How to Combine Tree-Search Methods in Reinforcement Learning? A Reproductibilty Study

Alexandre Morinvil 1897222, Maude Nguyen-The 1843896 & Adam Prévost 1947205
Department of Computer and Software engineering
Polytecnique Montréal University
Montréal, H3T 1J4, CAN
{alexandre.morinvil, maude.nguyen-the, adam.prevost}@polymtl.ca

1 Introduction and Problem definition

This paper aims at validating the results of "How to Combine Tree-Search Methods in Reinforcement Learning?" (Efroni et al., 2019), a paper by Yonathan Efroni, Gal Dalal, Bruno Scherrer and Shie Mannor. The paper is mainly a theorical contribution to the field and many of their mathematically proven results are only backed by a few experimental results. We will be reproducing their experiments and expanding upon them to validate more of their theoretical results. We also wish to compare these results between them to get new insights on the problem at hand.

In reinforcement learning, finite-horizon lookahead policies are abundantly used. Monte Carlo Tree Search is a good example of a specific planning method. Referring to the planning problem as a tree search is reasonable. A good approach is to only store the values at the leaves (at the horizon, at the last lookahead step) and use this information to iteratively climb back up the tree and finally update the policy. This approach, however, is non-contractive in general.

The authors of "How to Combine Tree-Search Methods in Reinforcement Learning?" (Efroni et al., 2019) tackle this problem by expanding upon standard policy iteration, specifically starting from h-greedy policies. They come up with two variants, hm-PI and hλ-PI, and they introduce a notion called *multiple-step greedy consistency* to prove the contraction of their new policy iteration algorithms.

2 Background and Motivation

2.1 Multiple-step greedy consistency

It is already known that the h-greedy policy with regard to v^π is strictly better than π (Bertsekas & Tsitsiklis, 1995), which allows us to formulate the following equation :

$$v^{\pi_h} \geq v^\pi \tag{1}$$

This property can be used to prove the convergence of algorithms by proving the above mentioned equation holds true as the policies and values evolve through a given algorithm, however, the caveat is that to make use of this property, we would be required to obtain the exact value estimation, which can be a very challenging endeavor. Therefore, deriving from the above postulate, the article introduces the concept of "h-greedy consistency" to study the convergence of algorithms while using partial evaluations of value functions. A pair of value function and policy (v, π) is h-greedy consistent if :

$$T^\pi T^{h-1} v \geq T^{h-1} v \tag{2}$$

An intuition to understand the h-greedy consistency is that a pair of value function and policy (v, π) is h-greedy consistent if the policy π improves $T^{h-1}v$.

The two partial evaluation functions considered in the article are the "Modified Policy Iteration" (m-PI : $(T^{\pi_k})^m v_k$) and the "Lambda policy iteration" ($\lambda_P I$: $T_\lambda^{\pi_k} v_k$).

2.2 Proposed algorithms

The article uses the h-PI procedure which is a policy iteration procedure derived from h-greedy policy. This algorithm is simply the standard policy iteration algorithm, but using h-lookahead rather than 1-step greedy. The policy update is made using this equation:

$$\pi_k \leftarrow \arg\max_{\pi} T^{\pi} T^{h-1} v_k \tag{3}$$

where T^π is the expected Bellman operator, such that $T^\pi v = R^\pi + \gamma P^\pi v$, and T^{h-1} is the Bellman optimality operator, which takes the maximum over the actions in the Bellman operator, repeated $h-1$ times. This algorithm comes from the results of Bertsekas & Tsitsiklis (1995) and Efroni et al. (2018c). Through this algorithm, we can then perform an exact value estimation.

Algorithm 1: *h*-PI

```
initialize: $h \in \mathbb{N} \setminus \{0\}, v_0 = v^{\pi_0} \in \mathbb{R}^{|S|}$
while $v_k$ changes do
    $\pi_k \leftarrow \pi \in G_h(v)$
    $v_{k+1} \leftarrow v^{\pi_k}$
    $k \leftarrow k + 1$
end
return $\pi, v$
```

The article then proposes two algorithm that perform a partial value evaluation. Both new algorithms are based on *h*-PI.

The first algorithm that the authors propose is a new version of the Modified-PI algorithm first proposed by Puterman & Shin (1978). The original algorithm uses the m-return operator to do a partial evaluation of the policy at each iteration so that $v_{k+1} \leftarrow (T^{\pi_k})^m v_k$. This value evaluation, however, does not guarantee the contractiveness of the algorithm, therefore the version of the algorithm which directly uses this value evaluation method is referred to as non-contractive hm-PI (NC-hm-PI).

The proposed hm-PI algorithm uses the h-1 steps backed-up optimal values to use the m-return operator on. The new update for the state value estimate is then $v_{k+1} \leftarrow (T^{\pi_k})^m T^{h-1} v_k$. This decision ensures the contractivenes of the algorithm with the proof provided in the article. This new method is presented in Algorithm 2.

Algorithm 2: *hm*-PI

```
initialize: $h, m \in \mathbb{N} \setminus \{0\}, v \in \mathbb{R}^{|S|}$
while stopping criterion is false do
    $\pi_{k+1} \leftarrow \pi \in G_h^{\delta_{k+1}}(v_k)$
    $v_{k+1} \leftarrow (T^{\pi_{k+1}})^m T^{h-1} v_k + \epsilon_k$
    $k \leftarrow k + 1$
end
return $\pi, v$
```

The second algorithm that the authors propose is uses λ-PI, which was first introduced by Bertsekas & Ioffe (1996), for the partial evaluation. The value function estimate is updated using the λ-return operator, noted T_λ^π. This operator can be computed using the following equation.

$$T_\lambda^\pi v = v + (I - \gamma\lambda P^\pi)^{-1}(T^\pi v - v) \tag{4}$$

As with the new version of Modified-PI, in the new hλ-PI algorithm proposed, the update for the value function is computed using the backed-up optimal value from the h-1 planning steps such that $v_{k+1} \leftarrow T^{\pi}_{\lambda} T^{h-1} v_k$ to ensure contractiveness. The version that uses directly the backed up value functions for the evaluation would thus be considered the non-contractive version. This new algorithm is also robust to noisy updates, which can be modelled by adding an error term ϵ_k to the update of v_k. Algorithm 3 presents this new method.

Algorithm 3: $h\lambda$-PI

```
initialize: $h, m \in \mathbb{N} \setminus \{0\}, \lambda \in [0,1], v \in \mathbb{R}^{|S|}$
while stopping criterion is false do
    $\pi_{k+1} \leftarrow \pi \in G_h^{\delta_{k+1}}(v_k)$
    $v_{k+1} \leftarrow T_{\lambda}^{\pi_{k+1}} T^{h-1} v_k + \epsilon_k$
    $k \leftarrow k + 1$
end
return $\pi, v$
```

Thus, the work completed in this project uses the three algorithms presented above: h-PI, hm-PI and hλ-PI and the non contractive form of the two latter ones, NC-hm-PI and NC-hλ-PI.

2.3 THE EXPERIMENTAL METHOD

The authors used the following environment from Efroni et al. (2018a) and so will we. The simulations are conducted on a simple $n \times n$ deterministic grid-world with a discount factor of 0.97. The possible actions are {'up', 'down', 'right', 'left', 'stay'}. In each experiment, a random state has a reward of 1 and all other states have rewards drawn uniformly from $[-0.1, 0.1]$. There is no terminal state, the optimal policy should bring the agent to the state with reward 1 and stay there. The initial state values are drawn from a Gaussian distribution, with mean 0 and variance 1. Policy iteration convergence is defined as $||v^* - v_k||_\infty \leq 10^{-7}$.

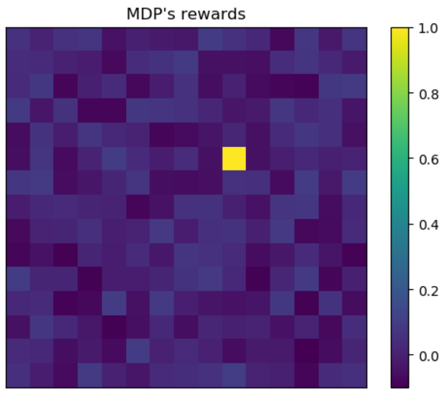
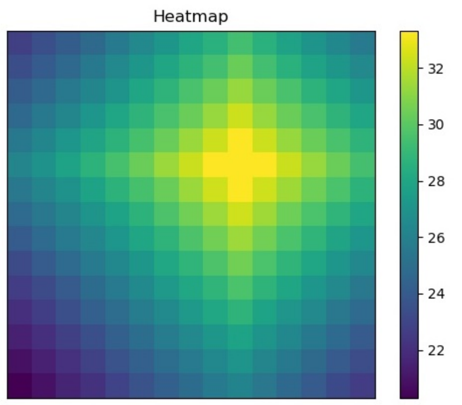
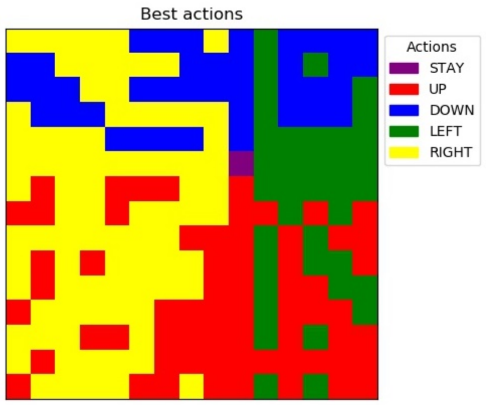


Figure 1: MDP Environment rewards example on a 15 by 15 grid

The code we developed also includes tools used to visualize and represent the solutions obtained. The state values and best actions were represented visually in order to quickly validate the accuracy of the different algorithms while they were being re-implemented.



(a) Visual representation of the MDP environment state value determined by the h-PI algorithm



(b) Visual representation of the MDP environment's best actions determined by the h-PI algorithm

Figure 2: Visualization tools developed to study the algorithms

Performances are assessed through the total number of calls to the simulator. Each call takes a state-action pair and returns the current reward and the next state. It quantifies the total running time of the algorithm, and not the total number of iterations. For mathematical and time sake though, all calculations are done through linear algebra, thus the total number is increased manually with the amount corresponding to each operation. This correspondence is actually not specified in the paper. Different interpretations could lead to different numbers of calls, especially when taking into account that *in practice* policy iteration processes reuse data, such as the reward matrix, which can be stored to avoid calling the simulator again.

Furthermore, the value function obtained after a maximum of 10^6 queries was compared to the optimal value function v^*. The exact optimal value function was determined by running the h-PI algorithm on the Markov decision Process environment.

3 Related works

The parameters of the environment were entirely defined in a previous paper by the same authors (Efroni et al., 2018b). The proofs described here are also based on and use many arguments introduced in the aforementioned paper. In many ways, the original paper is a continuation of the previous works of the authors. However, they were not the first to come up with these *new* algorithms. The idea for hm-PI and hλ-PI are discussed as far back as in (Baxter et al., 1999) under different names, although not proven and empirically tested in the same way.

"The Role of Lookahead and Approximate Policy Evaluation in Policy Iteration with Linear Value Function Approximation" (Winnicki et al., 2021) heavily builds upon the proofs and mathematical techniques introduced in the original paper. It tackles the same problem, but in the case of very large MDPs, where these techniques become too computationally expensive. However, this paper does not show empirical results and is mainly theorical in nature, like the original paper. The original paper is mentioned elsewhere too, such as in Springenberg et al. (2020), but always only very briefly and not in any very meaningful way. As far as we know, our work is the first and only one reproducing and/or comparing the original paper's empirical results.

4 Detailed discussion of the Research/Analysis Questions studied

There were two main analysis objectives in the context of the work completed for this project. The first objective is to assesss the reproducibility of the results obtained in the article by performing the same experiments presented in the article with the hm-PI algorithm. The second objective is to pursue the analysis by performing equivalent experiments with the hλ-PI (which wasn't done in the article) and compare the results with the hm-PI algorithm.

4.1 Can we reproduce the empirical results of NC-hm-PI and hm-PI demonstrated in their experiments, using the same hyperparameters?

The first experiment done in the article consisted in comparing the number of queries for the NC-hm-PI and hm-PI algorithms for different values of h and m. The experimental total queries results obtained from the article are displayed below.

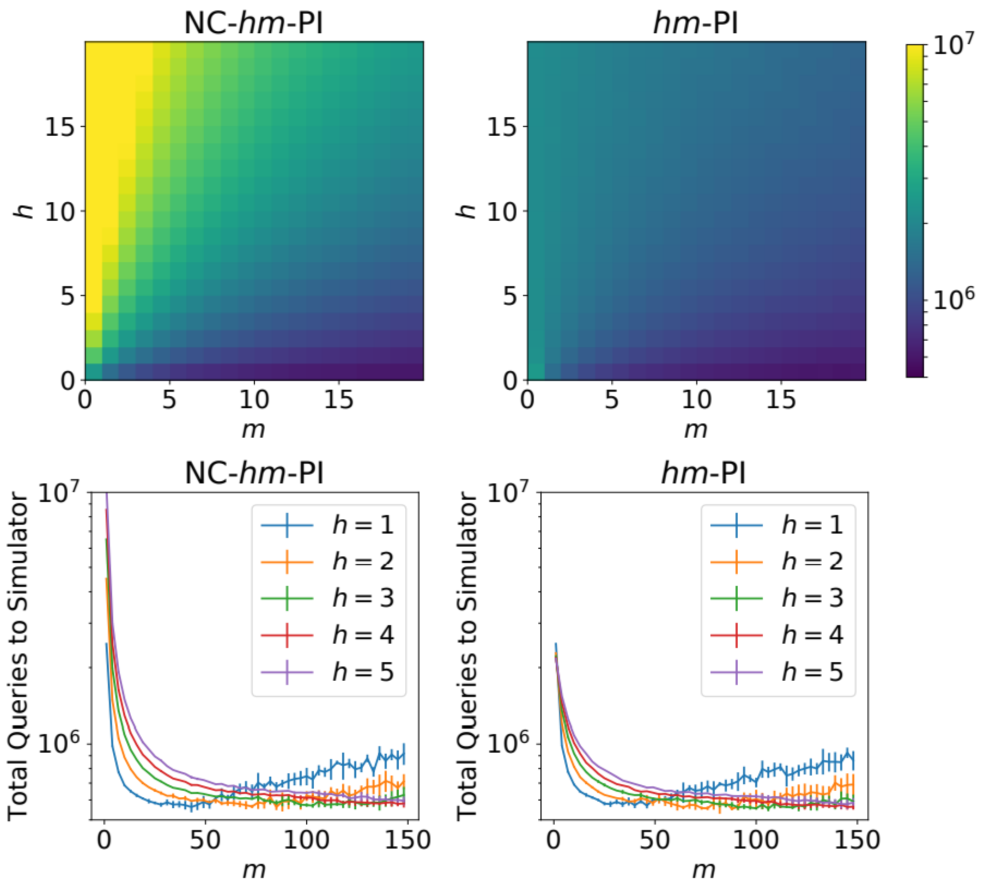


Figure 3: Results taken directly from the article : (Top) Noiseless NC-hm-PI and hm-PI convergence time as function of h and m. (Bottom) Noiseless NC-hm-PI and hm-PI convergence time as function of a wide range of m, for several values of h. In both figures, the standard error is less than 2% of the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

The more yellow area represents a combination of h, m hyper-parameters that have the worse performance (higher number of queries, which means that the algorithm takes longer). The article also highlights that for $h > 1$, the performance of NC-hm-PI deteriorates up to an order of magnitude compared to hm-PI and that as m increases, the the difference between hm-PI and NC-hm-PI decreases. Moreover, we can observe that, overall, the higher the m, the better the performance becomes.

The experiments were reproduced and the homologous graphs are displayed in the figure below.

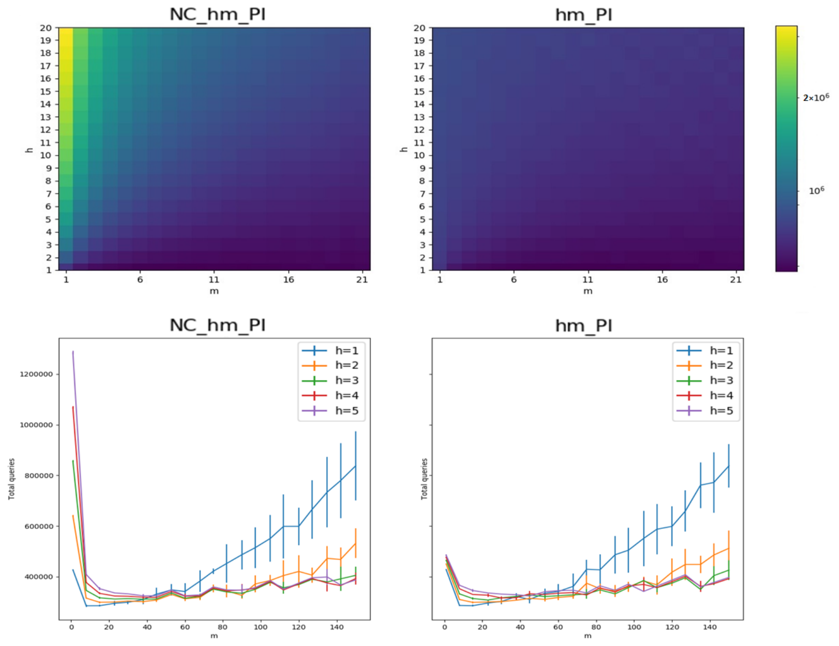


Figure 4: Experimental convergence times : (Top) Noiseless NC-hm-PI and hm-PI convergence time as function of h and m. (Bottom) Noiseless NC-hm-PI and hm-PI convergence time as function of a wide range of m, for several values of h.

We also compared the distance to the optimal value function obtained with the algorithms for a maximum number of queries of $4 \cdot 10^6$ and with the introduction of a 0.3 noise in the value iteration, as was performed in the original paper. The results of the article and our experimental results are presented in the figures below.

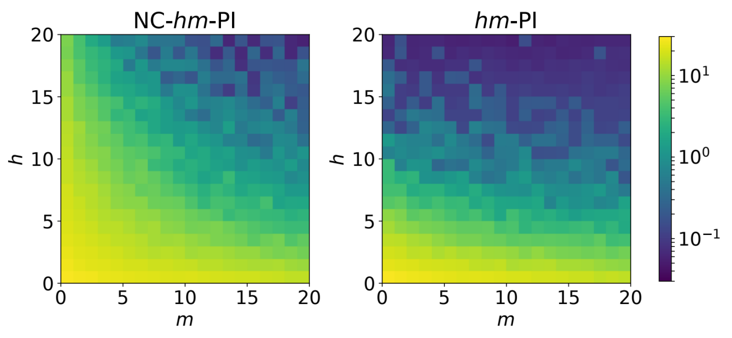


Figure 5: Distance from optimum obtained in the article for the NC-hm-PI and hm-PI algorithms

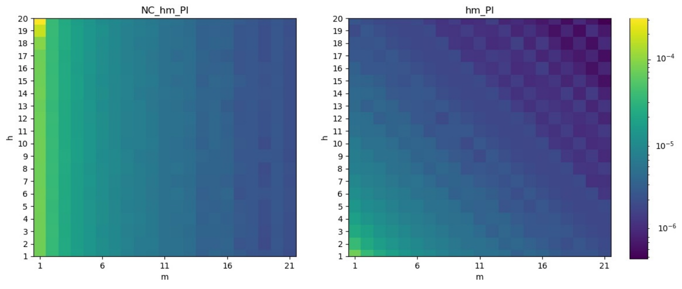


Figure 6: Distance from optimum obtained experimentally for the NC-hm-PI and hm-PI algorithms

From those figures, we observe that the general aspect of our results seems to be relatively consistent to the results obtained in the article. For a combination of a higher h and m, the value function obtained is much closer to the optimal value (as shown by the bluer tone at the upper right corner). Using the scale, we observe that the differences are much higher in the paper than in our results by some significant order of magnitude. This difference might be caused by the size of the grid we used, the way the authors compute the reference value v^* and also by a difference in the way the number of queries is calculated which would make it so we wouldn't reach the maximum number of queries that we set before having converged.

4.2 Do the same results hold true for NC-hλ-PI and hλ-PI?

As the authors do not provide any visualization of the performances of NC-hλ-PI and hλ-PI in the paper, we tested these two algorithms using the same MDP conditions and discount factor as with NC-hm-PI and hm-PI. The convergence times obtained depending on the hyperparameters h and λ are presented in Figure 7.

As was mentioned in the article and in a similar way to hm-PI, when values of λ are small and the value of h grows, NC-hλ-PI performs significantly worse than hλ-PI, whereas when λ values get closer to 1, both algorithms obtain comparable performances.

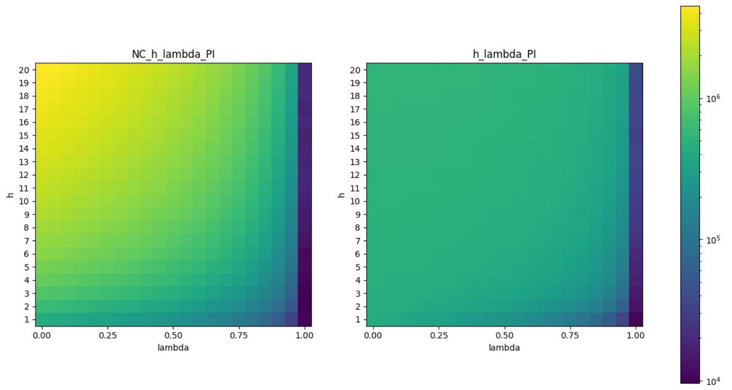


Figure 7: Experimental convergence time for hλ-PI algorithms

The 1-dimensional curves for small values of h shown in Figure 9 also demonstrate that as h gets bigger, there is a remarkable difference between NC-hλ-PI and hλ-PI, especially for small λ values.

Finally the distance to the optimal state values when noise is added to the policy evaluation is shown in Figure 9. As with hm-PI and NC-hm-PI, hλ-PI converges to better values when h grows, and systematically obtains better results than NC-hλ-PI, except for when h=1 or λ=1. In the case of λ=1, both algorithm become equivalent to doing the full policy evaluation, which is why there is little to no difference from the optimal v^* values which we computed using h-PI.

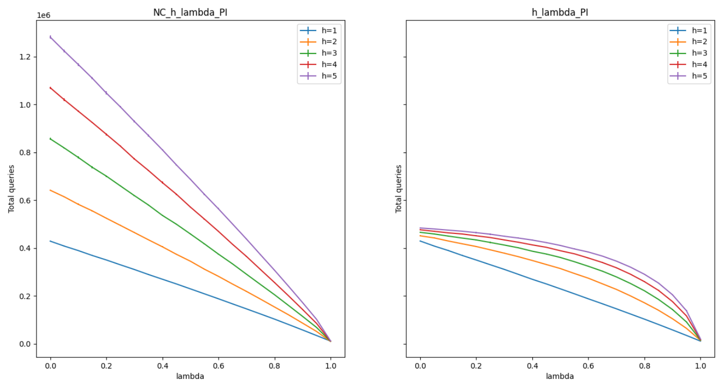


Figure 8: Experimental convergence curve for hλ-PI algorithms

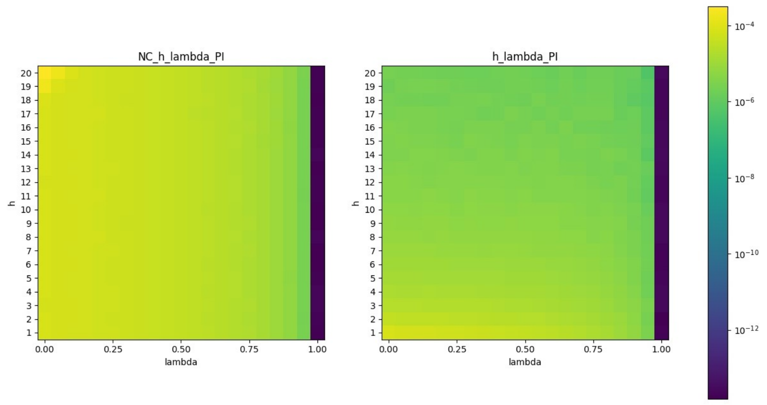


Figure 9: Experimental distance to optimum for noisy updates with hλ-PI algorithms

4.3 Which algorithm converges faster?

It is difficult to directly compare the convergence rate of hλ-PI and hm-PI, since the number of calls to the simulator to compute the λ-return is not well defined for continuing tasks. To be able to directly compare our results for hm-PI to those presented in the paper, we kept the same MDP which had no terminal state. The λ-return was computed using the algebraic operator shown in Equation 4, where the simulator only needed to be called once for every (state, action) pair to get the rewards matrix. More experiments should be done using episodic tasks to compare the time of convergence of hλ-PI and hm-PI using their version which sums over the returns.

5 Conclusion

In conclusion, we have reproduced the experiments done on hm-PI and explored hλ-PI. Some empirical results are similar, such as the total number of queries, and some are different, such as the distance from the optimal state values. However, even in the cases where they differ, our results still support the overall findings from the original work. For instance, our distances from the optimal state values have a different distribution from the paper's, but their values are also smaller, meaning even more accurate than expected. For these reasons, we can affirm the findings from the original paper. There are however some notable reproducibility issues, such as missing parameters, in particular the size of the grid world, and some unspecified procedures, in particular how to count calls *in practice*. The notation used is also generally unintuitive, which might be an issue on our part, but still hinders general reproducibility. These issues are likely the cause for our slight deviation from the paper's results.

6 Link to Code

https://github.com/AdamPrevost/INF8953.git

7 Contributions by each team member

7.1 Alexandre Morinvil 1897222

- Implementation of NC-hm-PI and hm-PI.
- Experiment runs for NC-hm-PI and hm-PI.
- Redaction of sections 2 (partially), 3 (partially) and 4.1.

7.2 Maude Nguyen-The 1843896

- Implementation of NC-hλ-PI and hλ-PI.
- Implementation of the experiment framework.
- Experiment runs for NC-hλ-PI and hλ-PI.
- Redaction of sections 2 (partially), 4.2 and 4.3.

7.3 Adam Prévost 1947205

- Implementation of the base framework and environment.
- Implementation of h-PI.
- Redaction of sections 1, 2 (partially) and 3.

References

Jonathan Baxter, Andrew Tridgell, and Lex Weaver. Tdleaf(lambda): Combining temporal difference learning with game-tree search. *CoRR*, cs.LG/9901001, 1999. URL https://arxiv.org/abs/cs/9901001.

Dimitri P. Bertsekas and Sergey Ioffe. Temporal differences-based policy iteration and applications in neuro-dynamic programming. Technical report, 1996.

Dimitri P. Bertsekas and John N. Tsitsiklis. Neuro-dynamic programming: an overview. *Proceedings of the IEEE Conference on Decision and Control*, 1:560–564, 1995. ISSN 0191-2216. Proceedings of the 1995 34th IEEE Conference on Decision and Control. Part 1 (of 4) ; Conference date: 13-12-1995 Through 15-12-1995.

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. Beyond the one step greedy approach in reinforcement learning. *CoRR*, abs/1802.03654, 2018a. URL http://arxiv.org/abs/1802.03654.

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. Beyond the one step greedy approach in reinforcement learning, 2018b.

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. Beyond the one-step greedy approach in reinforcement learning. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1387–1396. PMLR, 10–15 Jul 2018c. URL https://proceedings.mlr.press/v80/efroni18a.html.

Yonathan Efroni, Gal Dalal, Bruno Scherrer, and Shie Mannor. How to combine tree-search methods in reinforcement learning, 2019.

Martin L. Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. *Manage. Sci.*, 24(11):1127–1137, jul 1978. ISSN 0025-1909. doi: 10.1287/mnsc.24.11.1127. URL https://doi.org/10.1287/mnsc.24.11.1127.

Jost Tobias Springenberg, Nicolas Heess, Daniel Mankowitz, Josh Merel, Arunkumar Byravan, Abbas Abdolmaleki, Jackie Kay, Jonas Degrave, Julian Schrittwieser, Yuval Tassa, Jonas Buchli, Dan Belov, and Martin Riedmiller. Local search for policy iteration in continuous control, 2020.

Anna Winnicki, Joseph Lubars, Michael Livesay, and R. Srikant. The role of lookahead and approximate policy evaluation in policy iteration with linear value function approximation, 2021.