Implementing Exploration Bonuses In The Racetrack Programming Environment

Kaylyn Wiese

March 2025

1 Introduction

In this experiment, we seek to explore the impact of exploration bonuses when they are applied in different steps of the Dyna-Q algorithm.

2 Experiment Background and Parameters

2.1 Experiment Background

For this experiment, we chose to work with the environment previously built for the Racetrack Programming experiment. For more information on the environment and track setup, please see the Racetrack Programming document, which details how the tracks are built and the movement of the agent (car) throughout the tracks. We also chose to work with the Dyna-Q algorithm to help the agent learn to navigate the grid world. Similar to Q-learning, Dyna-Q's goal is to learn an optimal action-value function (Q(s,a)). Dyna-Q builds on this basic principle by forcing the agent to learn directly from real interactions with the environment as well as from planning using a learned model of the environment.

2.2 The Dyna-Q Algorithm

Unlike model-free variations of reinforcement learning, Dyna-Q implements a model that helps the agent learn. To put it simply, imagine if you were placed in the middle of a corn maze and needed to find your way out. In normal Q-learning, you would try moving in different directions and begin to understand the reward from taking those actions. For example, "If I turn left, I hit a wall" or "If I move right, I get closer to the exit". While Dyna-Q incorporates those real-life learning moments, it also forces you to walk around the maze once, which will allow you to better understand how the maze works based on what you've seen. It also allows you to "practice" navigating and understanding the maze in your head. Because you don't need to explore everything "physically" in the maze, it allows you to learn faster and become smarter by imagining what could happen based on what you've already seen. These are the same principles applied to the agent navigating a grid world. Because Dyna-Q combines real experiences with simulated ones, the agent is able to learn faster and more efficiently.

2.2.1 Q-Learning Update: Real Experience

As discussed above, our agent still learns from real experiences in Dyna-Q. Every time the agent takes an action in the environment and observes the reward and next state that come from that action, it updates Q-values using the Bellman Equation.

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

Where:

- s = current state

- a = action taken
- r = reward received
- s' = next state
- α = learning rate
- γ = discount factor

2.2.2 Model Learning

After each real step, the Dyna-Q algorithm also updates a model:

$$Model(s, a) = (s', r)$$

This means that for every state, action pair, the model stores the resulting next step (s') and the reward (r) in a dictionary.

2.2.3 Planning Steps

After updating the Q-value from the real step, the Dyna-Q algorithm also performs simulated updates where it:

- Randomly samples state-action pairs that were previously observed
- Use the model to get the next state and the resulting reward
- Apply the same Q-learning updates

It's important to note that these Q-learning updates are not real updates, but rather simulated updates. That is to say that they come from the model, rather than the real-life experience of the agent.

2.3 Exploration Bonus Expansion: Dyna-Q+

With the background of the baseline Dyna-Q algorithm in mind, this experiment seeks to expand on our original by factoring in an exploration bonus. In reinforcement learning, an exploration bonus can be added in to encourage the agent to try actions that have yet to be explored. When applied in the Dyna-Q algorithm, an exploration bonus can help push the agent to a more directed exploration strategy that can handle stochastic environments or environments with sparse rewards. In the case of Dyna-Q+, the reward function used during planning changes and incorporates the exploration bonus and indirectly impacts the Q-values:

$$r' = r + \kappa \cdot \sqrt{\tau(s, a)}$$

Where:

- r = the real reward
- κ = exploration bonus coefficient (how much exploration you want)
- τ(s,a) logs how long it's been since the agent last visited that state-action pair.

This means that the bonus is only applied during the planning phase of the Dyna-Q+ algorithm and real-world updates are left untouched.

2.4 Exploration Bonus In Action Selection

While the Dyna-Q+ algorithm implements the exploration bonus in the Q-value updates, the problem statement of this experiment asks us to attempt to implement the exploration bonus in the action selection rather than modifying the reward function to optimistically inflate the Q-values during the planning step, we apply it to the Q-values during action selection.

$$a = \arg\max_a \left[Q(s,a) + \kappa \cdot \text{Bonus}(s,a) \right]$$

This changes how the agent actually selects the action.

2.4.1 Measurement

Similar to our initial experiment (Racetrack Programming), we will measure the learning speed and time to convergence of the agent in these two scenarios. We will also measure the variance.

2.5 Initial Model Outcome (Dyna-Q)

We ran our baseline Dyna-Q model on track a (the simplest track). After running it over 1,000 episodes, we saw that it converged with higher rewards than we saw in our last experiment with Monte Carlo control algorithms. This is because the agent is able to engage in planning via simulated model-based experience.

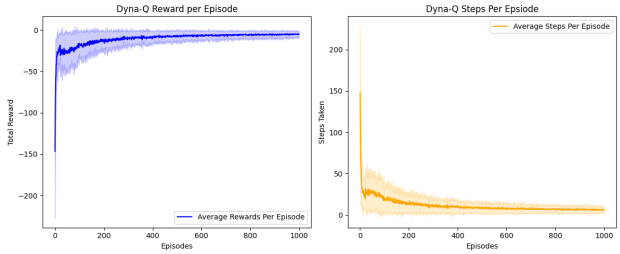


Figure 1: Dyna-Q Baseline with no Exploration Bonus

2.6 Dyna-Q+

After running the baseline, we modified our algorithm so the exploration bonus is incorporated into the reward function of the planning phase to update the simulated Q-values in the model. While the model converged, we saw initial lower rewards. This comes from the fact that the agent is intentionally exploring more instead of just setting off down a path of already known rewards. While the differences are subtle from the baseline Dyna-Q model, the Dyna-Q+ model also experiences slightly less variance mid-training as the agent starts to learn what actions are good as τ grows less quickly in relation to prior steps, effectively decaying the exploration bonus. Ultimately, we see the agent start to stabilize as it converges. Once again, the nuances here are subtle in the charts, but the agent ultimately converges to a better policy than our baseline, because the exploration bonus we implemented should drive a better method of exploration for the agent. This means the agent has more of a possibility of uncovering high reward paths it might have ignored otherwise.

2.7 Dyna-Q+ with Modified Exploration Bonus

After running our Dyna-Q+ model with the traditional exploration bonus in the reward update of the planning process, we modified our algorithm to incorporate the exploration bonus into the real-world action selection of the algorithm. This means that when the agent is selecting an action in real life, it incorporates the exploration bonus. This adds an additional layer of variability as the agent is incorporating the exploration bonus into its real-life action selections. While the differences are subtle, our curves saw our agent experience more variability in the initial episodes.

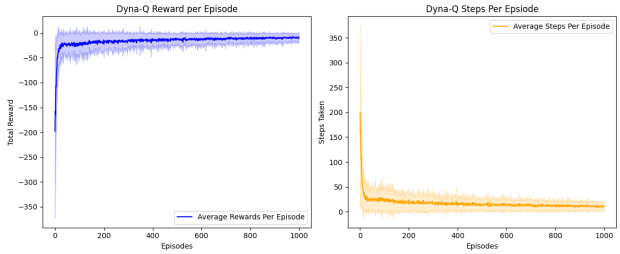


Figure 2: Dyna-Q+ with Exploration Bonus

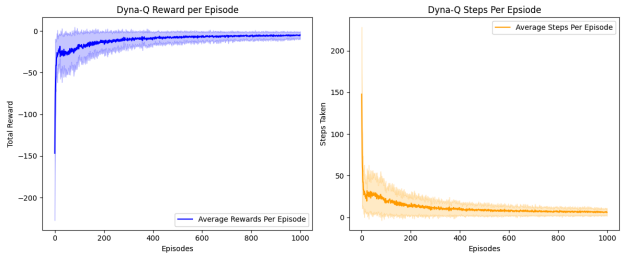
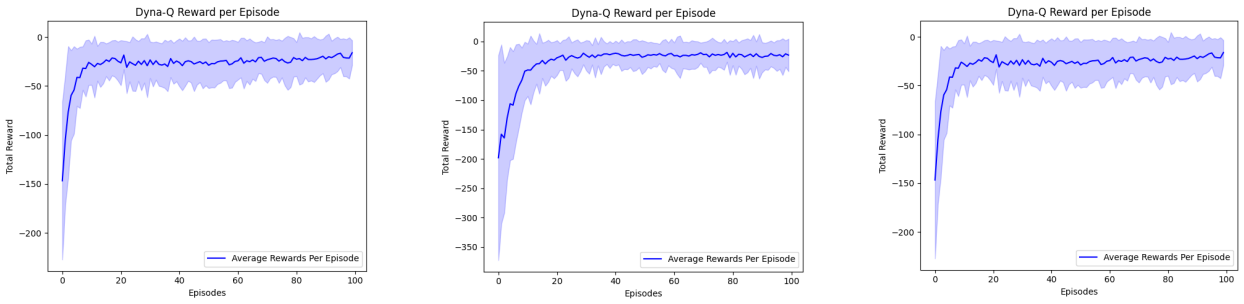


Figure 3: Dyna-Q+ with Exploration Bonus In Action Selection

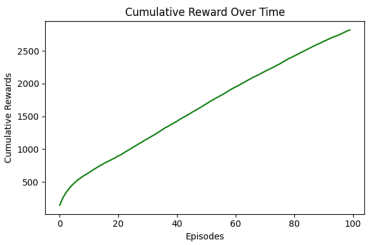
3 Comparing The Three Models

Because the track we tested the agent on was relatively simple, the nuances in the differences between the models were rather subtle. To help get a better understanding of these differences, we zoomed in on the first 100 episodes. When looking at our base Dyna-Q model compared to our Dyna-Q+ model over the first 100 episodes, we can clearly see a much lower initial reward with a significantly larger amount of variability. However, it's important to note that while Dyna-Q+ experiences more variability and lower rewards at the outset, that variability starts to drop off as the model starts to approach convergence. When we look at the Dyna-Q+ and compare that further to the Dyna-Q+ with the exploration bonus in the action selection, we don't see as low of initial rewards, but we don't see the variability diminish as much as it did when the exploration bonus was applied in the planning phase. This is because the exploration bonus impacts the agent's real-life decisions. When we look further at the cumulative rewards of each iteration, we see the highest cumulative rewards coming from our Dyna-Q+ algorithm with the exploration bonus in the planning phase.

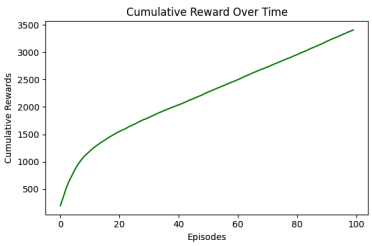


(a) Dyna-Q Base Model (b) Dyna-Q+ (c) Dyna-Q+ With EB In Action

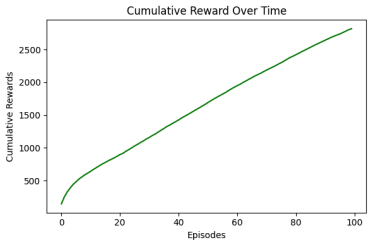
Figure 4: All three models zoomed into first 100 episodes.



(a) Dyna-Q Base Model



(b) Dyna-Q+



(c) Dyna-Q+ With EB In Action

Figure 5: Cumulative performance of all three models.

4 Dyna-Q Conclusion

After implementing three different models with three different iterations of the exploration bonus (no exploration bonus, exploration bonus in planning, and exploration bonus in action selection), we can conclude that implementing our exploration bonus in the planning phase allows the model to better explore while also reducing variability. While the Dyna-Q base has initially better rewards, we don't see as optimal of a policy achieved, because the agent isn't able to explore as much. Conversely, while the Dyna-Q+ algorithm with the exploration bonus implemented in the action selection has higher initial rewards, the variability continues throughout the model, as the agent is continuing to explore in real life since that's where the exploration bonus is implemented.

5 Expanding Our Experiment

In order to get a better understanding of the impact of the exploration bonus, we elected to implement it on a much simpler algorithm. In our previous experiment with Racetrack Programming, we used the Monte Carlo control algorithm to train the agent to navigate the gridworld, so it made sense that we would continue to expand on the work of those algorithms to see if we could improve them with the exploration bonus.

After running the results over 10k episodes, we zoomed in to see what it looked like at 100 episodes (often times, the 10,000 episode graphs are too small for us to really see the details in this format). Below are the findings from the initial Monte Carlo control base model with no exploration bonus, the Monte Carlo control model with the exploration bonus in the Q-value updates, and the Monte Carlo control model with the exploration bonus in the action selection. All models were run on Track A, which is the simplest track built during the initial racetrack programming experiment.

5.1 Baseline Monte Carlo Control Model

We utilized the same baseline Monte Carlo control model that we built for our original racetrack programming experiment and ran it again to see the results of the model with no exploration bonus. Here, the model converged as expected around 100 episodes. While it had low initial rewards and high initial variance, we saw that the agent was able to eventually find a good path and stabilize performance.

5.2 Bonus In Q-Value Updates

We took the base Monte Carlo control model that we had originally created for our Racetrack Programming experiment and slightly modified it so that the agent incorporated the exploration bonus into the Q-value updates. This modified the update function as follows:

$$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{N(s,a)} G_i^{\text{bonus}}$$

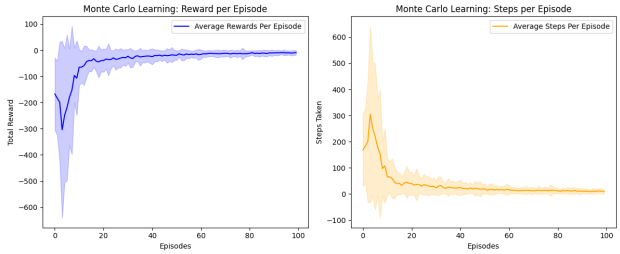


Figure 6: Monte Carlo Control baseline model with no exploration bonus

Where the bonus is computed as:

$$G_i^{\text{bonus}} = \sum_{t=0}^{T_i} \gamma^t \left(r_t + \frac{c}{\sqrt{N(s_t, a_t) + 1}} \right)$$

Where:

- s = state
- a = action
- N(s,a)= visit count for state-action pair
- Q(s,a) = action value estimate for state-action pair
- r_t = reward at timestep
- γ = discount factor for how much future rewards matter
- c = exploration coefficient to scale the magnitude of the exploration bonus

Our c parameter was set at one and would remain set at one for this iteration of the experiment. The c coefficient in the exploration bonus is a tunable parameter where the higher the c parameter is, the larger your exploration bonus. It's also important to note, that when we ran a similar methodology with our Dyna-Q+ algorithm, the Q-values were only updated in the simulated model. In the case of Monte Carlo, the algorithm is model-free, so all Q-values that are updated are the real Q-values. The outcome of this means that while we might be able to get faster convergence from the agent, we can also end up with Q-values that don't reflect the actual return.
Another important difference between the exploration bonus in Dyna-Q and Monte Carlo is that our Dyna-Q exploration bonus is rooted in the time since the state-action pair was visited (τ(s,a)). At each step, τ(s,a) is incremented and then reset when the pair is visited. This is possible because Dyna-Q uses a model of the environment, which makes it easier to maintain these counts. In contrast, Monte Carlo is model-free and therefore, we can only apply updates to state-action pairs we actually have encountered. So instead of the equation that we saw for the Dyna-Q exploration bonus, we compute the inverse of the count of visits.

Our model managed to start to converge around 100 episodes, though it was still experiencing a bit more variance than our initial Monte Carlo control model with no exploration bonus. Another variation that came with this model was that we also saw higher initial rewards. We can infer that this happens because we're adding some level of inflation to the Q-values. While this can speed convergence, because the agent is more likely to pick actions that it hasn't tried, it can cause variance to hang around longer than we might like, because the bonus is still impacting it further downstream when we're trying to converge and we might not want to be exploring as much. To put it simply, the Q-values that we're using down the road can remain artificially optimistic causing them to be biased where we are seeking stability from the agent. However the count should eventually increase over time, causing the bias to diminish.

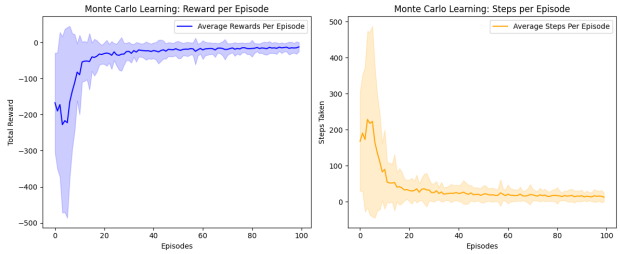


Figure 7: Monte Carlo Control with exploration bonus in Q-value updates

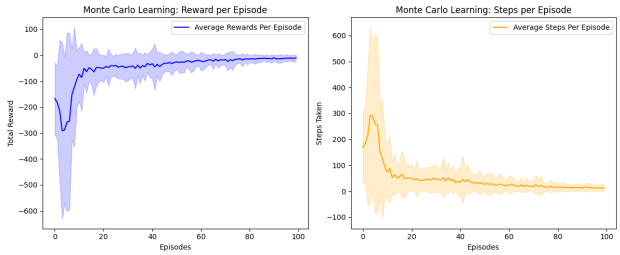


Figure 8: Monte Carlo Control with exploration bonus in action selection

5.3 Bonus In Action Selection

In our final model, we incorporated the bonus into the action selection. We return to our standard Monte Carlo control model. However, this time we leave the Q-values alone and instead incorporate the exploration bonus into the action selection. This means that our Q-values will maintain unbiased estimates. However, when an action is selected, the agent chooses the action that maximizes:

$$Q(s, a) + b(s, a)$$

In this case, our exploration bonus will only affect the behavior of our agent (the policy), not the learned Q-values.

In this model, our agent successfully converged around 100 episodes. However, it had lower initial rewards. This is expected because we are not artificially inflating the Q-values. While it might not be immediately obvious in the graphs, using this strategy should eventually lead to a more stable policy with Q-values that are reflective of the actual returns. Therefore, even though adding the bonus to the Q-values speeds up convergence, it might not be actually representative of the true rewards.

6 Further Iterations: Monte Carlo Off Policy

We implemented the same exploration bonus methodology to expand on this experiment, using an off-policy Monte Carlo model. In this method, the agent learns the value of a target policy while behaving according to another policy. Because we used the same model implemented in the final version of our Racetrack Programming experiment, it's also important to note that we utilized ε-decay. We implemented this as part of our Monte Carlo off-policy model to help reduce variance as the agent converges on an optimal path to the finish line. In ε decay, we decay the ε parameter as the episodes progress to a very small target ε that will encourage less exploration and more exploitation bringing it closer to the target policy. As it gets closer to the target policy, we'll see less variance in the sampling weights and therefore less variance overall. We implemented this model for comparison across a baseline model with no exploration bonus, a model with the exploration bonus in the Q-value updates, and finally a model with the exploration bonus in action selection. The results for each model are summarized below.

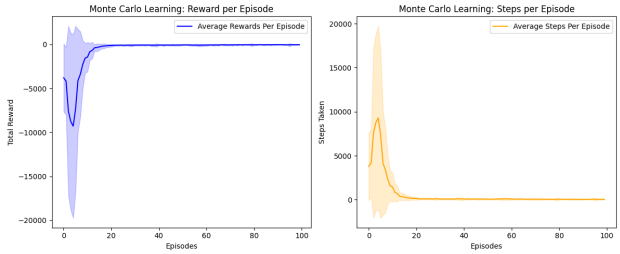


Figure 9: Monte Carlo Control Off-Policy Baseline

6.1 Off-Policy Baseline

In our baseline model, we witnessed the agent converge rather quickly with reduced variance after its initial exploration when compared to our on-policy baseline model. This is to be expected, as while our off-policy method introduces more variance due to high exploration, the smaller ε at the end of the training forces the agent to exploit more resulting in reduced variance and better performance.

6.2 Off-Policy With Exploration Bonus In Q-Values

We implemented our off-policy model with the exploration bonus in the Q-values, still maintaining our ε-decay. In both our on and off-policy methods, our Q-values are optimistically biased, because we are implementing our exploration bonus in the Q-values. However, with our off-policy method, we are seeing a much bigger push for the agent to explore initially. This, along with importance sampling introduces high amounts of variance early on. However, as the training progresses and the ε decays, we see the agent converge to a more stable policy with less variance. While the Q-values are still biased, the agent is able to better exploit what it has learned. Finally, while our convergence might appear faster (though it is quite subtle), we still need to keep in mind that our Q-values are still biased, which may lead to a policy that is suboptimal because the bias tricks the agent into thinking it has arrived at an optimal path. However, as the visit counts increase for state-action pairs, the bias will become gradually reduced over time. However, it's important to note that even with reduced bias, the learning may have already been skewed towards a suboptimal section of the state, causing the agent to select a suboptimal policy.

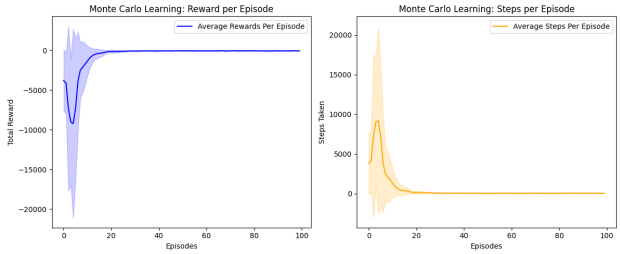


Figure 10: Monte Carlo Control Off-Policy Exploration Bonus In Q-Values

6.3 Off-Policy With Exploration Bonus In Action Selection

In our off-policy model with our exploration bonus in action selection, we see that when the agent picks the action based on the maximum that includes the exploration bonus, we get a higher amount of variance initially when compared to our on-policy method with our exploration bonus in action selection. This comes not only from the exploration bonus but also the importance sampling introduced in our off-policy methods. However, our ε-decay ultimately helps our model to become more stable later on. However, it is still less stable than our off-policy baseline with no exploration bonus, because even with ε-decay, our agent is still forced to explore more than it might like to later on. Because the exploration bonus lies in the action

selection, the agent is pushed to continue exploring less-visited actions, even if it has already found an efficient path forward. Ultimately, this will lead to slower convergence and less stability overall.

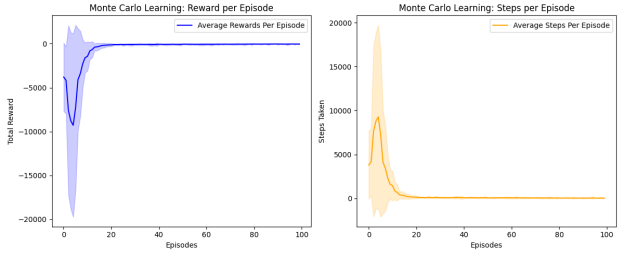


Figure 11: Monte Carlo Control Off-Policy Exploration Bonus In Action Selection

7 Exploring More Complex Environments

While all of this might seem arbitrary looking at an agent exploring an extremely simple racetrack (in all our training runs above, we used track a, which is the simplest of all the tracks), these subtle differences become increasingly more important when the environment gets more complex. In combinatorially complex environments where rewards are much more sparse, the chance of randomly hitting a reward become much lower, which makes the exploration bonus in those cases all that much more valuable. However, there are cases where the exploration bonus can cause problems. Namely, when we implement it in the Q-values, the overly optimistic bias of the Q-values can cause the agent to over-value suboptimal regions for longer than we would like to see. This is why it might be more favorable in larger, more complex environments to implement the exploration bonus in action selection.

8 Monte Carlo Conclusion

In this section experiment, we introduced an exploration bonus in both on and off-policy Monte Carlo models. In both models, we introduced the exploration bonus to uncover its impact on variance, convergence speed, and stability. While the models featuring the exploration bonus in the Q-values might produce optimistically biased Q-values, they can still be useful for what we want to do. Not only that but with enough episodes, the bias can diminish. However, the rate at which the bias diminishes depends on environmental complexity and other factors such as the exploration strategy and training time. This method can also speed up convergence and force the agent to be more aggressive in the paths it takes early on. However, if your goal is to learn the most accurate, reliable policy, the exploration bonus should ideally be in the action value selection. While placing the exploration bonus in the action values might slow down convergence, we end up with a more unbiased policy that is better for long-term policy optimization. This differs from our Dyna-Q where we found that placing the exploration bonus in the reward update that impacted the simulated model resulted in better results long term. This is because Monte Carlo is model-free and therefore, both exploration bonus implementations directly impact the agent's real-world experience rather than one of them existing only in the simulated experiences. Below is a summary chart of all models and exploration bonus placements.

9 Final Conclusion

After implementing an exploration bonus across several algorithms and in varying places, it's clear that the placement of the exploration bonus impacts the agent's exploratory capabilities, as well as its ability to find the optimal path. For future iterations of this experiment, it might be interesting to test this in more complicated environments, including continuous or combinatorially complex environments.

Policy Type	On Policy			Off Policy		
Exploration Bonus	**Baseline**	**Q-Values**	**Action Selection**	**Baseline**	**Q-Values**	**Action Selection**
Early Training Variance	Moderate	Moderate	A moderate amount of variance due to the exploration bonus.	High because of sampling importance	High, because of the exploration bonus and importance sampling	High, because of the exploration bonus and importance sampling.
Late Training Variance	Moderate (fixed epsilon value means that variance remains when compared to off-policy)	Moderate (because the exploration bonus continues to push the agent to explore)	Still a moderate amount of variance, but compared to other models, but decreased compared to early stage training.	Low, because epsilon decay forces the agent to become greedy.	Moderate, because epsilon begins to decay, but the bias stays around	Moderate, because epsilon begins to decay, but the exploration bonus still influences
Convergence Speed	Moderate	Faster, because it receives an optimistic boost from the exploration bonus	Relatively slow	Fast	Fast	Slower
Policy Stability	Stable (eventually settles)	Semi-stable (biased estimates, because of exploration bonus)	Less stable, because the agent continues to explore with the exploration bonus in the action selection.	Very Stable	Semi-stable (biased estimates, because of exploration bonus, depends on c parameter)	Semi-stable (not biased, but the agent is still exploring)
Q-Value Bias	No Q-value bias	Q-Value bias due to exploration bonus	No Q-value bias, because exploration bonus is at action selection	No Q-value bias	Q-Value bias due to exploration bonus	No Q-value bias, because exploration bonus is at action selection

Figure 12: Summary of model performance