Assignment 3: Model-based reinforcement learning

Adrien Im (s3984389) & Bence Válint (s3796426)
Group Number: 4

1 INTRODUCTION

Reinforcement learning is an important approach in artificial intelligence for agents to learn optimal behavior through different interactions with their environment. Within reinforcement learning, model-based agents have the advantage to construct a representation of their environment to enhance their ability to learn. This report will investigate two model-based reinforcement learning algorithms, more specifically the Dyna and Prioritized Sweeping (PS) algorithms. These algorithms will be compared with a model-free algorithm to investigate the effect of a 'mental simulation' on the algorithms' performances. Each algorithm will be investigated with a range of planning iterations between real-world steps in both a stochastic and a static environment. The environment that will be used to conduct this research will be the Windy Gridworld environment which is a basic grid where there is a certain chance for wind. The agent starts in the position $(0, 3)$ and aims to achieve the goal at $(7, 3)$. Each step gives an immediate reward of -1 and reaching the goal gives a reward of $+100$. In the grid there is 'wind' in columns 3, 4, 5 and 8 that pushes the agent up by a grid cell based on a random chance controlled by the $wind\_proportion$ parameter. In columns 6 and 7 the same wind is in place but it pushes the agent up two grid cells instead of one.

2 DYNA

2.1 Methodology

The main goal of reinforcement learning is to approximate the optimal action-value function $Q^*(s, a)$ which is the expected return of taking action a in the state s following the optimal policy π^*. Q-Learning approaches the optimal state action value using the formula: $Q(s, a) \leftarrow Q(s, a)+\alpha[r+\gamma \max_{a'} Q(s', a')-Q(s, a)]$, where α is the learning rate, γ is the discount factor, r is the immediate reward and s' is the next state. Throughout the experiment, a learning rate of 0.2 was used. The model-based reinforcement learning approach, Dyna, has the distinct feature of simultaneously constructing an environmental model. For each transition (s, a, r, s'), the agent updates its model. This is done by maintaining transition counts $n(s, a, s')$ and reward sums $R_{\text{sum}}(s, a, s')$. By keeping track of these, the agent can estimate the transitional probabilities $\hat{P}(s'|s, a) = \frac{n(s,a,s')}{\sum_{s''} n(s,a,s'')}$ and the expected rewards $\hat{R}(s, a, s') = \frac{R_{\text{sum}}(s,a,s')}{n(s,a,s')}$. The agent can also plan ahead using simulated experiences on the learned model. For each step in n steps, a previously visited state-action pair (s, a) at random. Subsequently, it selects a next state s' corresponding to the transition probabilities $\hat{P}(s'|s, a)$ and calculates the expected reward $\hat{r} = \hat{R}(s, a, s')$. Lastly, the agent performs a Q-Learning update using the formula: $Q(s, a) \leftarrow Q(s, a)+\alpha[\hat{r}+\gamma \max_{a'} Q(s', a')-Q(s, a)]$. To approach the optimal state-action values, the agent first observes its current state s, then selects an action using the $\epsilon-$greedy policy. It receives a reward r and the next state s' and based on these it updates its Q-values. The agent updates its transition counts $n(s, a, s')$ and reward sums $R_{\text{sum}}(s, a, s')$. Finally, it performs n planning updates using simulated experiences from the model [1].

2.2 Results

To investigate the difference that the number of plan steps has on the episode returns over a period of 10001 we setup an experiment in a stochastic environment (wind is applied 90% of the time) and an experiment in a static environment (wind is always applied).

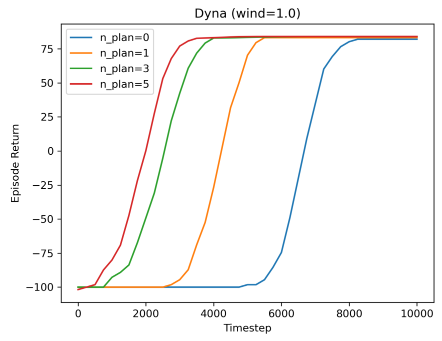

Figure 1: Learning curves of Dyna agents and a baseline in the Windy Gridworld environment (wind = 1.0). Results are averaged over 20 repetitions and smoothed.

Starting with the simple static environment where it is always windy, the clear advantage of Dyna compared to Q-Learning is visible in Figure 1. The blue line with no planning is representative of Q-Learning while the other lines are Dyna agents with different number of planning steps. The Q-Learning agent shows the slowest learning curve requiring around 8,000 timesteps to reach optimal performance. From the experiment it is also visible that the Dyna agents with a higher number of planning steps have a faster learning curve reaching optimal performance much earlier, clearly highlighting the advantage of Dyna; learning through simulating experiences with planning.

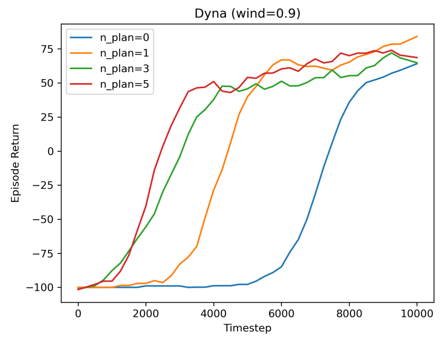

Figure 2: Learning curves of Dyna agents and a baseline in the Windy Gridworld environment (wind = 0.9). Results are averaged over 20 repetitions and smoothed.

In regards to the stochastic environment in figure 2, while the curves are not as clear as in the static environment the Dyna agents are visibly outperforming the Q-Learning agent. The Q-Learning agent has a slow learning curve, while the Dyna agents begin learning and increasing the episode returns much earlier. Similarly to the static environment, the agents with a large number of planning steps show quicker learning represented by larger episode returns at earlier timestamps. The curves are visibly less smooth due to the randomness caused by the stochastic environment and agents can not yield maximum episode returns, but the large difference in the learning efficiency of Dyna and Q-Learning agents is clearly visible.

2.3 Interpretation

The above findings have several implications for reinforcement learning applications. In the Dyna agent, each planning step provides additional value updates without requiring an incrementing timesteps. This results in the learning of the agent being almost multiplied by the number of planning steps. This is also supported by the graphs, for example in Figure 1 it takes around 8000 timesteps for the Q-Learning model to reach optimal performance, while the agent with 5 plan steps takes a little over 2000 timesteps to reach optimal performance; little more than $\frac{1}{5}$th of timesteps. The diminishing returns are also visible on both charts. The difference in learning between the Q-Learning agents and the Dyna agent with one planning step is much greater than the difference in learning between the Dyna agents with 3 and 5 planning steps. This highlights the sample efficiency versus computation trade-off. While adding extra planning steps does increase the sample efficiency since the agent learns quicker, after a certain steps of planning adding extra time planning steps increases computation time with little to no effect on sample efficiency. Therefore, Dyna agents perform substantially better than Q-Learning agents by performing additional state action updates through planning. This allows them to have a greater episode return and approach the optimal performance earlier. However, planning comes at the cost of computation, so it is important to consider the sample efficiency versus computation trade-off.

3 PRIORITIZED SWEEPING

3.1 Methodology

Prioritizes sweeping (PS) is another sophisticated model-based reinforcement algorithm. Similarly to Q-Learning the agent maintains a Q-value function $Q(s,a)$ for each state-action pair representing the expected cumulative reward: $Q(s,a) \approx \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\right]$. To model the environment the agents keeps track of the transition counts $n(s,a,s')$ and reward sums $R_{\text{sum}}(s,a,s')$ exactly like the Dyna agent. However it also keeps track of the predecessor sets $Pred(s)$ which contains all the state action pairs (s',a') that can lead to state s. By keeping track of these the agent can derive the transitional probabilities $\hat{P}(s'|s,a) = \frac{n(s,a,s')}{\sum_{s''} n(s,a,s'')}$ and the expected rewards $\hat{R}(s,a,s') = \frac{R_{\text{sum}}(s,a,s')}{n(s,a,s')}$, the same way the Dyna agent can. To learn the agent applies the learning rule: $Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha\delta_t$, where α is the learning rate, and δ_t is the temporal difference error at timestep t. The temporal difference error can be calculates using the formula: $\delta_t = [r_t + \gamma \times \max_{a'} Q(s_{t+1},a')] - Q(s_t,a_t)$. For the goal state the $\max_{a'} Q(s_{t+1},a')$ is set to zero. What makes PS unique is priority based magnitude of temporal difference error. Therefore the priority of a state action pair can be calculated using: $p(s,a) = |\delta_{s,a}|$, where $\delta_{s,a}$ temporal difference error of state action pair (s,a). If this priority exceeds the cutoff threshold θ then the state-action pair is added to the priority queue. During planning this agent selects the highest priority state-action pair (s,a) and finds a next state s' according to the transitional probabilities $\hat{P}(s'|s,a)$. Next it computes the expected reward $\hat{r} = \hat{R}(s,a,s')$, and finally it updates the Q-value using the formula: $Q(s,a) \leftarrow Q(s,a) + \alpha[\hat{r} + \gamma \max_{a'} Q(s',a') - Q(s,a)]$. After updating a Q-value the model adds all the state action value pairs (s_{pred}, a_{pred}) that could lead to s to the predecessor set $Pred(s)$. It also calculates the expected reward $\hat{r}_{pred} = \hat{R}_(s_{pred}, a_{pred}, s)$ and using the expected reward it calculates the potential temporal difference error if (s_{pred}, a_{pred}) were updated: $\delta_{pred} = \hat{r}_{pred} + \gamma \times \max_{a'} Q(s,a') - Q(s_{pred}, a_{pred})$. If it exceeds the cutoff threshold θ it is added to the priority queue. This is repeated n times to achieve $n - step$ planning. During evaluation the model uses a greedy action selection [1].

3.2 Results

To investigate the performance of the prioritized sweeping model, an experiment was constructed for PS agents with a range of $0 - 5$ planning steps. The experiments were carried out in the same manner as the experiment with Dyna agents; one in a static environment where it is always windy and one in a stochastic environment with 90% wind.

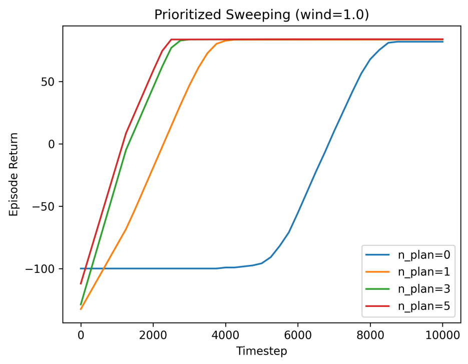

Figure 3: Learning curves of Prioritized Sweeping and a baseline in the Windy Gridworld environment (wind = 1.0). Results are averaged over 20 repetitions and smoothed

First looking at the static experiment it is clear that PS achieves learning much faster than Q-Learning. While Q-Learning reached optimal performance at around 8,000 timesteps, the PS agent with 5-step planning managed to do it in 2,000 timesteps. A pattern can be seen in this environment, where the greater number of planning steps results in faster optimal performance. Moreover, the same pattern as in Dyna agents can be seen with diminishing returns; the improvement in learning from 0 step planning to 1 step planning is significantly greater than the improvement from 3 step to 5 step planning agents. This also highlights the sample efficiency versus computation dilemma due to the diminishing returns. The PS agent's learning curve differs greatly from the Dyna agent's learning curve because the learning curve is much closer to linear. While the Dyna agent resembles the learning curve of the Q-Learning agent, the PS agent has a sharp linear learning curve starting from the first timestep unlike the Dyna agents.

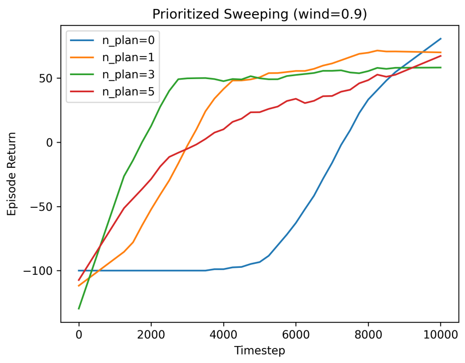

Figure 4: Learning curves of Prioritized Sweeping and a baseline in the Windy Gridworld environment (wind = 0.9). Results are averaged over 20 repetitions and smoothed.

Looking at the stochastic graph in Figure 4, the learning curves differ greatly from the static environment. The main difference is that the PS agent with the largest number of planning steps does not learn the fastest. It is still clear that PS agents show greater episode returns earlier in the timesteps, but the pattern of a greater number of planning steps resulting in greater optimal performance is not shown anymore. The agent with 3 planning steps depicts the quickest learning which also resembles a linear pattern until the maximum performance of the agent is reach at episode return = 50. Although quick learning is achieved, the other agents reach a greater maximum return. Interestingly, the Q-Learning agent reaches the maximum episode return in the last timestep.

3.3 Interpretation

The first and most noticeable difference between Q-Learning and prioritized sweeping is the learning curve. The very linear nature of the PS's learning curve is due to its ordered approach to value propagation. PS agents process state action pairs in order of their expected impact. This is measured by the temporal difference error, which is the basis for the priority queue. Due to a constant number of planning steps, value information is spread through the state space at a relatively constant rate, resulting in a linear learning curve. Each real experience and n step plan allows approximately $k(n+1)$ states to receive accurate value updates, where k is a constant. The improvement is roughly proportional to the number of states with accurate values; therefore, the improvement in episode return is proportional to $k(n+1)$ which is a constant. The diminishing returns are due to the same factor. A small increase in the number of planning steps n is significant at a low n but at higher values the difference is negligible. Regarding the stochastic model seen in Figure 4, the reason why the PS agent with 5 planning steps has a lower learning rate is due to the stochastic nature of the model. Although using five planning steps allows more information to be learned, it also proportionally allows for noise to be learned. At the beginning of the training the agent's model is based on very little number of samples, which can lead to transitional probabilities getting highly skewed. With five planning steps, the model becomes more susceptible to overfit to these skewed transitional probabilities, creating a suboptimal policy that leads to less learning. Later in the training, when the model obtains more samples, the 5 planning steps become beneficial since the transnational probabilities will approximate the model's actual probabilities and the agent will be able to leverage more information and refine the policy to optimal due to more thorough backward propagation of value information. With greater planning steps, the agent has more opportunities to observe low probability, but potentially significant transitions, which leads to more robust performance. The interesting phenomenon of the Q-Learning agent achieving a greater return per episode can be due to its model-free advantage. Q-Learning does not require any planning and only uses experience gained by actual steps. It also does not rely on estimated transitional probabilities and rewards. Moreover, each update solely incorporates true environmental dynamics rather than approximations. This allows the Q-Learning agent to be free of any model bias and therefore converge to a more accurate representation of the true value function.

4 COMPARISON

To investigate the differences between Dyna and Prioritized Sweeping agents it is important that we evaluate the two algorithms in the same environment. We therefore simulated and plotted both a Dyna and a PS agent on the same environment starting with the static environment.

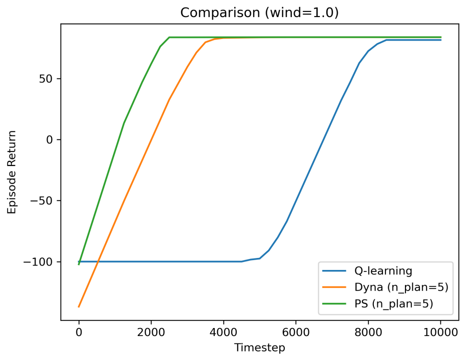

Figure 5: Learning curves of Dyna and Prioritized Sweeping agents with 5 planning steps in the Windy Gridworld environment (wind = 1.0). Results are averaged over 20 repetitions and smoothed.

Figure 5 clearly depicts that the PS agent seems to have a faster learning and achieves the optimal performance much quicker then the Dyna agent. This can be explained by the difference in the two algorithm's approach to exploration. PS keeps track of which state action will have the greatest expected impact. The states with the largest prediction error get updates first and when there is a large value change the model back propagates and updates the predecessor states too. This creates a cascading effect where information about the obtained rewards flows efficiently backward through the state space. On the other hand, the Dyna agent updates state action pairs randomly. This results in some updates with minimal or no impact leading to a slower learning.

Looking at the differences in the stochastic environment in Figure 6 there are bigger differences between the two agents. First,

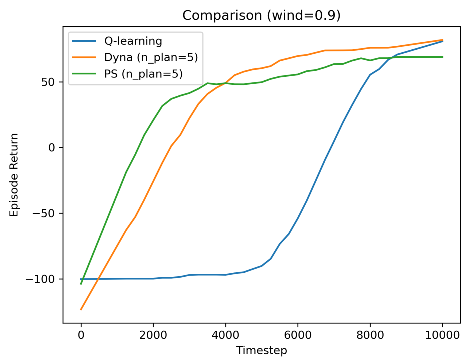

Figure 6: Learning curves of Dyna and Prioritized Sweeping agents with 5 planning steps in the Windy Gridworld environment (wind = 0.9). Results are averaged over 20 repetitions and smoothed.

due to the previously explained reason, the PS agent achieves a quicker learning and has greater episode returns until around 4,000 timesteps when the Dyna agent yields a higher episode return. While the learning of the Dyna agent is slower initially due to its randomness in exploration, in the stochastic environment the comprehensive exploration becomes a big advantage after sufficient learning leading to higher episode returns. This is because of how the PS agent treats stochasticity. Its prioritization mechanic is highly sensitive to noise caused by the randomness of the environment leading to higher temporal difference errors and an over-prioritization of states whose values fluctuate due to environmental stochasticity. This leads to less exploration of states that might be impactful and thus a lower maximum episode return.

Algorithm	Average Runtime per Repetition (s)
Q-learning	2.80 ± 1.41
Dyna	2.83 ± 1.82
PS	3.11 ± 2.26

Table 1: Average runtime per repetition (in seconds) for Q-learning, Dyna, and Prioritized Sweeping over 20 runs. Planning-based methods show higher computational cost, with PS being the slowest.

Table 1 above clearly highlights the computation versus sample efficiency tradeoff. While the PS agent and the Dyna agents learn quicker they also take more time per timestep due to the extensive computation required for the planning. The Q-Learning algorithm has the lowest average runtime per repetition due to it not using any planning followed by the Dyna algorithm. The difference is minimal with only 0.03s more time per repetition on average. This is because although it performs planning it does it in a much quicker way than PS. PS needs to calculate the temporal difference error adding to the runtime and it also needs to keep track of a priority queue which also significantly increases its runtime. This results in quicker learning as seen in figure 5 but results in a more computationally intense algorithm shown by table 1.

5 REFLECTION

To evaluate the strengths and weaknesses of model-based and model-free algorithms it is essential to consider the learning curves. Model-based algorithms are more sample efficient, leading to quicker learning and higher episode returns earlier into the learning process. This is due to their ability to learn from simulated experiences amplifying the information gain per timestep. This also leads to the model-based agents to quickly adapt if the environment would change. However, this strength comes with a key limitation. While the model-free agent learns much slower it explores all the state space leading to the highest potential episode return while model-based algorithms might not achieve optimal performance. This phenomena is especially seen in figure 6, where the PS agent doesn't reach the maximum episode return. The planning of the model-based algorithms also comes at the cost of computation leading to slightly slower times per timestep as seen in table 1.

Dyna and PS both have distinct benefits. While PS achieves quicker learning and a faster adaptation, it lacks final performance. On the other hand, the Dyna agent learns quicker but yields a higher episode return. This highlights the practical differences in implementation; a PS algorithm would excel in tasks where it has to consistently adapt to changing environments while the Dyna agent performs best in stable environments. A robust solution would be a combination of the two where the priority based approach is used for the start of the training and then eventually it slowly changes into a random exploration to ensure the maximum potential episode return.

With the Q-values initialized to 0 and a -1 step reward agents begin with an optimistic value estimate compared to the true expected reward. This newly created optimism encourages systematic exploration because the unexplored state-action pairs will appear better than they likely are. This results in a form of optimistic initialization which was explored in the context of model-free agents. This will promote exploration in bandit problems and significantly different exploration patterns. Model-based environments will quickly overcome this created bias towards exploration through their ability of back propagation and planning. On the other hand Q-Learning might take more time to overcome the optimal initialization.

6 CONCLUSION

The findings of the paper reveal important considerations for selecting an algorithm in reinforcement learning. The model-based agents' enhanced learning capabilities and adaptability to a change in the environment are clearly depicted. However, this advantage comes at a considerable trade-off for computational resources. Furthermore, our findings show that in stochastic environments model-based agents may struggle to reach optimal performance. Model-based algorithms excel in dynamic environments and show exceptional learning capabilities, but model-free agents remain crucial in highly stochastic environments, where they guarantee optimality and in situations with limited computational resources.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.