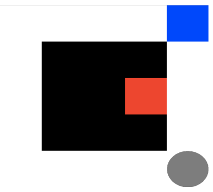
Two Interesting MDPs

MDP #1 (small)

In this first MDP world, the agent needs to get to the destination in a timely manner. There are two options presented in front of the agent:

1. Take the risky route, where the agent might slip into a hole and get injured.
2. Take the safe route, but this route is three times longer than the risky route.



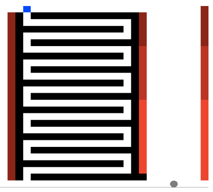
Figure[1.1]
MDP #1 (small)

Figure[1.1] shows the grid world for this MDP. This MDP is presented as a 5x5 grid world with 17 states. The gray circle is the starting state. The blue square is the goal state. The red square is the hazard (the hole). The black squares are the walls. The white squares are all possible states. There is a reward of -0.1 in white squares - the longer the agent takes, the less cumulative rewards. The goal state has a reward of +10, and the hazard has a reward of -5. In each state, the agent has 4 possible actions: up, down, left, right, and the agent would stay put if he runs into a wall. For each action the agent takes, there is a 80% chance of landing in the expected state, but there is a 20% chance of landing in any other adjacent states.

MDP #2 (large)

In this second MDP world, the agent also needs to get to the destination in a timely manner. Three options are presented in front of the agent:

1. Take the bridge right in front of the agent, which has hazards on both sides, and the hazards become more dangerous as the agent gets closer to the destination.
2. Take the insanely lengthy road on the left of the agent, which is absolutely safe.
3. Take the narrow cliff on the left edge of the world, where the agent has a chance of getting injured falling off the cliff.



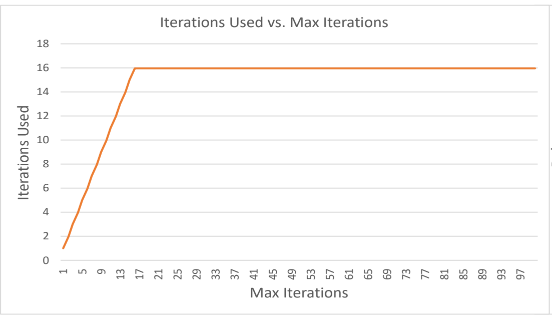
Figure[1.2]
MDP #2 (large)

Figure[1.2] shows the grid world for this MDP. This MDP is presented as a 27x27 grid world with 509 states. Most of the colored items in this world mean the same as they do in the previous grid world. However, in this one, there are three types of hazards, and they are presented with 3 shades of red. The darker the red, the more dangerous, with rewards of -1, -2 and -3. Similarly, for each white square, there is a -0.1 reward. The goal state has a reward of 10. The actions and transition functions are also the same as the previous grid world.

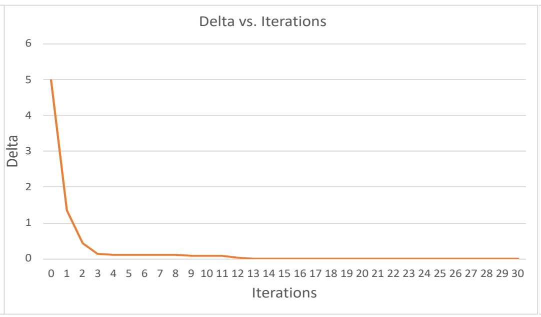
Given these two lively scenarios and the grid worlds, it would be interesting to see what policies the value iteration and policy iteration will generate, and if they line up to the initial expectation. It would also be interesting to see how q-learning behaves while solving these two problems. These two MDP models can easily map to real world path planning problems, such as automated driving in a known world or GPS route planning with the hazards representing something like heavy traffic.

Small MDP - Value Iteration

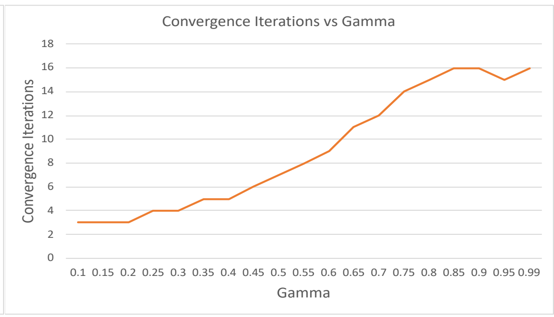
convergence: utility value change threshold set to 0.01



Figure[2.1]

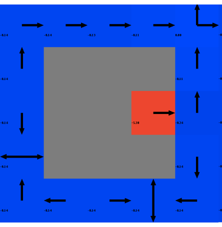


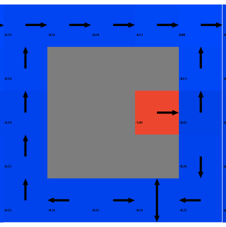
Figure[2.2]

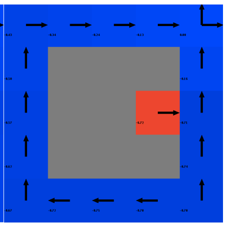


Figure[2.3]

Figure[2.1 - 2.3] represent a series of experiments performed to gain a good understanding of the value iteration's behavior because BURLAP library doesn't provide very useful details on how algorithms perform internally. Figure[2.1] shows how many iterations are actually ran given the max iterations. This graph shows that at gamma = 0.99 (gamma is the discount factor), 16 iterations are enough for the values to converge in this MDP world. In Figure[2.2], delta value in an iteration is the maximum change for any utility value. The delta drops more rapidly at first because the utilities are expected to change more dramatically in the beginning. The curve starts converging at around 4 iterations because the previous computations are enough to get the utilities to acceptable ranges. The computations after the 4th iteration are more like fine tuning the utilities. At the 16th iteration, the delta finally drops to 4.10E-04, which, according to the predefined threshold of 0.01, marks convergence. 0.01 seems to be a suitable threshold to mark convergence because not much change appears after this. Figure[2.3] shows that, given different gamma values, how many iterations are need to converge. The gamma value provides the agent a concept of “finite horizon”, which allows the agent to see a certain number of steps ahead, and helps reaching convergence. The larger the gamma value, the farther ahead the agent is able to take into consideration. The graph directly demonstrates that smaller gamma value does help convergence, but the policies converged to are different at different gamma. With a small gamma value, the agent is expected to be “short sighted” because it only sees what is next to it. With a large gamma value, although convergence takes longer, the final policies should fully reflect the entire grid world because the agent would have knowledge about the entire world (farther vision).

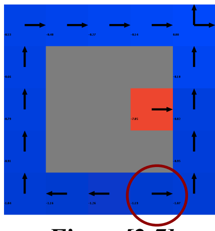

Figure[2.4]
gamma = 0.3


Figure[2.5]
gamma = 0.6


Figure[2.6]
gamma = 0.9

Figure[2.4 - 2.6] fully demonstrate what is expected. When gamma value is at 0.3, the agent seems extremely confused on the bottom and left side of the world because with the “short sightedness”, the agent cannot even see the goal nor the hazard. Without this important knowledge, agent cannot tell which side would be a better option as they have the exact same utility values. When gamma increases to 0.6, the agent is able to vaguely see the goal when it is on the left side of the world because there is nothing interfering with the “vision”. The agent is still not able to see the goal when it is on the bottom because the hazard there on the shorter route is interfering with the “vision” (rewards slightly cancel out), and the goal on the other route is just too far. Therefore the agent is still confused when it is at the bottom. When the gamma value reaches 0.9, the agent finally seems to have a clearer vision on the entire grid. The policies in each state seem reasonable and expected. The hazard does force the agent on the bottom to avoid the risk and choose the other route.

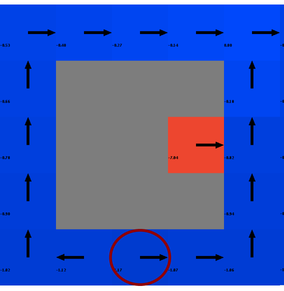
Figure[2.7] shows something interesting because when gamma is at 0.9, the agent seems to have enough knowledge to make reasonable decisions for every state. However, when gamma is changed to 0.99, a state around the bottom right region actually has a policy change. That state has the policy that avoids the hazard route at gamma = 0.9, but then changed to taking the risk at gamma = 0.99. This makes sense because within the “finite horizon”, the gamma/ discount factor essentially means how much to trust in the future. The higher the gamma value, the more a state would weigh the utility from a farther state. In this case, that bottom right state is trusting in the goal state more, and since the goal state has such a high reward, it actually results in a policy change in that state with such a minor gamma change. **The most reasonable and expected policies are resulted at gamma = 0.99 and takes 16 iterations to converge**.

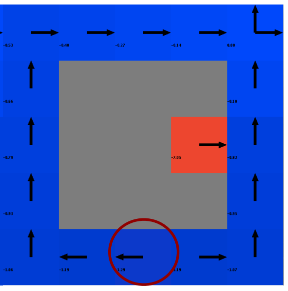

Figure[2.7]
gamma = 0.99

<u>Small MDP - Policy Iteration</u>

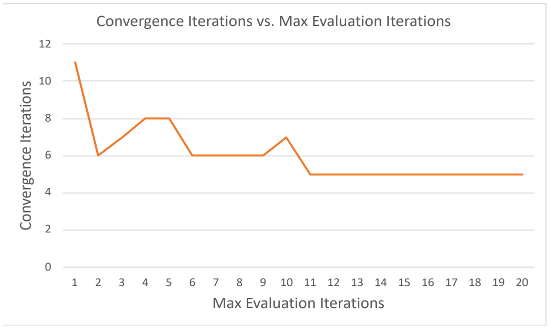
To better compare value iteration and policy iteration, the gamma is kept at 0.99, so the agent sees the world in the same way. Figure[2.1 - 2.3] will not be recreated for policy iteration because the behaviors observed are almost identical. This is because the policy iteration uses almost the same value update formula as the value iteration. The only difference is that the policy iteration only updates utilities based on one set policy in each state, whereas the value iteration updates utilities based on all possible policies. This difference ideally would make policy iteration have a better time complexity than value iteration because value iteration has a time complexity of O(S^2A) and policy iteration has a time complexity of

O(S^2). However, with such a small MDP, the time complexity difference doesn't make a noticeable difference.


Figure[3.1]
max delta = 0.103


Figure[3.2]
max delta = 0.102

Theoretically, to find the optimal policies, the convergence of policies is more important than the convergence of utilities. The convergence of policies is expected way before the convergence of utilities. Therefore, through experimenting different max delta values, which is the parameter marking convergence, the policy is found to converge at around max delta = 0.102. During this process of finding policy convergence, <u>the max evaluation iterations variable is kept at 1, which means for each set of policy, there will be only 1 iteration of utility updates based on this policy. Then the policies will change based on the new utilities.</u> (We will experiment with this variable later). Figure[3.1 - 3.2] demonstrates the final change in policies before convergence. The last policy changed is the one at the bottom, which is reasonable, because based on the analysis from before, this state would be one of the most confused states in this MDP.

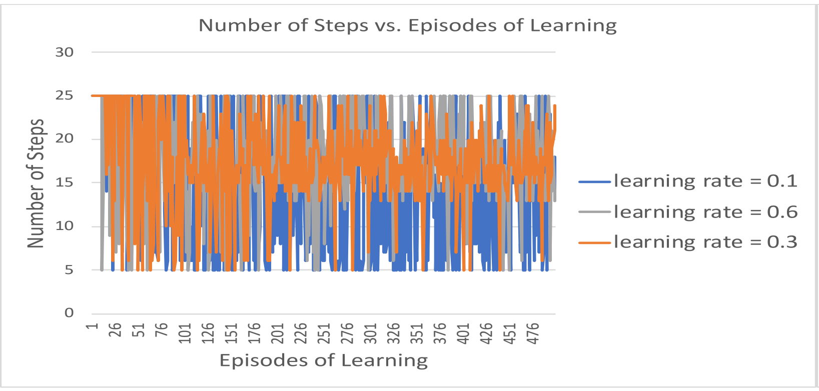


Figure[3.3]

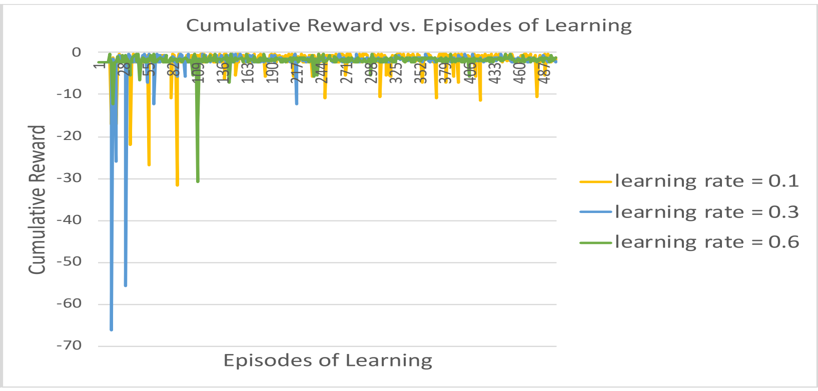
Figure[3.3] is a Convergence Iterations vs. Max Evaluation Iterations graph. The max evaluation iterations variable is explained above. The convergence iterations here are the policy iterations that results in policy convergence. All the resulting policies in this process are identical to the most optimal policies in Figure[2.7] and Figure[3.2]. Although this graph shows some noisy ups and downs, the general trend of the graph is still going down, which is expected. This is one of the important advantages of policy iteration where increasing the number of evaluation iterations would help the algorithm converge faster because more knowledge is gained with one policy iteration. The random increases of convergence iterations in the graph make sense as well. Given the same set of policies and different number of evaluation iterations, different utilities are produced. Completely different sets of policies can be evaluated from these different utilities in the next policy iteration. Based on the randomness of the initial set of policy, the initial utilities are prone to be biased in a certain level. Large number of evaluations on the initial set of policies might actually put too much trust in the initial policies, thus needing more adjustment later to counter this initial bias. Since the value iteration takes 16 iterations to converge to the optimal policies, it is intuitive and reasonable to set the max evaluation iterations also to 16 because this value is definitely enough without being unreasonably large. In Figure[3.3], the curve also seems to be at a convergence at max evaluation iterations = 16. **The policy iteration converges to optimal policies in just 5 iterations with max evaluation iterations at 16 and max delta at 0.102, which is much faster than value iteration**.

<u>Small MDP - Q-Learning</u>

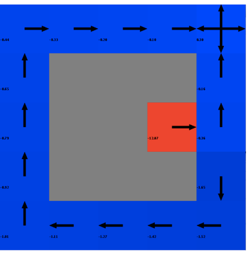
For the previous value iteration and policy iteration, the algorithm has knowledge about the entire MDP world, so they are essentially just planning the route without actually running in the MDP. Q-learning, on the other hand, doesn't know the reward and transition functions. It is a reinforcement learning algorithm that actually runs in the MDP episode by episode to learn the reward and transition functions through trial and error. In the end, use the knowledge gained to generate the most optimal policies.



Figure[4.1]



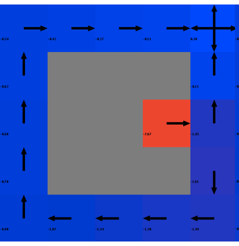
Figure[4.2]



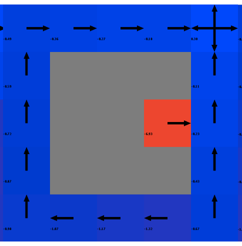
Figure[4.3]

Figure[4.1] is a Number of Steps vs. Episodes of Learning graph at three learning rate values. It is worthwhile to note that with these different learning rates, the policies generated are exactly the same (Figure[4.3]). The episode number indicates how many episodes of learning are taken place. After the episodes of learning, the optimal policies are evaluated. The number of steps indicates that after the learning, when the agent actually acts in the MDP once following the optimal policies, what the actual number of steps the agent takes to reach the goal. This can be a bit noisy due to the randomness of the transition function, which is why the curves are very noisy. For the three learning rates of 0.1, 0.3 and 0.6, learning rate of 0.1 (blue) seems to result in the worst performance as it fluctuates in a big range through out this graph. For both learning rate of 0.3 (orange) and 0.6 (gray), both curves have a similar behavior, which is that the fluctuations become more constrained and controlled as the episodes of learning increase. There are times when the agent keeps being unlucky and end up taking more steps and vice versa. Taking out the consideration for the extreme cases, the middle region describes the average performance of the agent in the MDP, which is around 15-17 steps. This makes sense because Figure[4.3] shows that the optimal path from the start to the goal is 12 steps, and with the uncertainty in the transition function, 15-17 steps would be a reasonable average performance. Although the orange curve and the gray curve have very similar behaviors, the orange curve shows a slightly more obvious convergence trend in the curve, and it fluctuates way less with the increase of episode numbers than the gray one, which suggests that the performance seems more stable when the learning rate is set to 0.3. Figure[4.2] shows the same as Figure[4.1], but instead of number of steps, Figure[4.2] shows the final reward the agent ends up with traveling from the start to the goal. Similarly, the behavior of the reward fluctuating in the beginning is expected, and the fluctuation becomes more controlled and constrained as the agent has more episodes to learn. Similar to Figure[4.1], learning rate of 0.1 (yellow) does result in the worst performance comparing to other learning rate values, and learning rate of 0.3 (blue) also looks slightly better than learning rate of 0.6 (green) in here because the blue one seems to be completely covered by the green curve, indicating the green curve fluctuates in a larger range than the blue one. This observation stays consistent with the observation from Figure[4.1].

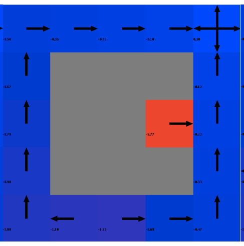
Compared to value and policy iterations, Q-learning has a worse performance and requires many more iterations, which is expected because Q-learning is in an unfair competition with value and policy iterations. Q-learning, without knowledge about the world, tries to learn in this world; whereas the value and policy iterations simply plan policies given all the information about the world. Looking at the final policies generated (Figure[4.3]), although different from the optimal, seem absolutely reasonable. The policies here tries to avoid the hazard more than the optimal policies from value and policy iterations. This is because during the Q-learning process, the danger of hazard is more emphasized. This can be resulted from a number of reasons: the initial policy is just biased towards the hazard first, the randomness of the exploration function, etc. Since the default exploration function only has an exploration rate of 0.1, if the agent finds first that there is a reasonable solution on the long route, the agent will keep exploiting this route instead of exploring more of other options, thus falling into a local optimum. In this case, modifying the exploration function should be able to optimize the result.



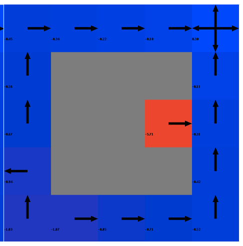
Figure[4.4]
epsilon = 0.3



Figure[4.5]
epsilon = 0.5



Figure[4.6]
epsilon = 0.6



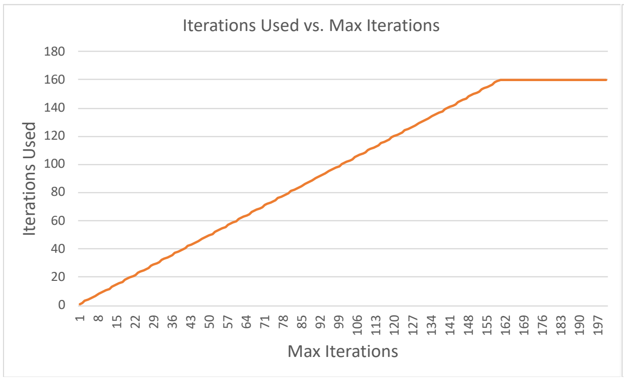
Figure[4.7]
random

Figure[4.4 - 4.7] demonstrates the results after the different exploration strategies are used. Figure[4.4 - 4.6] uses the epsilon greedy with different epsilon values. Higher epsilon values means more randomness while picking actions during learning. When epsilon increases to 0.3, the output policies still look the same as before(Figure[4.3]). Figure[4.5] and [4.6] support that epsilon values of 0.5, 0.6 help explore the space a lot better, and achieve very similar results to the optimal. This observation supports the claim above about local optimum, and epsilon of 0.5 and 0.6 help the algorithm escape local optima. Since Q-learning is based on a lot of randomness by nature, each time running would actually result in a different result. There are times that running Q-learning with epsilon at 0.5 actually result in the optimal policies. The exploration strategy of RandomPolicy (completely random) is used as well to observe the behavior if given more freedom. The outcome from this strategy seems slightly messier because of the unbalance of exploitation and exploration; this strategy is too unbalanced on the exploration side for this MDP.

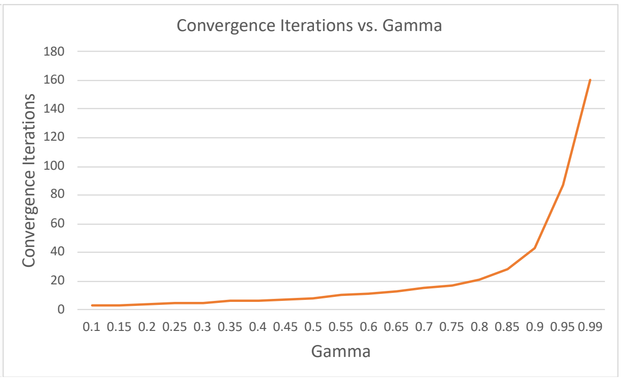
In conclusion, the Q-learning's performance is not bad at all compared to value and policy iteration given that it doesn't have knowledge about reward and transition function. The Q-learning is able to reach the same optimal policies given learning rate at 0.3 and a epsilon greedy exploration strategy with epsilon at 0.5 or 0.6. However, Q-learning requires way more iterations than value and policy iterations. The random exploration strategy is also experimented and doesn't seem to work better than the epsilon greedy strategy.

Large MDP - Value Iteration

convergence: value change threshold set to 0.01

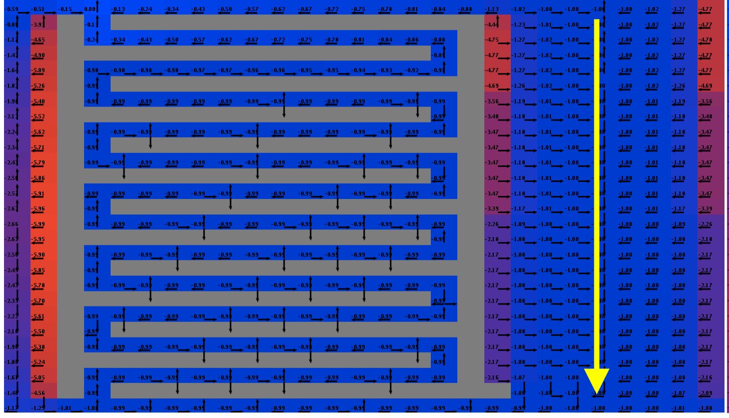


Figure[5.1]

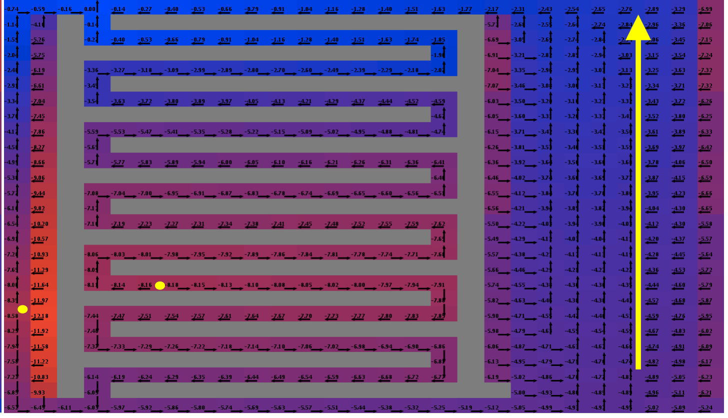


Figure[5.2]

Figure[5.1 - 5.2] are the same graphs as Figure[2.1] and [2.3], but for the larger MDP. Figure[5.1] illustrates how many iterations it takes for value iteration to converge in this MDP of 500+ states while the max delta is predefined to 0.01. This figure clearly shows that the value iteration converges after 160 iterations. Figure[2.2] was useful to understand the delta/ change in utilities at each iteration within value iteration. This figure is not recreated here for the larger MDP because BURLAP doesn't actually provide the delta for each iteration, so to collect the actual deltas, the debugger has to be used to manually step through iterations. 160 iterations are impractical to manually record using the debugger (Figure[2.2] only required 30, which was fine). There is really no point to recreate it either because the change in delta behavior is more specific to the value iteration algorithm than to a certain MDP, so even it is recreated here, the behavior would look almost identical to Figure[2.2] (just like how identical the curves are in Figure[2.1] and Figure[5.1]). Figure[5.2] describes the number of iterations to converge at different gamma values. Just like Figure[2.3], the iterations needed increase exponentially as the gamma value increases. However, in Figure[2.3] there is a slight converging behavior at the tail of the curve. This curve here seems to keep going up at the tail. This is because of the size of the MDP. As described before, gamma/discount factor describes how far ahead the agent can see or how far the "finite horizon" is for the agent. In the small MDP, a small gamma may already be enough to set the finite horizon equivalent to the size of the grid; whereas in a large MDP, even gamma 0.9 might not be enough to provide the agent with the vision on the entire world. On this grid, the start and the goal are at least 47 steps apart. According to the update function of value iteration, the reward of 10 at the goal worths $10 * 0.9^{47} = 0.0706965049$ to the start state if the gamma is 0.9. However, since $10 * 0.99^{47} = 6.23525394891$, gamma of 0.99 is actually a lot different from gamma of 0.9 in terms of how much the goal worths for the start state, which explains the boost of convergence iterations from gamma at 0.9 to gamma at 0.99. Figure[5.3] and Figure[5.4] provide the policies generated at these two different gamma values (might need to zoom in). The yellow arrows describe the main direction of the policies on the bridge, which highlights the impact

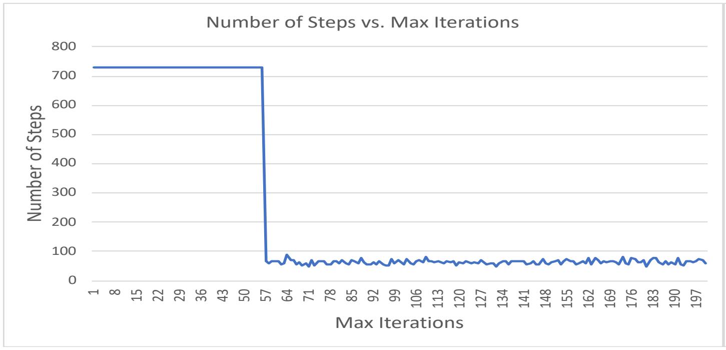


Figure[5.3] *gamma = 0.9*

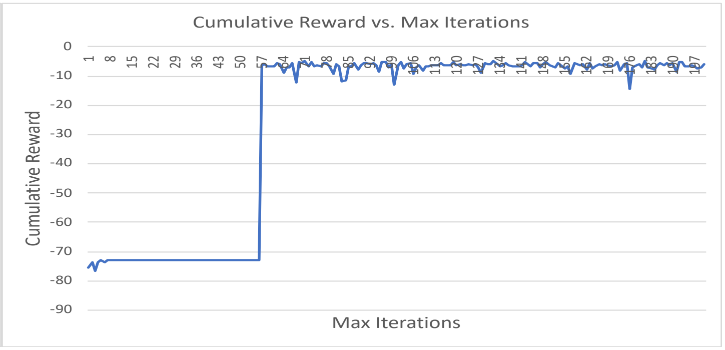


Figure[5.4] *gamma = 0.99*

of "vision" described above. the agent in Figure[5.3] doesn't want to go on the bridge because it can't see the goal until approaching the other end of the bridge; whereas the agent in Figure[5.4] knows that going on the bridge will provide a short and safe enough path to the goal, so it wants to go on the bridge from the start. This major change in behavior is consistent with the values calculated above with gamma at 0.9 and 0.99. In figure[5.3], the agent seems confused at the bottom of the grid and on the lengthy road because the gamma of 0.9 is not big enough for it to see the goal from these regions, therefore all the agent knows is that everything seems the same, so there really are no better policies or worse policies. The policies in Figure[5.4] are actually the most optimal policies in this world. The bridge seems to be the most optimal path as it is short and not that risky. The agent would even want to come to the bridge after it is already on the lengthy road. The yellow dot on the lengthy road marks the point of changing policies. Before the yellow dot, the road seems way too long, so the agent just wants to return to the bridge, but after the agent passes the yellow dot, it is too deep in the road to return. Similarly, The yellow dot on the cliff on the left edge of the world marks the similar behavioral change. After the agent passes the yellow dot, it is actually willing to take the high risk on the cliff because the goal is right at the end of this cliff.



Figure[5.5]

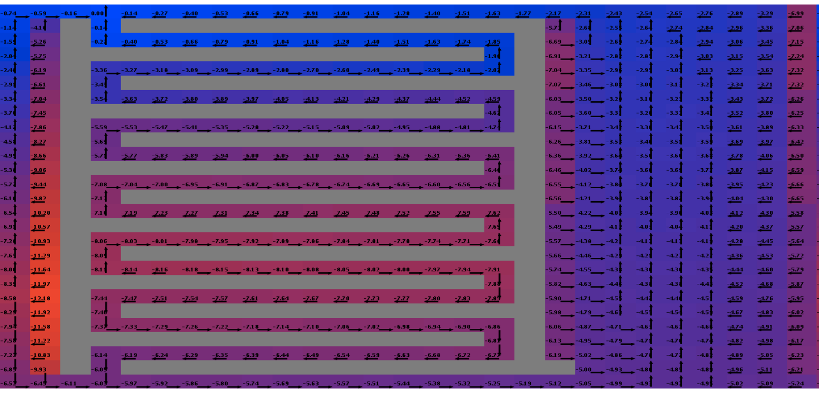


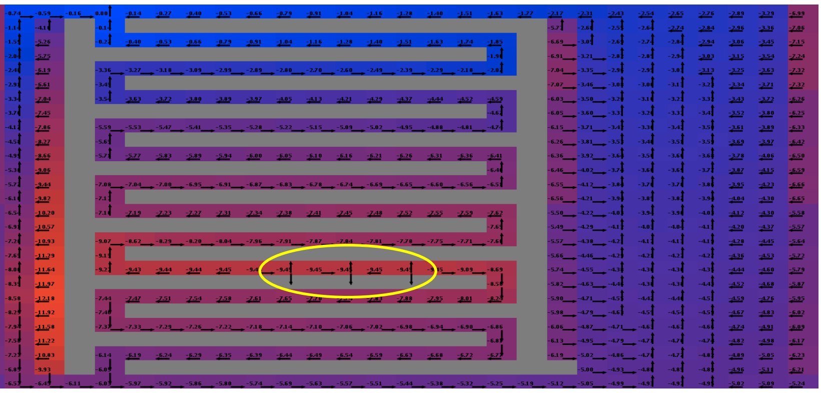
Figure[5.6]

Figure[5.5] and Figure[5.6] are the same graphs as Figure[4.1] and Figure[4.2], but for value iteration on this larger MDP. These were not created for the value iteration on the small MDP because, on such a small MDP, value iteration converges after just a few iterations, so the curves wouldn't show anything but noise. In Figure[5.5] and Figure[5.6], the curves seem to have similar behaviors in the sense of that the curves suddenly change at around the 57th iteration. The steps taken is constantly high initially, and after the 57th iteration, the number of steps suddenly drop to around the optimal. This actually reveals a lot about the process of value iteration in this MDP. The constantly high number of steps in the beginning means that the policy evaluated at the starting position is always going to a suboptimal path, such as the lengthy road. As more iterations take place, the policies start to become more reasonable, so the 57th iteration marks the change of policy in the start state from a suboptimal one to an optimal one. The sudden drop shows that, the rest of the optimal path has actually already converged, at least the bridge region, so as soon as the policy in the start state is evaluated to "up", the agent is scheduled to go on the optimal path.This means that the policies for the bridge region probably already converged at around 57 iterations, which makes sense because policies are known to converge earlier than values, especially when they are just part of the policies in this MDP. Figure[5.6] doesn't need more explanation because it is completely consistent with Figure[5.5] (Reward stays low until a policy change at the 57th iteration), and it strongly supports the analysis done on Figure[5.5]. **The value iteration is able to converge to the optimal policies in this large MDP with 160 iterations.**

Large MDP - Policy Iteration

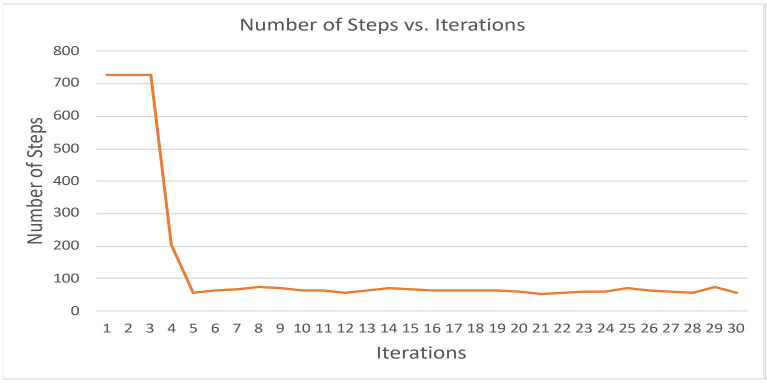
Based on the analysis for the policy iteration in the smaller MDP, and the knowledge gained about the behavior of policy iteration, the evaluation iteration is set to 160 because the value iteration converged in 160 iterations on this MDP, so it is definitely enough for the evaluation step in policy iteration (as analyzed before for Figure[3.3]) . Figure[3.3] would not be recreated here because to construct the graph, for each change in the evaluation iteration number, the experiment of policy iteration has to be rerun. Due to the size of this MDP, manually running the policy iteration for 100+ times is simply not feasible. On the other hand, since the effect of evaluation iteration number is already observed in Figure[3.3], and a corresponding value is already selected based on that observation, there is really no point of constructing this graph. Given the evaluation iteration is set to 160, and max delta is currently set to 0.01, the policy iteration is observed to converge at 25 policy iterations. Figure[6.1] shows the policies generated in the end of this process. These policies are actually identical to the ones generated by value iteration, which means that policy iteration is able to converge down to the most optimal policies on large MDP as well and the performance is even better because it only takes policy iteration 25 iterations converge; whereas the value iteration would take 160 iterations. Although the max evaluation iteration number in policy iteration is actually set to 160, there are rare cases that all 160 is used and these 160 iterations are much faster than those in value iteration because only one policy set is evaluated rather than all possible policies.


Figure[6.1]

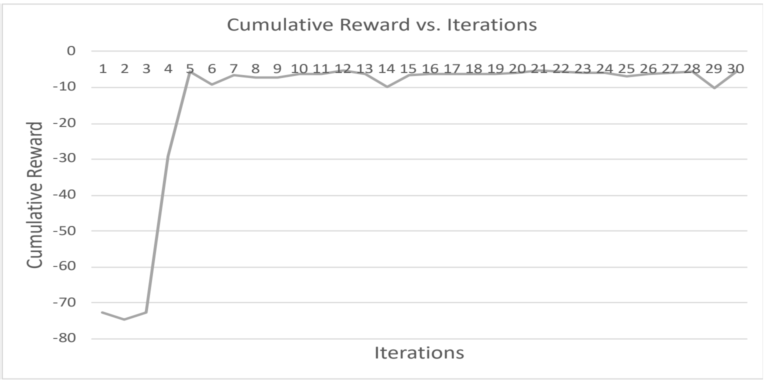

Figure[6.2] *Max Delta = 0.5*

Knowing the policy iteration is able to converge to the optimal policies just like value iteration, and also with way less iterations, the next step is to optimize this algorithm based on the knowledge of policies converge earlier than values, which is slightly mentioned in the previous section. Figure[6.2] actually marks the last step before the policies finally converge. When the threshold of delta is set to 0.5, there are only a few policies slightly off from the optimal solution, which are circled in Figure[6.2]. These are reasonably the last policies to converge because this is the most confusing region in the grid: the lengthy road doesn't seem that bad because the agent is already in it and it is also very safe, but going to the bridge is also good because the agent is not that deep in the lengthy road yet, and the bridge is the optimal path. When the threshold of delta is slightly changed to 0.4, the final policies are able to converge to the optimal ones in Figure[6.1]. This demonstrates that delta of 0.4 is enough to help the policy iteration converge here, which also supports that policies converge way before values.

Figure[6.3] and [6.4] are the same graphs as Figure[5.5] and[5.6], but with policy iteration. These are also extremely consistent with Figure[5.5] and [5.6]. For Figure[6.3], before around 4 policy iterations, the number of steps taken by the agent are constantly high because the policy at the starting state still leads the agent to an suboptimal path, as described before. However, only after 4 iterations, the policy at the



Figure[6.3]

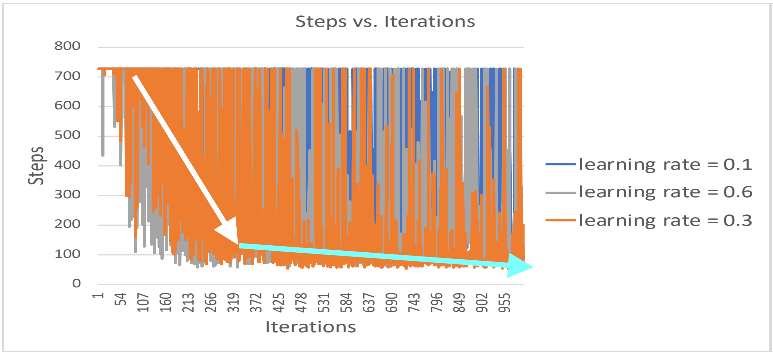


Figure[6.4]

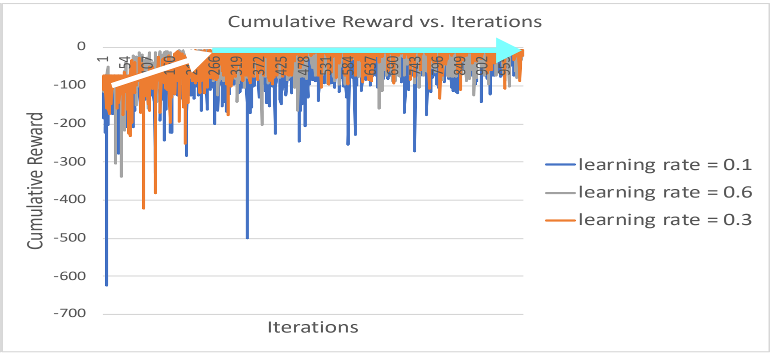
start states is able to lead the agent onto the optimal path. Note that it only takes the policy iteration 4 iterations to take the agent into the optimal path, but it would take the value iteration around 57 iterations. After the agent finds the optimal path, policies take 20 more iterations to converge. Just as described before, The first region to have the policies converge would be the bridge, and then it would be the cliff because there are not many states on there, and after the bridge has the optimal policies converged, the bottom states on cliff could start to make the decision to go to the bridge. After all, the lengthy road region would have its policies converged. Therefore, the later 20 iterations in the policy iteration would be handling the policy convergence on the cliff and the lengthy road. This analysis can be supported by Figure[6.2] because it shows that the states on the lengthy road seem the most confused, thus most difficult to converge. Figure[6.4], just like Figure[5.6], provides a strong evidence and support for the analysis on Figure[6.3]. Figure[6.4] also shows that there is an important policy change at around 4 and 5, that suddenly leads the agent to the optimal path. **The policy iteration with max delta at 0.4 and max evaluation iterations at 160 is able to converge to the optimal policies with just 25 policy iterations.**

Large MDP - Q-Learning

For this large MDP, the value and policy iteration are also able to converge to the optimal policies. Q-learning is then experimented to see if this optimal policies can be learned by reinforcement learning through trial and errors on the grid. Learning rate is set to 0.1, 0.3 and 0.6 for three separate experiments.



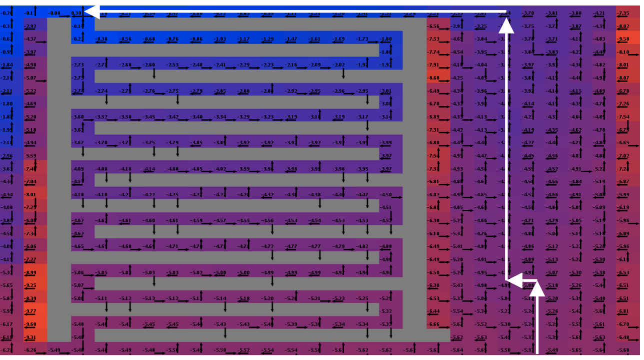
Figure[7.1]



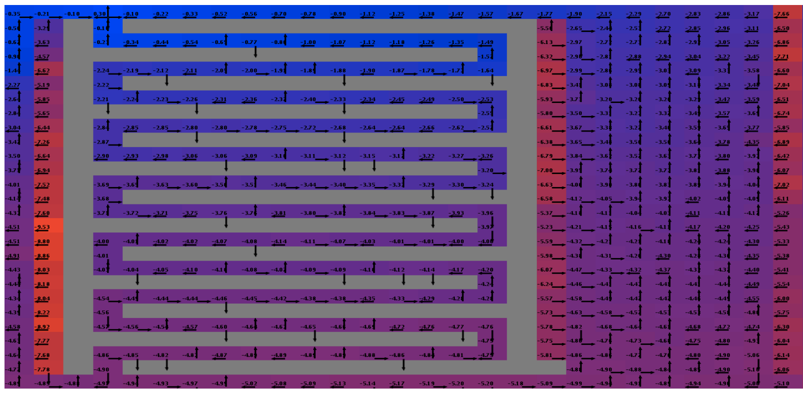
Figure[7.2]

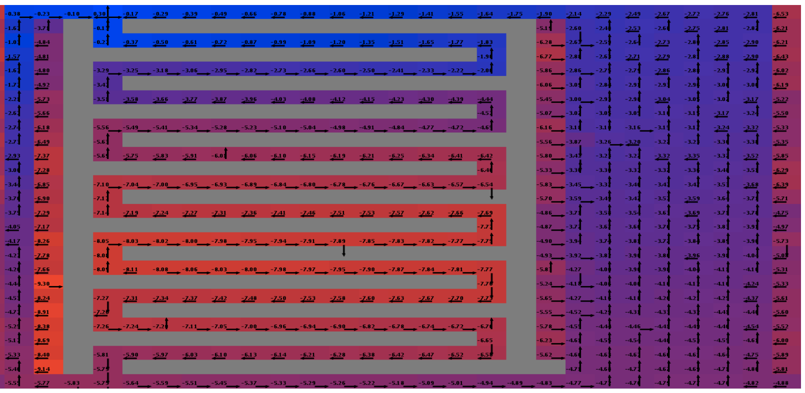
Figure[7.1] and [7.2] are the same graphs as Figure[4.1] and [4.2], but for a larger MDP. Figure[7.1] basically describes how many steps the agent actually takes to get to the goal from the start while using the policies learned given a certain number iterations of learning. Figure[7.2] describes what the cumulative reward the agent actually ends up with while using the policies learned given a certain number iterations of learning. First of all, to compare the results generated at different learning rate settings, the learning rate of 0.3 (orange) seems slightly better because this curve shows a more obvious trend of improvement in both figures, and the fluctuations on the orange curve seem more constrained and controlled at the tail than other curves, thus demonstrating improvement incurred by learning.

If the trends in Figure[4.1] and [4.2] are not very obvious due to the small MDP, the trends here should be pretty obvious. There are obviously two phases in the trends in Figure[7.1] and [7.2], marked with a white and blue arrow. The white arrow marks the trend in the first phase, where the agent is mainly exploring the MDP for better policies and paths, resulting in large improvement in the number of steps taken and cumulative reward collected. The blue arrow marks the trend in the second phase, where the agent is exploiting the current best policies, resulting in adjusting and fine tuning the policies. These behaviors are actually consistent with the behaviors from the previous figures in value and policy iteration, where the curves are also behaving in two phases. The behaviors in Figure[7.1] and [7.2] describe the process where the policies directing the agent onto the bridge can be learned relatively quickly, at around 300 iterations, but it takes an another 600+ iterations to fine tune these policies. Although the improvement for the second phase is not very apparent in Figure[7.2], the improving trend can be observed in the second phase of Figure[7.1].


Figure[7.3] *learning rate = 0.3*

Figure[7.3] shows the policies generated by reinforcement learning while the learning rate is set to 0.3 given 1000 iterations. The policies here make a lot of sense. The optimal policies from the start to the goal in this case are roughly described by the white arrows, which is almost identical to the most optimal ones. The policies on the cliff also seem roughly complete with some states slightly off, but the policies on the lengthy road just merely demonstrate how confused the agent is. The policy difference on these three paths once again support the analysis earlier about bridge policies converge before the cliff policies, and the lengthy road policies converge last. In this large MDP world, although the policies are not converged to the same optimal as value and policy iteration, the policies generated in Figure[7.3] are definitely reasonable and acceptable for a learner that doesn't have access to the reward and transition functions. The time taken for the reinforcement learning to run on this world is insanely long comparing to value and policy iteration. The iterations to converge here is also incomparable to the value and policy iteration because given these 1000 iterations, the policies still don't seem to converge yet, which is why some of the policies on the grid are still off and don't make sense such as the ones on the lengthy road).


Figure[7.4] *epsilon = 0.6*


Figure[7.5] *random policy*

Next, some other exploration strategies are explored to observe the different results. The q-learning in BURLAP is default to the exploration strategy of epsilon greedy with the epsilon set to 0.1. Two more strategies are tested: epsilon greedy with the epsilon at 0.6 and the random policy strategy. Figure[7.4] and [7.5] show the policies learned with these two exploration strategies. Figure[7.4] doesn't show significant improvement compared to Figure[7.3], where the lengthy road is still confusing, and the bridge and the cliff are both partially converged to optimal policies. However, Figure[7.5] shows a significant improvement at the lengthy road, where many of the states are converged to optimal policies, and the bridge and cliff regions are about just as good as in Figure[7.4] and [7.3]. this result is somewhat expected because in this large MDP, the lengthy road would be the most difficult to converge, and other regions are actually easy to converge since they are very straight forward about which way is optimal and which way is not, and there are just not that many states in these two regions. All algorithms seem to do well in these two regions. On the lengthy road, it is difficult for q-learning because the original set of policies on the lengthy road are definitely very messy. While running in the lengthy road, it is extremely difficult to receive useful feedbacks through exploiting the known paths. Every step would just respond with a -0.1 reward. What is needed to solve the lengthy road region is exploring instead of exploiting because exploring provides a higher chance to receive different feedbacks than exploiting, which is why the random policy strategy actually work better here than other strategies. **Q-learning is able to reach almost optimal policies with learning rate at 0.3 and random policy exploration strategy given 1000 learning iterations.**

Conclusion:

The two interesting MDPs describe very different decision making problems, yet under the same structure. Both of these can be mapped to multiple real-world problems as discussed in the first section of this analysis paper. The smaller MDP mainly functions as observing and understanding the behaviors of value iteration, policy iteration and q-learning. Then the larger MDP is used to actually test out the three algorithms to better understand and interpret the behaviors and results.

For the two MDPs, both value iteration and policy iteration are able to coverage to the same optimal answer, but the process is much slower on the larger MDP than on the smaller MDP. In both MDPs, policy iteration takes way less iteration to converge than value iteration. In the smaller MDP, policy iteration takes just 5 iterations to converge, while the value iteration takes 16 iterations to converge. In the larger MDP, policy iteration takes just 25 iterations converge, while the value iteration takes 160 iterations to converge. It can be concluded that the policy iteration converges faster than value iteration. As explained above, policy iteration converges faster because that policy iteration only updates the utilities given one set of policies instead of all possible policies like in value iteration, so the time complexity is much lower than value iteration. Also, it is observed that policies converge way before values converge, therefore, by nature, policy iteration can converge with less iterations than value iteration can. From the convergence number alone, it also looks like that the larger the MDP is, the better policy iteration performs compared to value iteration. Policy iteration just naturally seems intuitive for solving the optimal policies in MDPs because it emphasizes on policies and value iteration emphasizes on the utility values. However, if we need more details and insights from the actual utility values, value iteration is more useful.

Q-learning is selected as a reinforcement learning algorithm to solve these two MDPs. In the first MDP, it actually performs pretty well. It is able to learn the most optimal policies using learning rate of 0.3 and the greedy epsilon exploration strategy with epsilon at 0.5 or 0.6. In the second MDP, although it is not able to learn the most optimal policies for all states, it is able to converge to very reasonable and almost optimal policies using learning rate of 0.3 and the random policy exploration strategy. This result, in the case of such a large MDP, is outstanding. However, in terms of both iterations and clock time, q-learning takes much longer time to run than other two algorithms because, essentially, the other two algorithms are merely path planning algorithms with all the information about the reward and transition functions and q-learning is a learning algorithm that has to actually run in the MDPs to learn the reward and transition functions. From a pure objective perspective, q-learning does have worse performance than the other two algorithms. Different exploration strategies are also experimented with q-learning in both MDPs: the epsilon greedy strategy with different epsilon values (different epsilon values actually incur very different behaviors) and the random policy strategy. Epsilon greedy works better than random policy strategy in the smaller MDP, and the random policy strategy works better than any epsilon greedy strategy in the larger MDP. In the smaller MDP, epsilon greedy seems to have the best performance when epsilon is at 0.5 or 0.6, where the exploration and exploitation are the most balanced. In the larger MDP, random policy strategy has the best performance because in such a large grid, exploring is way more important than exploiting. It definitely wouldn't be optimal if the learner starts to exploiting before the optimal paths have even been explored.