \max_{\pi_1,..,\pi_{h-1}} \mathbb{E}^{\pi_0 \cdots \pi_{h-1}}_{|\bullet} \left[\sum_{t=0}^{h-1} \gamma^t r(s_t, \pi_t(s_t)) + \gamma^h v(s_h) \right]$$

$$= \arg\max_{\pi_0} \mathbb{E}^{\pi_0}_{|\bullet} \left[r(s_0, \pi_0(s_0)) + \gamma \left(T^{h-1} v\right)(s_1) \right], \tag{5}$$

where the notation $\mathbb{E}^{\pi_0 \cdots \pi_{h-1}}_{|\bullet}$ corresponds to conditioning on the trajectory induced by the choice of actions $(\pi_0(s_0), \pi_1(s_1), \ldots, \pi_{h-1}(s_{h-1}))$ and a starting state $s_0 = \bullet$.

As the equality in (5) suggests that π_h can be interpreted as a 1-step greedy policy w.r.t. $T^{h-1} v$. We denote the set of h-greedy polices w.r.t v as $\mathcal{G}_h(v)$ and is defined by

$$\forall v,\ \mathcal{G}_h(v) = \{\pi : T^\pi T^{h-1} v = T^h v\}.$$

This generalizes the definition of the 1-step greedy set of policies, generalizing, (3), and coincides with it for $h = 1$.

Remark 1. *The h-greedy policy can be obtained by solving the above formulation with DP in linear time (in h). Other than returning the policy, the last and one-before-last iterations also return $T^{\pi_h} T^{h-1} v$ and $T^{h-1} v$, respectively. Another, conceptually similar option would be using Model Predictive Control to solve the planning problem and again retrieve the above values of interest (Negenborn et al. 2005; Tamar et al. 2017). Given a 'nice' mathematical structure, this can be done efficiently. When the model is unknown, finding π_h together with $T^{\pi_h} T^{h-1} v$ and $T^{h-1} v$ is possible with model-free approaches such as Q-learning (Jin et al. 2018). Alternatively, $\pi_h(s)$ can be retrieved using a tree-search of*

depth h, starting at root s (see Figure 1). The search again returns $T^{\pi_h}T^{h-1}v$ and $T^{h-1}v$ "for free" as the values at the root and its descendant nodes. While the tree-search complexity in general is exponential in h, sampling can be used. Examples for such sampling-based tree-search methods are MCTS (Browne et al. 2012) and Optimistic Tree Exploration (Munos 2014).

Algorithm 1 h-PI

Initialize: $h \in \mathbb{N} \setminus \{0\},\ v_0 = v^{\pi_0} \in \mathbb{R}^{|\mathcal{S}|}$
while v_k changes **do**
  $\pi_k \leftarrow \pi \in \mathcal{G}_h(v)$
  $v_{k+1} \leftarrow v^{\pi_k}$
  $k \leftarrow k+1$
end while
Return π, v

As was discussed in (Bertsekas and Tsitsiklis 1995; Efroni et al. 2018a), one can use the h-greedy policy to derive a policy-iteration procedure called h-PI (see Algorithm 1). In it, the 1-step greedy policy from PI is replaced with the h-greedy policy. This algorithm iteratively calculates an h-step greedy policy with respect to v, and then performs a complete evaluation of this policy. Convergence is guaranteed after $O(h^{-1})$ iterations (Efroni et al. 2018a).

4 h-Greedy Consistency

The h-greedy policy w.r.t v^π is strictly better than π, i.e., $v^{\pi_h} \geq v^\pi$ (Bertsekas and Tsitsiklis 1995; Efroni et al. 2018a). Using this property for proving convergence of an algorithm requires the algorithm to perform exact value estimation, which can be a hard task. Instead, in this work, we replace the less practical exact evaluation with partial evaluation; this comes with the price of more challenging analysis. Tackling this more intricate setup, we identify a key property required for the analysis to hold. We refer to it as *h-greedy consistency*. It will be central to all proofs in this work.

Definition 1. *A pair of value function and policy (v, π) is h-greedy consistent if $T^\pi T^{h-1}v \geq T^{h-1}v$.*

In words, (v, π) is h-greedy consistent if π 'improves', component-wise, the value $T^{h-1}v$. Since relaxing the evaluation stage comes with the h-greedy consistency requirement, the following question arises: while dispatching an algorithm, what is the price ensuring h-greedy consistency per each iteration? As we will see in the coming sections, it is enough to ensure h-greedy consistency only for the first iteration of our algorithms. For the rest of the iterations it holds by construction and is shown to be guaranteed in our proofs. Thus, by only initializing to an h-greedy consistent (v_0, π_0), we enable guaranteeing the convergence of an algorithm that performs partial evaluation instead of exact in each its iterations. Ensuring consistency for the first iteration is straightforward, as is explained in the following remark.

Remark 2. *Choosing (v, π) which is h-greedy consistent can be done, e.g., by choosing $v = \frac{R_{\min}}{1-\gamma}$ (i.e., set every entrance of v to the minimal possible accumulated reward) and $\pi = \pi_h \in \mathcal{G}_h(v)$. Furthermore, for any value-policy, $(\bar{v}, \pi)$, that is not h-greedy consistent, let*

$$\Delta = \frac{\max_s \left(T^{h-1}\bar{v} - T^\pi T^{h-1}\bar{v}\right)(s)}{\gamma^{h-1}(1-\gamma)} > 0,$$

and set $v = \bar{v} - \Delta$. Then, (v, π) is h-greedy consistent. This is a generalization to the construction given for $h = 1$ (see (Bertsekas and Tsitsiklis 1995), p. 46).

h-greedy consistency is an h-step generalization of a notion already introduced in previous works on 1-step-based PI schemes with partial evaluation. The latter are known as 'optimistic' PI schemes and include Modified PI and λ-PI (Bertsekas and Tsitsiklis 1995). There, the initial value-policy pair is assumed to be 1-greedy consistent, i.e. $T^{\pi_1}v_0 \geq v_0$, e.g., (Bertsekas and Tsitsiklis 1995), p. 32 and 45, (Bertsekas 2011), p. 3, (Puterman and Shin 1978)[Theorem 2]. This property served as an assumption on the pair (v_0, π_1).

To further motivate our interest in Definition 1, in the rest of the section we give two results that would be used in proofs later but are also insightful on their own. The following lemma gives that h-greedy consistency implies a sequence of value-function partial evaluation relations (proof in (Efroni et al. 2018b)).

Lemma 1. *Let (v, π) be h-greedy consistent. Then,*

$$T^\pi T^{h-1}v \leq \cdots \leq (T^\pi)^l T^{h-1}v \leq \cdots \leq v^\pi.$$

The result shows that v^π is strictly bigger than $T^{h-1}v$. This property holds when $v = v^{\pi'}$, i.e., when v is an exact value of some policy and was central in the analysis of h-PI (Efroni et al. 2018a). However, as Lemma 1 suggests, we only need h-greedy consistency, which is easier to have than estimating the exact value of a policy (see Remark 2).

The next result shows that if π is taken to be the h-greedy policy, using partial evaluation results in a γ^h contraction toward the optimal value v^* (proof in (Efroni et al. 2018b)).

Proposition 2. *Let v and $\pi_h \in \mathcal{G}_h(v)$ be s.t. (v, π_h) is h-greedy consistent. Then, for any $m \geq 1$ and $\lambda \in [0, 1]$,*

$$||v^* - (T^{\pi_h})^m T^{h-1}v||_\infty \leq \gamma^h ||v^* - v||_\infty \text{ and}$$
$$||v^* - T_\lambda^{\pi_h} T^{h-1}v||_\infty \leq \gamma^h ||v^* - v||_\infty.$$

In (Efroni et al. 2018a)[Lemma 2], a similar contraction property was proved and played a central role in the analysis of the corresponding h-PI algorithm. Again, there, the requirement was $v = v^{\pi'}$. Instead, the above result requires a weaker condition: h-greedy consistency of (v, π_h).

5 The h-Greedy Policy Alone is Not Sufficient For Partial Evaluation

A more practical version of h-PI (Algorithm 1) would involve the m- or λ-return w.r.t. v_k instead of the exact value. This would correspond to the update rules:

$$\pi_k \leftarrow \arg\max_\pi T^\pi T^{h-1}v_k, \tag{6}$$
$$v_{k+1} \leftarrow (T^{\pi_k})^m v_k \text{ or } v_{k+1} \leftarrow T_\lambda^{\pi_k} v_k. \tag{7}$$

Indeed, this would relax the evaluation task to an easier task than full policy evaluation.

The next theorem suggests that for $\pi_h \in \mathcal{G}_h(v)$, even if (v, π_h) is h-greedy consistent, the procedure (6)-(7) does not necessarily contract toward the optimal policy, unlike the form of update in Proposition 2. To see that, note that both $\gamma^m + \gamma^h$ and $\frac{\gamma(1-\lambda)}{1-\lambda\gamma} + \gamma^h$ can be larger than 1.

Theorem 3. *Let $h > 1, m \geq 1$, and $\lambda \in [0, 1]$. Let v be a value function and $\pi_h \in \mathcal{G}_h(v)$ s.t. (v, π_h) is h-greedy consistent (see Definition 1). Then,*

$$||v^* - (T^{\pi_h})^m v||_\infty \leq (\gamma^m + \gamma^h)||v^* - v||_\infty, \quad (8)$$

$$||v^* - T_\lambda^{\pi_h} v||_\infty \leq \left(\frac{\gamma(1-\lambda)}{1-\lambda\gamma} + \gamma^h\right)||v^* - v||_\infty. \quad (9)$$

Additionally, there exist a γ-discounted MDP, value function v, and policy $\pi_h \in \mathcal{G}_h(v)$ s.t. (v, π_h) is h-greedy consistent, for which (8) *and* (9) *hold with equality.*

The proof of the first statement is given in (Efroni et al. 2018b), and the proof of the second statement is as follows.

Proof of second statement in Theorem 3. We prove this by constructing an example. Fix $h > 1$ and consider the corresponding 4-state MDP in Figure 2. Let v be $v(s_0) = v(s_2) = v(s_3) = 0$, $v(s_1) = -\frac{1}{1-\gamma}$. Also, let $\pi_h \in \mathcal{G}_h(v)$. For this choice, observe that $T^{h-1}v \leq T^{\pi_h}T^{h-1}v$, i.e., (v, π_h) is h-greedy consistent. The optimal policy from state s_0 is to choose the action 'up'. Thus, it is easy to see that, $v^*(s_0) = v^*(s_3) = \frac{1}{1-\gamma}$, and in the remaining of states it is easy to observe that $v^*(s_1) = v^*(s_2) = 0$.

Now, see that for any $h > 1$ $\left(T^{h-1}v\right)(s_1) = \left(T^{h-1}v\right)(s_2) = 0, \left(T^{h-1}v\right)(s_3) = \frac{1-\gamma^{h-1}}{1-\gamma}$. Thus, the h-greedy policy (by using (5)) is contained in the following set of actions $\pi_h(s_0) \in \{\text{right}, \text{up}\}, \pi_h(s_1) \in \{\text{right}, \text{stay}\}, \pi_h(s_2), \pi_h(s_3) \in \{\text{stay}\}$. For example, we see that taking the action 'stay' or 'right' from state s_1 and then obtain $T^{h-1}v$ have equal value:

$$\begin{aligned} &r(s_1, \text{'stay'}) + \gamma(T^{h-1}v)(s_1) \\ =&r(s_1, \text{'right'}) + \gamma(T^{h-1}v)(s_2) = 0. \end{aligned}$$

Let us choose an h-greedy policy, π_h, of the form: $\pi_h(s_0) = \text{right}, \pi_h(s_1) = \text{stay}, \pi_h(s_2) = \text{stay}$. Thus, from state s_0, the m-return has the value

$$\begin{aligned} &\left((T^{\pi_h})^m v\right)(s_0) = \sum_{i=0}^{m-1} \gamma^t r(s_i, \pi_h(s_i)) + \gamma^m v(s_{i=m}) \\ &= \frac{1-\gamma^m-\gamma^h}{1-\gamma} + \sum_{i=1}^{m-1} \gamma^i \cdot 0 + \gamma^m \left(-\frac{1}{1-\gamma}\right) \\ &= \frac{1-\gamma^m-\gamma^h}{1-\gamma} \end{aligned}$$

We thus have that

$$\begin{aligned} &||v^* - (T^{\pi_h})^m v||_\infty = |v^*(s_{1,0}) - (T^{\pi_h})^m v(s_{1,0})| \\ &= \frac{1}{1-\gamma} + \frac{\gamma^m + \gamma^h - 1}{1-\gamma} = (\gamma^m + \gamma^h)\frac{1}{1-\gamma} \end{aligned} \quad (10)$$

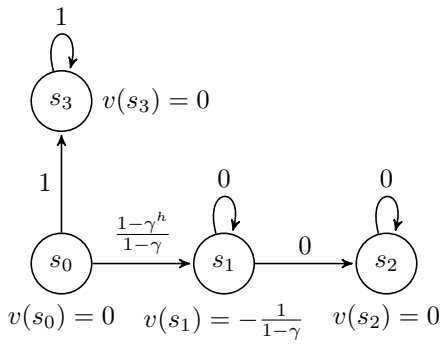

Figure 2: The MDP used in the proof of Theorem 3. NC-hm-PI and NC-$h\lambda$-PI may result in a new value that does not contract toward v^*.

It is also easy to see that $||v^* - v||_\infty = \frac{1}{1-\gamma}$. By using (10),

$$||v^* - (T^{\pi_h})^m v||_\infty = (\gamma^m + \gamma^h)||v^* - v||_\infty,$$

which concludes the tightness result on the first result in Theorem 3. The tightness proof of (9) easily follows using the same construction as above; for details see (Efroni et al. 2018b).

□

As discussed above, Theorem 3 suggests that the 'naive' partial-evaluation scheme would not necessarily lead to contraction toward the optimal value, especially for small values of h, m, λ and large γ; these are often values of interest. Moreover, the second statement in the theorem contrasts with the known result for $h = 1$, i.e., Modified PI and λ-PI. There, a γ-contraction was shown to exist (Scherrer 2013)[Proposition 8] and (Puterman and Shin 1978)[Theorem 2].

From this point onwards, we shall refer to the algorithms given in (6)-(7) and discussed in this section as Non-Contracting (NC)-hm-PI and NC-$h\lambda$-PI.

6 Backup the Tree-Search Byproducts

In the previous section, we proved that partial evaluation using the backed-up value function v, as given in (6)-(7), is not necessarily a process converging toward the optimal value. In this section, we propose a natural respective fix: back up the value $T^{h-1}v$ and perform the partial evaluation w.r.t. it. In the noise-free case this is motivated by Proposition 2, which reveals a γ^h-contraction per each PI iteration.

We now introduce two new algorithms that relax h-PI's (from Algorithm 1) exact policy evaluation stage to the more practical m- and λ-return partial evaluation. Notice that hm-PI can be interpreted as iteratively performing $h - 1$ steps of Value Iteration and one step of Modified PI (Puterman and Shin 1978), whereas instead of the latter, $h\lambda$-PI performs one step of λ-PI (Bertsekas and Ioffe 1996).

Our algorithms also account for noisy updates in both the improvement and evaluation stages. For that purpose, we first define the following approximate improvement operator.

Algorithm 2 hm-PI

Initialize: $h, m \in \mathbb{N} \setminus \{0\},\ v \in \mathbb{R}^{|\mathcal{S}|}$
while stopping criterion is false **do**
$\pi_{k+1} \leftarrow \pi \in \mathcal{G}_h^{\delta_{k+1}}(v_k)$
$v_{k+1} \leftarrow (T^{\pi_{k+1}})^m T^{h-1} v_k + \epsilon_k$
$k \leftarrow k+1$
end while
Return π, v

Algorithm 3 $h\lambda$-PI

Initialize: $h \in \mathbb{N} \setminus \{0\},\ \lambda \in [0,1],\ v \in \mathbb{R}^{|\mathcal{S}|}$
while stopping criterion is false **do**
$\pi_{k+1} \leftarrow \pi \in \mathcal{G}_h^{\delta_{k+1}}(v_k)$
$v_{k+1} \leftarrow T_\lambda^{\pi_{k+1}} T^{h-1} v_k + \epsilon_k$
$k \leftarrow k+1$
end while
Return π, v

Definition 2. *For $\hat{\delta} \in \mathbb{R}_+^{|\mathcal{S}|}$, let $\mathcal{G}_h^{\hat{\delta}}(v)$ be the approximate h-greedy set of policies w.r.t. v with error $\hat{\delta}$, s.t. for $\pi \in \mathcal{G}_h^{\hat{\delta}}(v)$, $T^\pi T^{h-1} v \geq T^h v - \hat{\delta}$.*

Additionally, the algorithms assume additive $\hat{\epsilon} \in \mathbb{R}^{|\mathcal{S}|}$ error in the evaluation stage. We call them hm-PI and $h\lambda$-PI and present them in Algorithms 2 and 3. As opposed to the non-contracting update discussed in Section 5, the evaluation stage in these algorithms uses $T^{h-1}v$.

We now provide our main result, demonstrating a γ^h-contraction coefficient for both hm-PI and $h\lambda$-PI.

The proof technique builds upon the previously introduced *invariance argument* (e.g., (Efroni et al. 2018a)[Theorem 9]). This enables working with a more convenient, shifted noise sequence. Thereby, we construct a shifted noise sequence s.t. the value-policy pair (v_k, π_{k+1}) in each iteration is h-greedy consistent (see Definition 1). We thus also eliminate the h-greedy consistency assumption on the initial (v_0, π_1) pair, which appears in previous works (see Remark 2). Specifically, we shift v_0 by Δ_0; the latter quantifies how 'far' (v_0, π_1) is from being h-greedy consistent. Notice our bound explicitly depends on Δ_0. The provided proof is simpler and shorter than in previous works (e.g. (Scherrer 2013)). We believe that the proof technique presented here can be used as a general 'recipe' for proving newly-devised PI procedures that use partial evaluation with more ease.

Theorem 4. *Let $h, m \in \mathbb{N} \setminus \{0\}, \lambda \in [0,1]$. For noise sequences $\{\epsilon_k\}$ and $\{\delta_k\}$, $\|\epsilon_k\|_\infty \leq \epsilon$ and $\|\delta_k\|_\infty \leq \delta$. Let*

$$\Delta_0 = \max\{0, \frac{\max_s \left(T^{h-1}v_0 - T^{\pi_1}T^{h-1}v_0\right)(s)}{\gamma^{h-1}(1-\gamma)}\}.$$

Then,

$$\|v^* - v^{\pi_{k+1}}\|_\infty \leq \gamma^{kh}||v^* - (v_0 - \Delta_0)||_\infty + \frac{(2\gamma^h\epsilon + \delta)(1-\gamma^{kh})}{(1-\gamma)(1-\gamma^h)}$$

and hence $\limsup_{k\to\infty} \|v^* - v^{\pi_k}\|_\infty \leq \frac{2\gamma^h\epsilon+\delta}{(1-\gamma)(1-\gamma^h)}$.

Proof. We start with the invariance argument. Consider the process with the alternative error in the evaluation stage, $\epsilon'_k = \epsilon_k - C_k e$, where $C_k = \frac{\max \delta_{k+1} + \gamma^{h-1}\max \epsilon_k - \gamma^h \min \epsilon_k}{\gamma^{h-1}(1-\gamma)}$, and e a vector of 'ones' of dimension $|\mathcal{S}|$. Next, given initial value v_0, let $v'_0 = v_0 - \Delta_0$. As described in Remark 2, this transformation makes (v'_0, π_1) h-greedy consistent. Since the greedy policy is invariant for an addition of a constant, i.e., for $\alpha \in \mathbb{R}$, $\mathcal{G}_h(v + \alpha e) = \mathcal{G}_h(v)$, and since $T^h(v + \alpha e) = T^h v + \gamma^h \alpha$, we have that the *sequence of policies generated is invariant* for the offered transformation.

Next, we use (Efroni et al. 2018c)[Lemma 6], which gives that the choice of C_k leads to a sequence of pairs of h-greedy consistent policies and values in every iteration. We now continue with a simpler analysis than in (Efroni et al. 2018a).

At this stage of the proof we focus on hm-PI. Define $d_k \stackrel{\text{def}}{=} v^* - (v'_k - \epsilon'_k)$ for $k \geq 1$, and $d_0 \stackrel{\text{def}}{=} v^* - v'_0$. We get

$$\begin{aligned}
d_{k+1} &= v^* - (T^{\pi_{k+1}})^m T^{h-1} v'_k && (11)\\
&\leq v^* - T^{\pi_{k+1}} T^{h-1} v'_k && (12)\\
&\leq v^* - T^h v'_k + \max \delta_{k+1}\\
&\leq (T^{\pi_*})^h v^* - T^h(v'_k - \epsilon'_k) - \gamma^h \min \epsilon'_k + \max \delta_{k+1}\\
&\leq (T^{\pi_*})^h v^* - (T^{\pi_*})^h(v'_k - \epsilon'_k) - \gamma^h \min \epsilon'_k + \max \delta_{k+1}\\
&= \gamma^h (P^{\pi_*})^h (v^* - (v'_k - \epsilon'_k)) - \gamma^h \min \epsilon'_k + \max \delta_{k+1}\\
&= \gamma^h (P^{\pi_*})^h d_k - \gamma^h \min \epsilon'_k + \max \delta_{k+1}.
\end{aligned}$$

The second relation holds by applying Lemma 1 on π_{k+1} and v_k which are h-consistent. Furthermore, by using the form of C_k and simple algebraic manipulations it can be shown that $-\gamma^h \min \epsilon'_k + \max \delta_{k+1} \leq \frac{2\gamma^h\epsilon+\delta}{1-\gamma}$. Thus,

$$d_{k+1} \leq \gamma^h (P^{\pi_*})^h d_k + \frac{2\gamma^h\epsilon + \delta}{1-\gamma}. \quad (13)$$

Iteratively applying the above relation on k, we get that

$$\begin{aligned}
d_k &\leq \gamma^{kh}(P^{\pi_*})^{kh} d_0 + \frac{(2\gamma^h\epsilon+\delta)(1-\gamma^{kh})}{(1-\gamma)(1-\gamma^h)}\\
&\leq \gamma^{kh}||d_0||_\infty + \frac{(2\gamma^h\epsilon+\delta)(1-\gamma^{kh})}{(1-\gamma)(1-\gamma^h)}. \quad (14)
\end{aligned}$$

To conclude the proof for hm-PI notice that $v^* - v^{\pi_{k+1}} \leq v^* - (v'_k - \epsilon'_k) = d_k$, which holds due to the second claim in (Efroni et al. 2018c)[Lemma 6] combined with Lemma 1. Since the LHS is positive, we can apply the max norm on the inequality and use (14):

$$||v^* - v^{\pi_{k+1}}||_\infty \leq \gamma^{kh}||d_0||_\infty + \frac{(2\gamma^h\epsilon+\delta)(1-\gamma^{kh})}{(1-\gamma)(1-\gamma^h)}.$$

Since $d_0 = v^* - v'_0 = v^* - (v_0 - \Delta_0)$, we obtain the first claim for hm-PI. Taking the limit easily gives the second claim, again for hm-PI:

$$\lim_{k\to\infty} ||v^* - v^{\pi_{k+1}}||_\infty \leq \frac{2\gamma^h\epsilon+\delta}{(1-\gamma)(1-\gamma^h)}.$$

The convergence proof for $h\lambda$-PI is identical to that of hm-PI, except for a minor change: the transition from (11)

to (12) holds due to the following argument:

$$\begin{aligned} d_{k+1} &\leq v^* - (1-\lambda)\sum_i \lambda^i (T^{\pi_{k+1}})^{i+1} T^{h-1} v'_k \\ &\leq v^* - (1-\lambda)\sum_i \lambda^i T^{\pi_{k+1}} T^{h-1} v'_k \\ &= v^* - T^{\pi_{k+1}} T^{h-1} v'_k, \end{aligned}$$

where the second relation holds by applying Lemma 1. This can be used since π_{k+1} and v'_k are h-greedy consistent according to (Efroni et al. 2018c)[Lemma 6]. This exemplifies the advantage of using the notion of h-greedy consistency in our proof technique. □

Thanks to using $T^{h-1}v$ in the evaluation stage, Theorem 4 guarantees a convergence rate of γ^h – as to be expected when using a T^h greedy operator. Compared to directly using v as is done in Section 5, this is a significant improvement since the latter does not even necessarily contract.

A possibly more 'natural' version of our algorithms would back up the value of the root node instead of its descendants. The following remark extends on that.

Remark 3. *Consider a variant of hm-PI and $h\lambda$-PI, which backs-up $T^{\pi_{k+1}} T^{h-1} v_k$ instead of $T^{h-1} v_k$. Namely, in this variant, the evaluation stage for hm-PI (Algorithm 2) is*

$$v_{k+1} \leftarrow (T^{\pi_{k+1}})^{m-1} (T^{\pi_{k+1}} T^{h-1} v_k) + \epsilon_k,$$

and for $h\lambda$-PI (Algorithm 3) it is

$$v_{k+1} \leftarrow \bar{T}_\lambda^{\pi_{k+1}} (T^{\pi_{k+1}} T^{h-1} v_k) + \epsilon_k.$$

The latter is (see proof in (Efroni et al. 2018b)) $\bar{T}_\lambda^\pi v \stackrel{def}{=} (1-\lambda)\sum_{j=0}^\infty \lambda^j (T^\pi)^j v = v + \lambda(I - \gamma\lambda P^\pi)^{-1}(T^\pi v - v)$ – a variation of the λ-return operator from (4)*, in which T^π is raised to the power of j and not $j+1$. The performance of these algorithms is equivalent to that of the original hm-PI and $h\lambda$-PI, as given in Theorem 4, since*

$$(T^\pi)^{m-1} T^\pi = (T^\pi)^m \text{ and } \bar{T}_\lambda^\pi T^\pi = T_\lambda^\pi.$$

Yet, implementing them is potentially easier in practice, and can be considered more 'natural' due to the backup of the root optimal value rather its descendants.

7 Relation to Existing Work

In the context of related theoretical work, we find two results necessitating a discussion. The first is the performance bound of Non-Stationary Approximate Modified PI (NS-AMPI) (Lesner and Scherrer 2015)[Theorem 3]. Compared to it, Theorem 4 reveals two improvements. First, it gives that hm-PI is less sensitive to errors; our bound's numerator has γ^h instead of γ. Second, in each iteration, hm-PI requires storing a single policy in lieu of h policies as in NS-AMPI. This makes hm-PI significantly more memory efficient. Nonetheless, there is a caveat in our work compared to (Lesner and Scherrer 2015). In each iteration, we require to approximately solve an h-finite-horizon problem, while they require solving approximate 1-step greedy problem instances.

The second relevant theoretical result is the performance bound of a recently introduced MCTS-based RL algorithm (Jiang, Ekwedike, and Liu 2018)[Theorem 1]. There, in the noiseless case there is no guarantee for convergence to the optimal policy[1]. Contrarily, in our setup, with $\delta = 0$ and $\epsilon = 0$ both hm-PI and $h\lambda$-PI converge to the optimal policy.

Next, we discuss related literature on empirical studies and attempt to explain observations there with the results of this work. In (Baxter, Tridgell, and Weaver 1999; Veness et al. 2009; Lanctot et al. 2014) the idea of incorporating the optimal value from the tree-search was experimented with. Most closely related to our synchronous setup is that in (Baxter, Tridgell, and Weaver 1999). There, motivated by practical reasons, the authors introduced and evaluated both NC $h\lambda$-PI and $h\lambda$-PI, which they respectively call TD-directed(λ) and TDLeaf(λ). Specifically, TD-directed(λ) and TDLeaf(λ) respectively back up v and $T^{\pi_h} T^{h-1} v$. As Remark 1 suggests, $T^{\pi_h} T^{h-1} v$ can be extracted directly from the tree-search, as is also pointed out in (Baxter, Tridgell, and Weaver 1999). Interestingly, the authors show that TDLeaf(λ) outperforms TD-directed(λ). Indeed, Theorem 3 sheds light on this phenomenon.

Lastly, a prominent takeaway message from Theorems 3 and 4 is that AlphaGoZero (Silver et al. 2017b; 2017a) can be potentially improved. This is because in (Silver et al. 2017b), the authors do not back up the optimal value calculated from the tree search. As their approach relies on PI (and specifically resembles to h-PI), our analysis, which covers noisy partial evaluation, can be beneficial even in the practical setup of AlphaGoZero.

8 Experiments

In this section, we empirically study NC-hm-PI (Section 5) and hm-PI (Section 6) in the exact and approximate cases. Additional results are found in (Efroni et al. 2018c). Our experiments demonstrate the practicalities of Theorem 3 and 4, even in the simple setup considered here.

We conducted our simulations on a simple $N \times N$ deterministic grid-world problem with $\gamma = 0.97$, as was done in (Efroni et al. 2018a). The action set is {'up','down','right','left','stay'}. In each experiment, a reward $r_g = 1$ was placed in a random state while in all other states the reward was drawn uniformly from $[-0.1, 0.1]$. In the considered problem there is no terminal state. Also, the entries of the initial value function are drawn from $\mathcal{N}(0,1)$. We ran the algorithms and counted the *total* number of calls to the simulator. Each such "call" takes a state-action pair (s,a) as input, and returns the current reward and next (deterministic) state. Thus, it quantifies the total running time of the algorithm, and not the total number of iterations.

We begin with the noiseless case, in which ϵ_k and δ_k from Algorithm 2 are 0. While varying h and m, we counted the total number of queries to the simulator until convergence, which defined as $||v^* - v_k||_\infty \leq 10^{-7}$. Figure 3 exhibits the results. In its top row, the heatmaps give the convergence

[1]The bound in (Jiang, Ekwedike, and Liu 2018)[Theorem 1] is not necessarily 0 for $\epsilon = 0$ since $B_\gamma, B'_\gamma, \mathbb{D}_0$ and $\mathbb{D}_1$ do not depend on the error and, generally, are not 0.

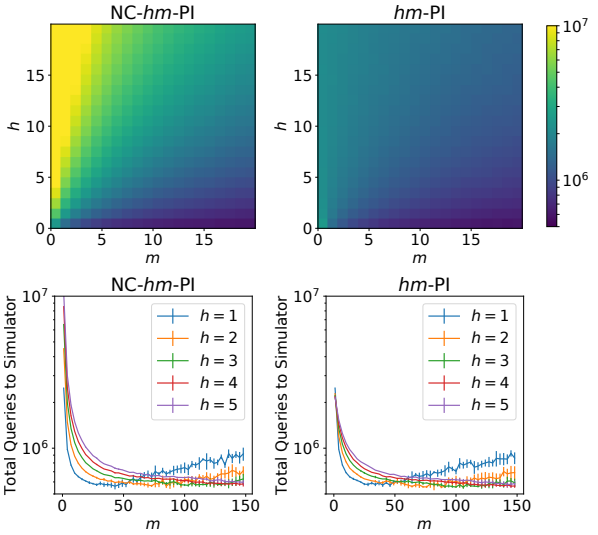

Figure 3: (Top) Noiseless NC-hm-PI and hm-PI convergence time as function of h and m. (Bottom) Noiseless NC-hm-PI and hm-PI convergence time as function of a wide range of m, for several values of h. In both figures, the standard error is less than $\%2$ of the mean.

time for equal ranges of h and m. It highlights the suboptimality of NC-hm-PI compared to hm-PI. As expected, for $h = 1$, the results coincide for NC-hm-PI and hm-PI since the two algorithms are then equivalent. For $h > 1$, the performance of NC-hm-PI significantly deteriorates up to an order of magnitude compared to hm-PI. However, the gap between the two becomes less significant as m increases. This can be explained with Theorem 3: increasing m in NC-hm-PI drastically shrinks γ^m in (8) and brings the contraction coefficient closer to γ^h, which is that of hm-PI. In the limit $m \to \infty$ both algorithms become h-PI.

The bottom row in Figure 3 depicts the convergence time in 1-d plots for several small values of h and a large range of m. It highlights the tradeoff in choosing m. As h increases, the optimal choice of m increases as well. Further rigorous analysis of this tradeoff in m versus h is an intriguing subject for future work.

Next, we tested the performance of NC-hm-PI and hm-PI in the presence of evaluation noise. Specifically, $\forall k, s \in \mathcal{S}$, $\epsilon_k(s) \sim U(-0.3, 0.3)$ and $\delta_k(s) = 0$. For NC-hm-PI, the noise was added according to $v_{k+1} \leftarrow (T^{\pi_k})^m v_k + \epsilon_k$ instead of the update in the first equation in (7). The value $\delta_k = 0$ corresponds to having access to the exact model. Generally, one could leverage the model for a complete immediate solution instead of using Algorithm 2, but here we consider cases where this cannot be done due to, e.g., too large of a state-space. In this case, we can approximately estimate the value and use a multiple-step greedy operator with access to the exact model. Indeed, this setup is conceptually similar to that taken in AlphaGoZero (Silver et al. 2017b). Figure 4 exhibits the results. The heatmap values are $||v^* - v^{\pi_f}||_\infty$, where π_f is the algorithms' output policy after $4 \cdot 10^6$ queries to the simulator.

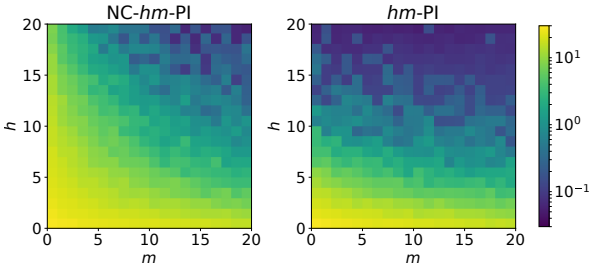

Figure 4: Distance from optimum (lower is better) for NC-hm-PI and hm-PI in the presence of evaluation noise. The heatmap values are $||v^* - v^{\pi_f}||_\infty$, where π_f is the algorithms' output policy after $4 \cdot 10^6$ queries to the simulator.

Both NC-hm-PI and hm-PI converge to a better value as h increases. However, this effect is stronger in the latter compared to the former, especially for small values of m. This demonstrates how hm-PI is less sensitive to approximation error. This behavior corresponds to the hm-PI error bound in Theorem 4, which decreases as h increases.

9 Summary and Future Work

In this work, we formulated, analyzed and tested two approaches for relaxing the evaluation stage of h-PI – a multiple-step greedy PI scheme. The first approach backs up v and the second backs up $T^{h-1}v$ or $T^{\pi_h}T^{h-1}v$ (see Remark 3). Although the first might seem like the natural choice, we showed it performs significantly worse than the second, especially when combined with short-horizon evaluation, i.e., small m or λ. Thus, due to the intimate relation between h-PI and state-of-the-art RL algorithms (e.g., (Silver et al. 2017b)), we believe the consequences of the presented results could lead to better algorithms in the future.

Although we established the non-contracting nature of the algorithms in Section 5, we did not prove they would necessarily not converge. We believe that further analysis of the non-contracting algorithms is intriguing, especially given their empirical converging behavior in the noiseless case (see Section 8, Figure 3). Understanding when the non-contracting algorithms perform well is of value, since their update rules are much simpler and easier to implement than the contracting ones.

To summarize, this work highlights yet another difference between 1-step based and multiple-step based PI methods, on top of the ones presented in (Efroni et al. 2018a; 2018c). Namely, multiple-step based methods introduce a new degree of freedom in algorithm design: the utilization of the planning byproducts. We believe that revealing additional such differences and quantifying their pros and cons is both intriguing and can have meaningful algorithmic consequences.

References

Baxter, J.; Tridgell, A.; and Weaver, L. 1999. Tdleaf (lambda): Combining temporal difference learning with game-tree search. *arXiv preprint cs/9901001*.

Bertsekas, D. P., and Ioffe, S. 1996. Temporal differences-based policy iteration and applications in neuro-dynamic programming.

Bertsekas, D. P., and Tsitsiklis, J. N. 1995. Neuro-dynamic programming: an overview. In *Decision and Control, 1995., Proceedings of the 34th IEEE Conference on*, volume 1. IEEE.

Bertsekas, D. P. 2011. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications* 9(3):310–335.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4(1):1–43.

Efroni, Y.; Dalal, G.; Scherrer, B.; and Mannor, S. 2018a. Beyond the one-step greedy approach in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, 1386–1395.

Efroni, Y.; Dalal, G.; Scherrer, B.; and Mannor, S. 2018b. How to combine tree-search methods in reinforcement learning. *arXiv preprint arXiv:1809.01843*.

Efroni, Y.; Dalal, G.; Scherrer, B.; and Mannor, S. 2018c. Multiple-step greedy policies in online and approximate reinforcement learning. *arXiv preprint arXiv:1805.07956*.

Jiang, D.; Ekwedike, E.; and Liu, H. 2018. Feedback-based tree search for reinforcement learning. In Dy, J., and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 2284–2293. Stockholmsmässan, Stockholm Sweden: PMLR.

Jin, C.; Allen-Zhu, Z.; Bubeck, S.; and Jordan, M. I. 2018. Is q-learning provably efficient? *arXiv preprint arXiv:1807.03765*.

Lai, M. 2015. Giraffe: Using deep reinforcement learning to play chess. *arXiv preprint arXiv:1509.01549*.

Lanctot, M.; Winands, M. H.; Pepels, T.; and Sturtevant, N. R. 2014. Monte carlo tree search with heuristic evaluations using implicit minimax backups. *arXiv preprint arXiv:1406.0486*.

Lesner, B., and Scherrer, B. 2015. Non-stationary approximate modified policy iteration. In *International Conference on Machine Learning*, 1567–1575.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Munos, R. 2014. From bandits to monte-carlo tree search: The optimistic principle applied to optimization and planning. Technical report. 130 pages.

Negenborn, R. R.; De Schutter, B.; Wiering, M. A.; and Hellendoorn, H. 2005. Learning-based model predictive control for markov decision processes. *Delft Center for Systems and Control Technical Report 04-021*.

Puterman, M. L., and Shin, M. C. 1978. Modified policy iteration algorithms for discounted markov decision problems. *Management Science* 24(11):1127–1137.

Puterman, M. L. 1994. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons.

Scherrer, B. 2013. Performance Bounds for Lambda Policy Iteration and Application to the Game of Tetris. *Journal of Machine Learning Research* 14:1175–1221.

Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. 2017a. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. 2017b. Mastering the game of go without human knowledge. *Nature* 550(7676):354.

Sutton, R. S.; Barto, A. G.; et al. 1998. *Reinforcement learning: An introduction*.

Tamar, A.; Thomas, G.; Zhang, T.; Levine, S.; and Abbeel, P. 2017. Learning from the hindsight plan–episodic mpc improvement. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 336–343. IEEE.

Veness, J.; Silver, D.; Blair, A.; and Uther, W. 2009. Bootstrapping from game tree search. In *Advances in neural information processing systems*, 1937–1945.