Reinforcement Learning

A study on Value Iteration, Policy Iteration & Q-Learning in Various Grid Worlds

Dudon Wai, dwai3
Georgia Institute of Technology
CS 7641: Machine Learning

Abstract: *This paper explores Value Iteration, Policy Iteration and Q-Learning and applies these three reinforcement learning algorithms to Markov Decision Processes that model Oil & Gas Drilling and Ikea Floor Plan Design.*

INTRODUCTION

This project will explore various reinforcement learning techniques an agent can use to make decisions. An analysis of two interesting Markov Decision Processes (MDPs) is conducted using two planning algorithms, Value Iteration and Policy Iteration, and one learning algorithm of choice, Q-Learning. The two MDPs are selected to demonstrate the different behaviors of reinforcement learning for MDPs with "small" and "large" numbers of states.

This report is organized into 4 sections. The first section, ***Introduction,*** provides a background on MDPs and the three algorithms that will be explored in the following sections. The second section explores the first problem, ***Drilling MDP,*** which is comprised of 3 subsections as outlined in the requirements. Part 1 corresponds to the project requirements, which presents Drilling MDP and describes why it is interesting for machine learning and industry application. Part 2 applies the planning algorithms to Drilling MDP and presents the performance results. Part 3 applies the learning algorithm and presents the performance. The third section explores the second problem, ***Ikea MDP,*** which is also structured into 3 parts. The final section, ***Comparative Analysis & Conclusion,*** amalgamates the findings and presents the behavior of the algorithms to understand the effects of discount factor and iterations on the policy, final reward and computing time.

Markov Decision Processes

As an overview, an MDP is defined as **MDP = (S,A,T,R,γ)** where:

- S (States): A set of all possible states, such as location in a 2D grid
- A (Action): A fixed set of actions, such as Up, Down, Left, Right
- T (Transition): The probability to go from one state to another for a given action
- R (Reward): The value of a given state and/or action
- γ (Discount Factor): Reduce the importance of future rewards, affecting how the MDP will plan

With the MDP defined, the algorithms can be applied to generate policies based on the expected reward of a set of states or actions, making improvements and iterating until improvements are no longer seen. MDPs are particularly useful in situations that require large sequences of actions, and where the actions are not entirely reliable and decisions are better made using expected value.

Reinforcement Learning Techniques

Value iteration works by applying the Bellman equation to evaluate progressively state-by-state until the solution converges. Initially, the utility is not known except for the utility of immediate reward. Using the immediate reward at the terminal state, the utility of the nearest states are calculated based on discounted reward, and this iterates until the utility of all states are known.

Instead of determining the values of all states, **policy iteration** creates an arbitrary policy and evaluates the expected value at all states under that policy. The method explores whether the value at a certain state can increase by modifying the policy (changing an action), and iterating until there is no improvement, after which it continues through the policy. This method also converges, however requires more calculation as it solves a set of linear equations when determining the expected value of all states.

While value iteration and policy iteration use *a priori* domain knowledge as the basis of the transition function, **Q-Learning** is model-free. The method works by assigning all states a Q-value, and visiting each state to re-assess the Q-value based on immediate and delayed rewards. This is similar to the concept of exploitation and exploration in random optimization. Q-learning strikes a balance between executing on immediate rewards and learning to explore delayed rewards. In the early stages, Q-learning will explore more and as it progresses, begin to exploit more.

Algorithm Implementation

The implementation of MDPs utilized existing Java code [1], which was adapted from original source code in the BURLAP Reinforcement Learning package [2]. The behaviors of the three reinforcement learning algorithms were explored, using various parameters (shown below on the right) to observe the impact on the convergence, computation

time and policy. The results were exported to CSV files and analyzed using Excel to explore trends and generate plots. Each experiment was run 3 times, and the average is reported.

The "borrowed" code contains similar examples to the BURLAP Grid World. The examples maintain the ideas of a cost per action, and a terminal reward, "real world randomness" and the goal of navigating from the bottom-left to the top-right. The examples are different Grid Worlds, of various sizes and objects/walls to add complexity. The selected examples for this project are called **Drilling** and **Ikea**. The following sections will discuss the problems in further detail.

Parameters
Domain (GridWorld)
Reward Function (100)
Cost Function (-1/action)
Terminal State Function (Bottom Right)
Discount Factor [0.1, 0.99]
MaxDelta (-1)
MaxIterations (500)
ActionProbability (0.8)

The original BURLAP package includes a Grid World example (Figure 1) in which the agent exists in a 2-dimensional plane. The **datum** is in the bottom-left corner (0,0) and has the choice of 4 **actions** to move in any cardinal direction to navigate the plane and avoiding walls (in black) to find the **terminal state** (blue). The agent is encouraged to reach the terminal state by forcing it to make an action, where all actions have a small negative **cost function** (for example, -1 per action) except when the terminal state is reached, in which a large **reward** is awarded (for example, 100). The objective of the agent is to maximize the net reward by reaching the terminal state in the fewest number of actions. As a way of modeling the "real world", BURLAP can make the actions **probabilistic** (instead of deterministic) where the agent's individual actions do not always have the intended consequence. For example, when the agent wants to go Up, there is an 80% chance the agent will actually go Up, but a chance to go Down, Left or Right (the remaining 20% is split equally 3 ways). To model the agent's priorities of the present and future, a **discount factor** is applied to the expected value of each action. **MaxDelta** allows the algorithms to end when the delta in utility between consecutive iterations is below a threshold. This was not used, as I=500 iterations were applied for each test to explore the behavior of the three algorithms across the two problems.

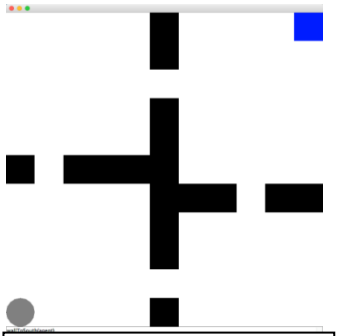


Figure 1: Original GridWorld

Specifically in Q-Learning, there are additional parameters to consider. The **Q_Initial** value is the value assigned to all states before updating the values with "truth". When Q_Initial is high, this initially makes all states viable for the agent to pursue, whereas when Q_Initial is low, all states are less viable and thus the agent is encouraged to exploit more than explore. The **exploration strategy** determines how the agent decides whether to explore or exploit. The decision process is a function of the approach (such as epsilon greed, random, or a form of simulated annealing) and the value of **epsilon** ε (a threshold for random generation below which an agent chooses to explore). Lastly, the **learning rate** is a relative measure of the importance of recent and older information when an agent makes a decision.

DRILLING MDP

Part 1: An Interesting MDP

Introduction

The purpose of the Drilling MDP was to explore the decision-making behavior of an agent when there are important "touch points" within the domain before reaching the terminal state. "Touch points" refer to states other than the terminal state that are beneficial to the agent such as multiple rewards. The Drilling MDP has other clear real-world applications in navigation – whether that is in trade, transportation, or computer gaming (such as first-person shooters).

This example is particularly interesting because of my work as a petroleum engineer. I develop plans to drill horizontal wells (see the picture on the right), which are characterized by:

Vertical section (0 to 6,000ft):
- Utilize the advantage of gravity while drilling and geological predictability

Build section (6,000ft to 9,000ft):
- Gradually redirect the well by building **inclination angle**
- ~3,000ft in length, and adds ~1,000ft in vertical depth

Horizontal section (9,000ft to 18,000ft):
- Maintain a flat profile through the **pay zone** or **target**
- This section of geology contains a large amount of hydrocarbons
- This section should stay within 10m in vertical depth (up or down)

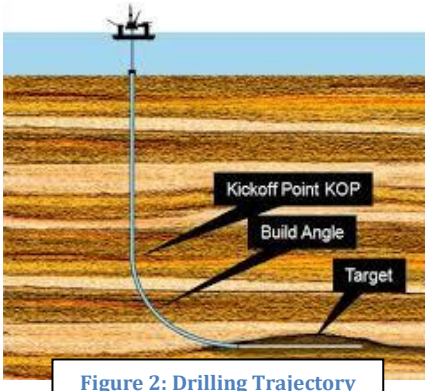


Figure 2: Drilling Trajectory

The objective for horizontal drilling is to optimize **a)** the hydrocarbon production with **b)** the well cost and integrity. The length of the horizontal section dictates the hydrocarbon production, while the well design (trajectory, equipment, protective barriers) and services determine the cost and integrity.

Figure 3 shows a bird's eye view of Canadian drilling activity (via Accumap, public software). This is an MDP problem because in land drilling, a single rig drills over 50 wells per year. The design and execution of each well is passed on to the next well, with minor improvements in design (or **policy**) to either improve production, reduce cost, or increase integrity. The Drilling MDP (lower right) is designed for a drilling rig (**agent**) to successfully drill to the pay zone or **reward**, while reducing the cost of getting there (**cost function**).

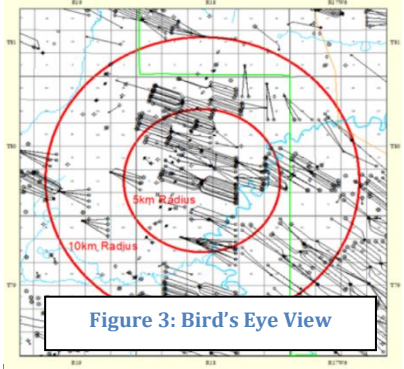


Figure 3: Bird's Eye View

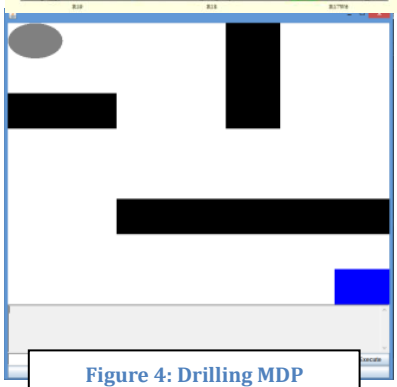


Figure 4: Drilling MDP

Implementation

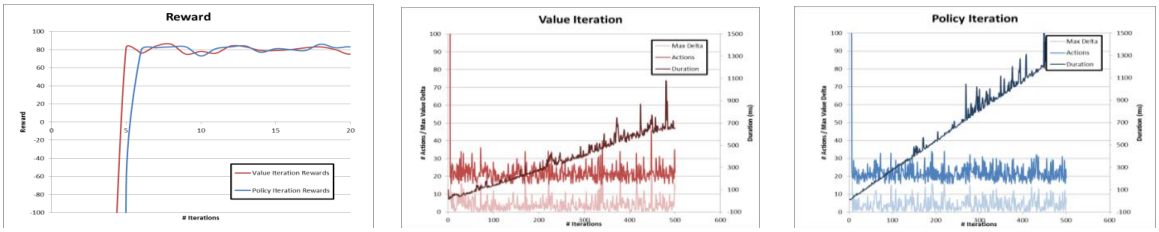
This MDP is a small 7x8 grid (56 states) in the vertical plane, similar to the cross-section of the earth, where the grey agent starts at (0, maxY) and tries to move to the blue target (maxX, 0). The black sections (10 states) represent areas of geology that are off-limits to drill (for example, impenetrable rock, or areas of high and unsafe pressure). To encourage the agent to maximize the horizontal length, black sections were used as **domain knowledge** to ensure the agent goes down vertically first. Another method to encourage this behavior is to spread out the reward along multiple states in the horizontal; however this simpler model was selected for more relevant comparison with the Ikea MDP.

As shown in the table in Algorithm Implementation, the cost per step is -1, the terminal reward is 100 and the actions are probabilistic. By default, the discount is 0.99, however the impact of this will be explored in a later section.

Part 2: Value Iteration & Policy Iteration

The following section presents the MDP behavior using a discount factor of 0.99. The results will be discussed in further detail, and as an overview, the Value Iteration (VI) and Policy Iteration (PI) share similar behavior in the number of iterations to reach convergence, the policy development, and the reward and number of actions at convergence. However VI and PI differ in the duration per iteration by a factor of 2.

Figure 5: Value Iteration and Policy Iteration Results Overview



Value Iteration

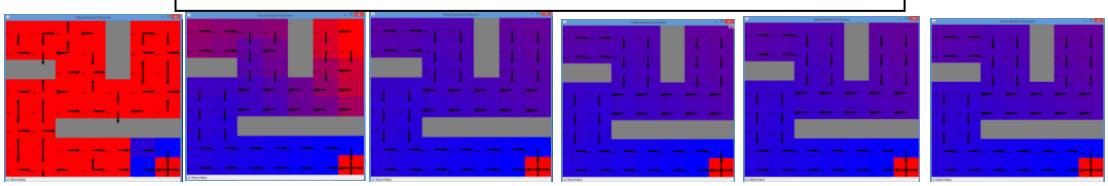
The Reward graph above shows that that Value Iteration had a large negative utility for the first 4 iterations. This is also reflected in the number of actions (>1000) in the initial process before converging very quickly at I=6. The iteration was determined by calculating the limit of the reward, 80.3, and identifying when two consecutive reward were within 5% of the limit (76.2). This can also be seen in the middle graph, where the delta and the number of actions drop to constant levels. It is interesting that the delta and the number of actions correlate, which demonstrates the model behaves as expected (since each step costs -1, and the delta will change by 1). It is also interesting that there is some noise in the delta, which is explained by the probabilistic nature of the problem. This accurately portrays drilling operations, as variances in the underground pressures (in the form of fluid pressure or rock stress) pose directional challenges to the drilling equipment.

The duration (dark red) grows linearly with the number of iterations, which is expected since the time will increase linearly with the increased number of actions. The small (and sometimes large) spikes in time are again, due to the probabilistic nature of the agent's actions.

The policy constructed using VI is shown below for Iterations 1, 10, 20, 100, 200 and 500. There are several observations to be made of the converged policy at I>20. Although the Reward converged at I=6, the policy for I=10 has demonstrates that changes in policy remain that have minimal impact on the Reward. As expected, the converged policy guides the agent away from the top right corner.

Observing the policy at I=1 and I=10, one notices how the algorithm learns. At I=1, the nearest states to the reward are evaluated for utility. VI continues to evaluate nearby states and at I=10, it begins to diverge into two paths - upwards and to the right. The policy at I=1 also shows the arbitrary actions (red) that VI has not yet reached and updated. A subtle observation is at I=10, where VI first ensures the agent avoids walls before refining the specific path it should take. This is because the cost of -1 is applied and it does not make progress towards the reward.

Figure 6: Value Iteration Policy Development



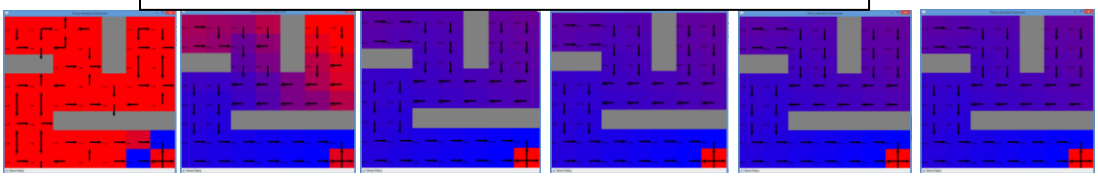
I =1 I=10 I=20 I=100 I=200 I=500

Policy Iteration

The Reward graph above shows that that Policy Iteration also converged to the same reward as VI, but slightly slower at I=7. PI also takes longer to perform iterations – almost exactly double. While this agrees with the expectation that PI takes longer per iteration, it is also expected that PI takes fewer iterations because PI evaluates the utility of each state/action pair for a given policy, which takes more time to compute but provides more information to learn from.

The additional iteration may be explained by the use of averaging only 3 runs, the probabilistic nature of the MDP or the initial states assigned prior to the first iteration. Overall, PI behaves very similarly to VI, also seen in the noise profile (+5/-5 reward) when the methods converge, and as seen in the policies. The methods converge to the same policies for I>20, and even the policies at I=1 and I=10 look nearly identical (slight difference along the bottom). It is interesting that VI shows more blue around the terminal state than PI. This suggests VI is closer to the converged solution than PI at I=1, explaining the fewer iterations. A further study on the initial state assignments is recommended.

Figure 7: Policy Iteration Policy Development



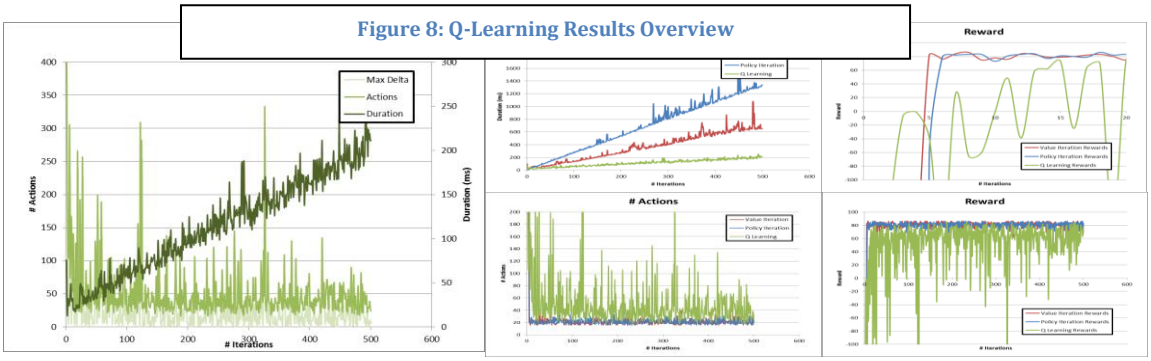
I =1 I=10 I=20 I=100 I=200 I=500

As discussed, VI is faster per iteration (as expected) and converges faster by 1 iteration (not as expected). However, 1 iteration is relatively small, and suggests further study into the way convergence is determined (which is when two consecutive runs are within 5% of the reward limit), the effect of probabilistic actions and the number of runs to produce accurate trends. Both methods converge to the same answer and comparison with the Ikea MDP may show how the number of states influences the performance of each method.

Part 3: Reinforcement Learning Algorithm of Choice

Q-Learning is a model-free algorithm that assigns a Q-value to all states and iteratively modifies a policy to exploit the immediate rewards and explore delayed rewards. The Q-Learning results for the Drilling MDP are below.

Figure 8: Q-Learning Results Overview

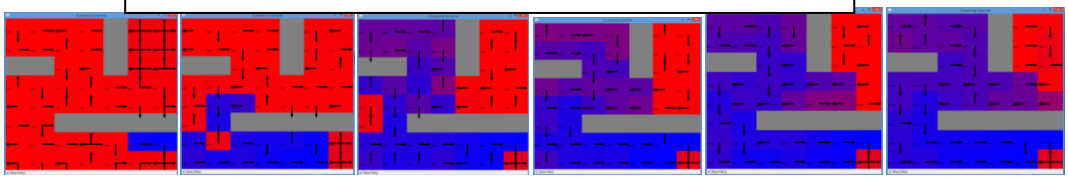


As the left graph shows, Q-Learning has behavior somewhat similar to VI and PI, such as the flat profile for I>100, and the linear increase in duration as the number of iterations increase.

There are a number of differences in QL that are not observed in VI or PI. Observing the Reward graph in the upper middle, VI and PI converge at Reward of 80, while QL has an average around 60, with significant drops in Reward during exploration. The upper right chart, showing iterations 0 to 20, shows the exploratory nature of QL, and also that it converges more slowly. The criteria for QL convergence was to approximate the limit of the converged utility using the average of utility between I=401 and I=500. This produces a converged utility of 60, and the iteration for convergence is I=14 (defined as when the utility for two consecutive iterations is within 5% of the converged utility). While this is double the number of iterations than for VI or PI, the time per iteration is considerably less. In fact, considering overall clock time to reach convergence, QL is faster than PI. This is explained by the model-free characteristic of QL, which frees the method from calculating the action-utility pairs, like in VI and PI.

Considering that QL does not have the benefit of knowing the model, rewards and cost function, it performs quite well. It is interesting that even at I=500, QL continues to explore policies that produce utilities below 50. The policies for I=1, 10, 20, 100, 200 and 500 are shown below. Unlike for VI and PI, the policies vary even for I>20, however the utility map generally remains unchanged. The "creativity" of QL is seen in the way it never settles on the policy seen in VI and PI. This explains why the total utility per iteration averages to 60 instead of reaching the 80 value. It is interesting that QL attributes the top right corner with very low Q-values (shown in red), to indicate that QL does not learn anything new during exploring the immediate and the delayed rewards of that quadrant.

Figure 9: Q-Learning Policy Development



I =1 I=10 I=20 I=100 I=200 I=500

IKEA MDP

Part 1: An Interesting MDP

Introduction

Marketing plays a role in how a customer interacts with a product, from awareness to purchase to utilization. This is particularly interesting for me because of my background in social media and online advertising (Google AdWords, Facebook Ads), where I measure the interactions people have with a website or advertisement, and examine the factors behind certain behaviors. For this MDP, I will explore how marketing design can manipulate how customers interact with products – not online, but offline in stores instead. The Ikea MDP was chosen as a more intuitive and widely relatable example of marketing design

Ikea is a world-wide chain of large Swedish furniture store. It is known for its floor plan design (see on the right) that maximizes the time customers spend in the mall. Even if a customer is looking for a specific office desk, the Ikea shopping experience takes the customer through specially crafted displays of kitchens, bedrooms and dining rooms in order to find the office furniture. The purpose is to increase the exposure that customers have with various products, to increase their chance of making additional purchases [4]. However, Ikea understands that some customers may get very frustrated by this, and they subtly include shortcuts between showrooms for customers who are specifically looking for a faster path.

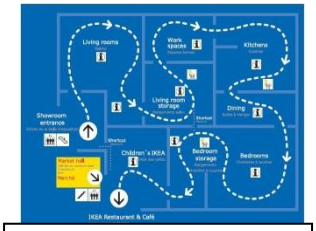


Figure 10: Ikea Floor Plan

The Ikea MDP models an agent navigating through the Ikea store, with a choice to take the wider path or the narrow shortcut straight to the terminal state. The expectation is that, at first, the agent follows the wider path because

Implementation

The MDP is a modification of the Drilling MDP, where a similar GridWorld is used but instead of a vertical cross-section of the earth, this is a 2D bird's eye view of the Ikea floor plan. This MDP is an 11x11 grid (121 states) with 26 walls for a total of 95 possible states, about double the number of states as the previous MDP problem. The agent begins at the front of the store at (0,0) and it finishes shopping when it reaches the terminal state at (maxX,0). The same Reward (100) and Cost function (-1) are carried over from the previous MDP, however further analysis may show interesting changes in behavior when these parameters are changed. The same discount factor of 0.99 is applied, and the probabilistic nature of the agent is maintained at 80%.

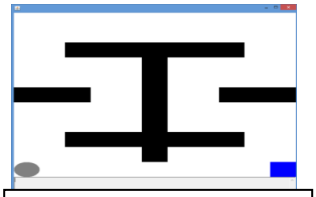
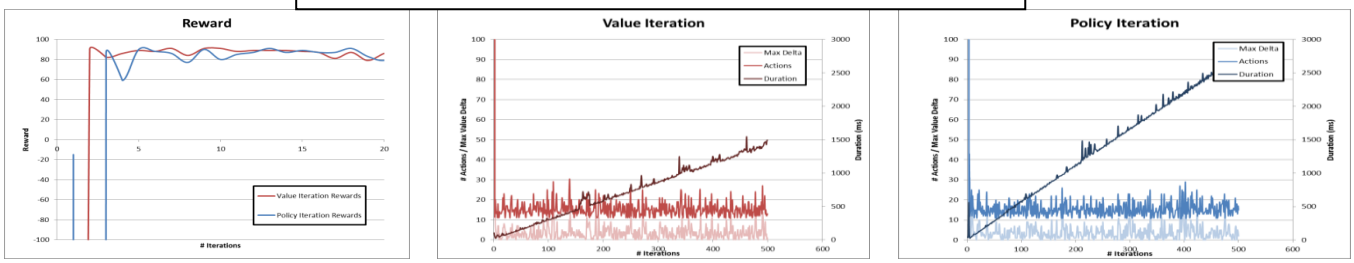


Figure 11: Ikea MDP

Part 2: Value Iteration & Policy Iteration

The following section presents the MDP behavior using a discount factor of 0.99. While VI and PI behave similarly for convergence, reward value and number of actions, the duration per iteration for PI is double that of VI.

Figure 12: Value Iteration and Policy Iteration Results Overview



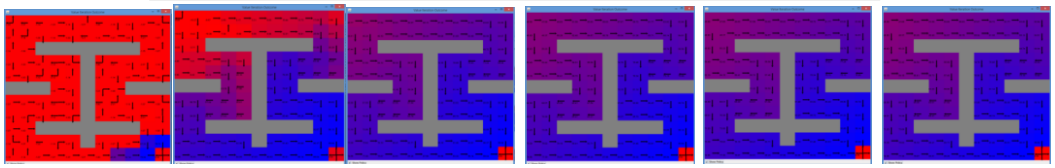
Value Iteration

The Reward graph above shows that Value Iteration had a large negative utility for the first iteration, which is expected as it has not updated the policy based on expected values. By the second iteration, VI achieves a total utility of 85 (averaged from 3 runs) which is within 5% of the average utility (for I=401 to I=500) of 87. Therefore, convergence is considered at I=3, when the total utility is within 5% for two consecutive iterations. Once VI has converged, it stays within 5%, fluctuating because of the probabilistic function of the agent's actions.

The duration of the first iterations are between 30 and 100ms, and this grows linearly to 1500ms at I=500. This is as expected, since the VI must make calculations for each action in the policy. The small wavering along the trend line is indicative of the probabilistic actions of the agent.

The policy for I=1, 10, 20, 100, 200 and 500 were constructed and are shown below. The display of I=1 shows the behavior of VI, which initially assigns actions arbitrarily, and calculates the expected value to create the policy backwards from the terminal state. Since VI converges at I=3 (from a total utility perspective), it has found the optimal path however it has not converged on a policy until I>20. This can be seen at I=10, where the top-left corner is not yet evaluated. It was expected that the wider path would encourage the agent to consider exploring the upper route, however it is clear that the agent quickly finds the shortcut and avoids the longer path entirely. For further study, the domain, cost and reward structure can be modified to encourage the agent to navigate the other route.

Figure 13: Value Iteration Policy Development



I =1 I=10 I=20 I=100 I=200 I=500

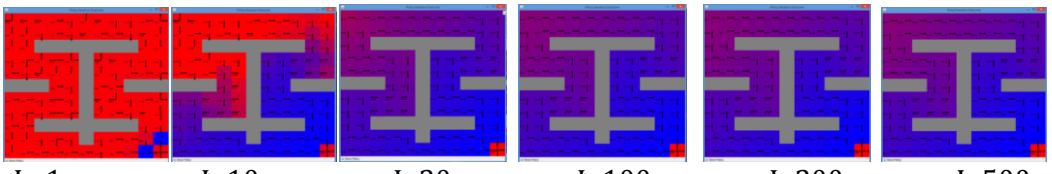
Policy Iteration

The Reward graph also shows that Policy Iteration performs very well too. PI meets the convergence criterion of falling within 5% of the limit (87) at I=4. The time it takes to run each iteration is approximately double that of VI, which is consistent with the understanding that, per iteration, PI re-evaluates the utility at all the states for a given policy. While initially expected that PI would converge in fewer iterations than VI, it is likely the problem design that favors VI.

While there are 95 possible states, the optimal path is straight along the bottom (11 states total, including initial and terminal). While this example was meant to be a problem with a "large number of states", it is clear that the number of ***reachable*** states does not necessarily have an impact on the complexity of the problem. The complexity of the problem depends on the number of states along and around the states the agent explores. Also, the number of possible (and comparable) paths encourages the agent to explore more of the domain. While the MDP was designed to encourage the agent to explore, the difference in utility between options is too great for the agent to pursue the top path. As mentioned in the VI analysis, the algorithmic behavior can be studied further with different a domain, cost and reward.

Comparing the policies for various iterations, the behavior is very similar to VI. Initially, the arbitrary policy produces poor results, but by I=10 the algorithm has mapped out half of the domain, which includes the initial state. At this point, the algorithm has found the local optimal path, and continues to explore the rest of the domain to confirm it is the global optima. By I=20, the domain is fully mapped and the optimal path is confirmed, and does not change for I>20. A slight difference from VI is seen at I=1 and I=10, where fewer states are evaluated than in VI. This explains why VI converges slightly sooner than PI.

Figure 14: Policy Iteration Policy Development



I =1 I=10 I=20 I=100 I=200 I=500

Between the two algorithms, VI converges faster than PI and the time per iteration is also shorter. In this example, VI performs better overall, producing the same result as PI using less resources.

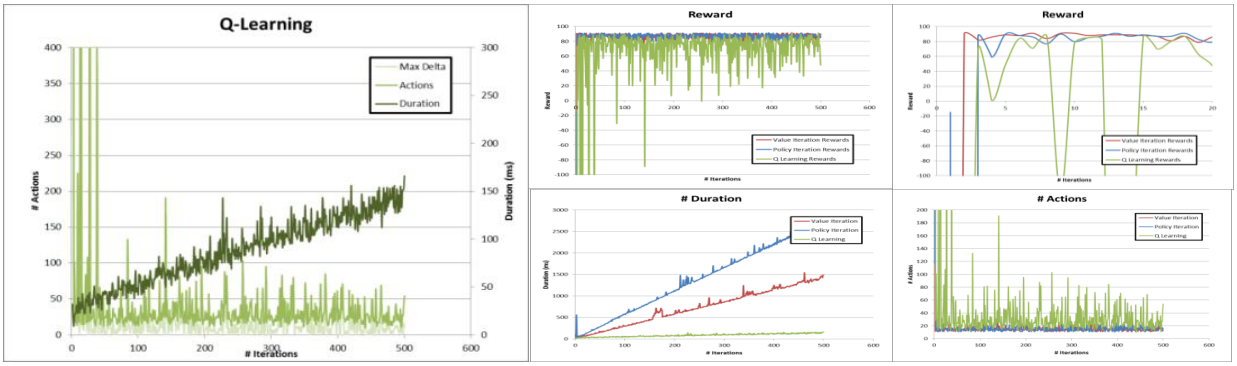
Part 3: Reinforcement Learning Algorithm of Choice

Q-Learning

As the left graph shows, Q-Learning shows a similar flat profile for the number of actions and the reward delta at I>50. Also, the duration for each iteration increases linearly, although the lower middle graph shows that QL computes

each iteration significantly faster than VI or PI. This computation advantage of Q-Learning is much more pronounced than in the Drilling MDP, which suggests that the number of total reachable states has a larger impact on the computing time for VI and PI.

Figure 14: Q-Learning Results Overview

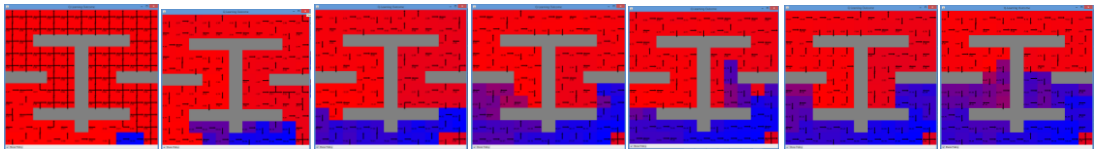


The Rewards in QL are clearly different from VI and PI. While QL is capable of generating policies that perform just as well as VI and PI (see top right graph, iterations 6, 8 and 10), the exploration component of the algorithm lowers the average performance of QL. The converged total utility of VI and PI were nearly 87, the converged total utility for QL was 73, including the drops below 50 net reward. QL meets the convergence criterion at I=18, at least 3x longer than VI or PI.

As opposed to VI and PI, QL does not simply change the policy incrementally to improve performance; it tends to be more "creative" and integrates very different approaches to explore the set of policies. This has an advantage of overcoming local optima, which can be particularly beneficial when the global optima is not intuitive, as in complex problems such as when AlphaGo beat Lee Sedol in 2015.

The policies for iterations between 1 and 500 are shown below. At I=1, it can be seen which states QL has begun to evaluate. QL explores radially from the initial state, and already in the first iteration, the Q-values suggest an optimal path along the bottom of the grid. At I=10, QL has begun to exploit this local optima and by I=20 QL has a strong understanding of the shortcut. At I>100, it can be observed that QL continues to explore beyond the local optima in search for another local optima. For demonstration, the policy for I=10,000 shows that QL has not been able to see value in exploring the entirety of the other route. This suggests the net rewards of the two routes are not competitive, however modifications to the domain may make this more interesting.

Figure 15: Q-Learning Policy Development



I =1 I=10 I=20 I=100 I=200 I=500

Further discussion on learning rate, exploration strategy and epsilon value and initial Q-values will follow in the ***Observations & Comparative Analysis*** section.

OBSERVATIONS & COMPARATIVE ANALYSIS

Overview of Results

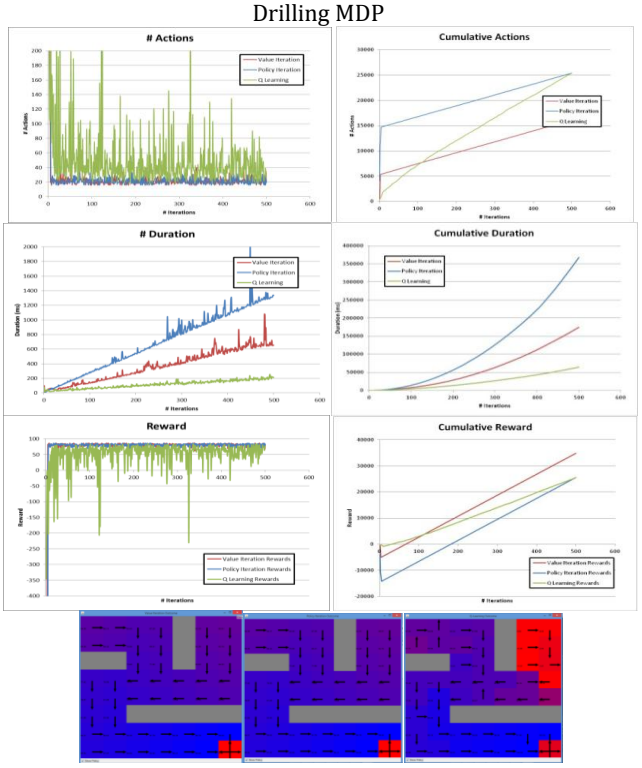
In brief, the results of Part 1, 2 & 3 are summarized in the charts above for both MDPs. The graphs demonstrate the exploratory nature of QL, seen by the many & frequent spikes in the reward and the number of actions. While exploring is useful in discovering new policies, the simplicity of both MDPs favor exploitation, which VI and PI are very effective at. However, VI and PI are seen to require more computing time in Drilling MDP and even more so in Ikea MDP.

In both MDPs, VI and PI converge in much fewer iterations than QL (by 3-5 times), and at a higher value of net reward (by 10-20%). Therefore, for these examples, the main advantage for QL is the computing time, but not the performance. Value Iteration appears to converge in slightly fewer iterations than Policy Iteration, with a duration of half as well. Overall, Value Iteration performs the best out of the 3 algorithms, for both MDPs. The similar results between Drilling MDP and Ikea MDP suggest that the models are quite similar, however variations of the MDPs may demonstrate more distinct characteristics between VI and PI.

The policies shown below for the policies at I=500 for all reinforcement learning algorithms. For both MDPs, VI and PI show the same policy, whereas QL shows similar sets of actions along the optimal path, but very different actions in areas with low Q-value, further away from the optimal path.

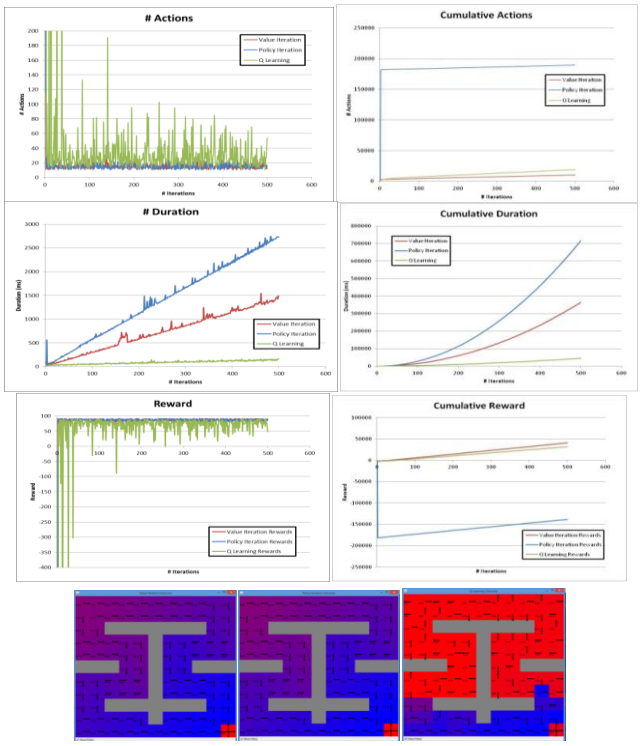
Figure 16: MDP Summary Results

Drilling MDP



VI PI QL

Ikea MDP



VI PI QL

Discount Factor

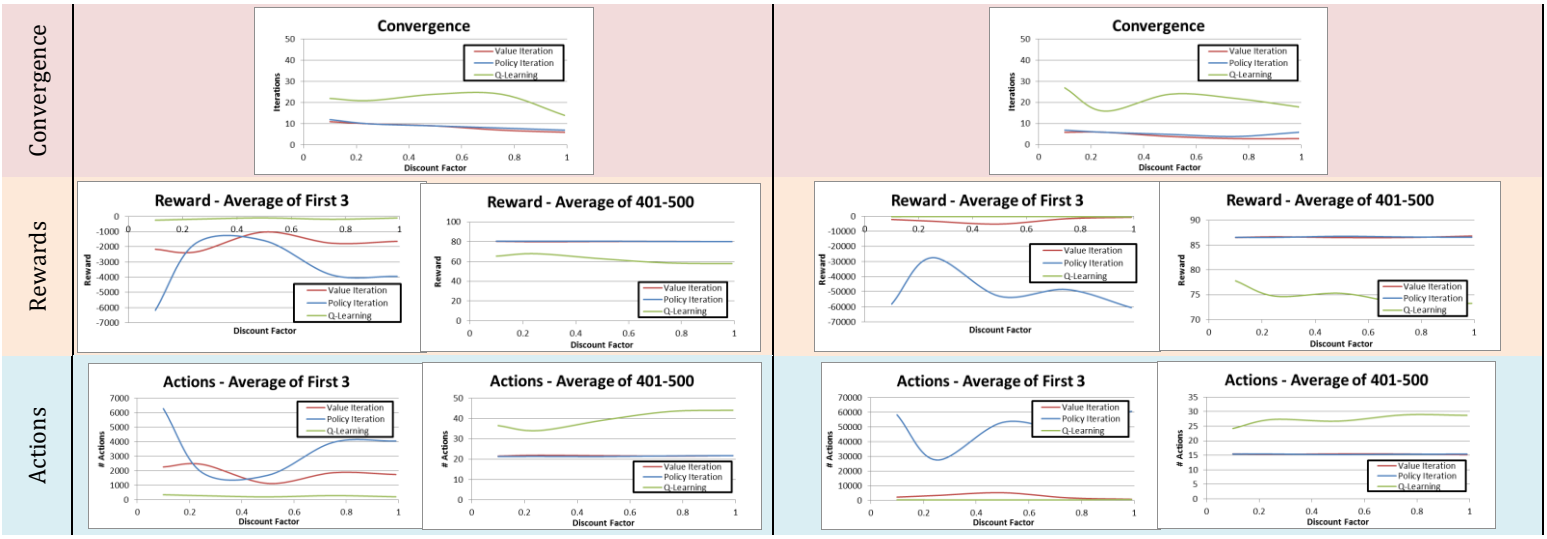
The results above were conducted using a discount factor of 0.99, which makes the agent value future rewards almost the same as present rewards. The following section presents the findings of how the discount factor affects net reward, convergence, the number of actions and the time. The graphs below evaluate the algorithms for various discount factors, but also present the initial behavior (average of the first 3 iterations) and the convergence behavior (average of the iterations between 401 and 500).

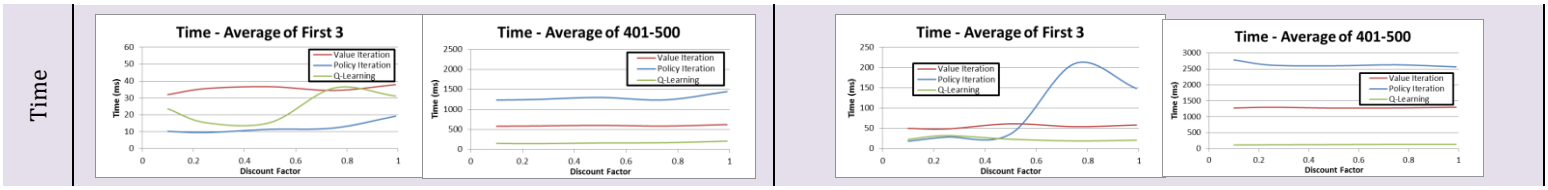
Overall, this study shows that convergence occurs slightly faster for high discount factors, which aligns with the intuition that the algorithms can explore different policies a little more if the final reward is not worth as much. Comparing the initial behavior for both MDPs, the variation between runs appears to show significantly despite the average of 3 runs. However, it is still observed that in the first 3 iterations, QL performs better than VI and PI overall across all discount factors. This characteristic may compensate for the fact that at convergence, QL does not reach the same net reward as the other algorithms. Also, it is observed that PI varies the most for all tests. It is suspected that this variation is largely due to differences between runs in general. This may suggest that the initial policy assigned by PI is much more random than VI or QL. This may explain why VI converges slightly faster than PI.

Figure 17: Discount Factor Results

Drilling MDP

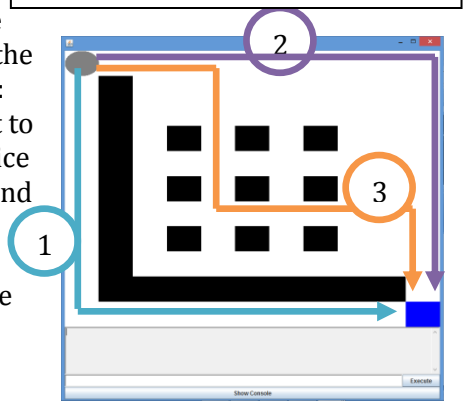
Ikea MDP





Ikea – Modified Domain

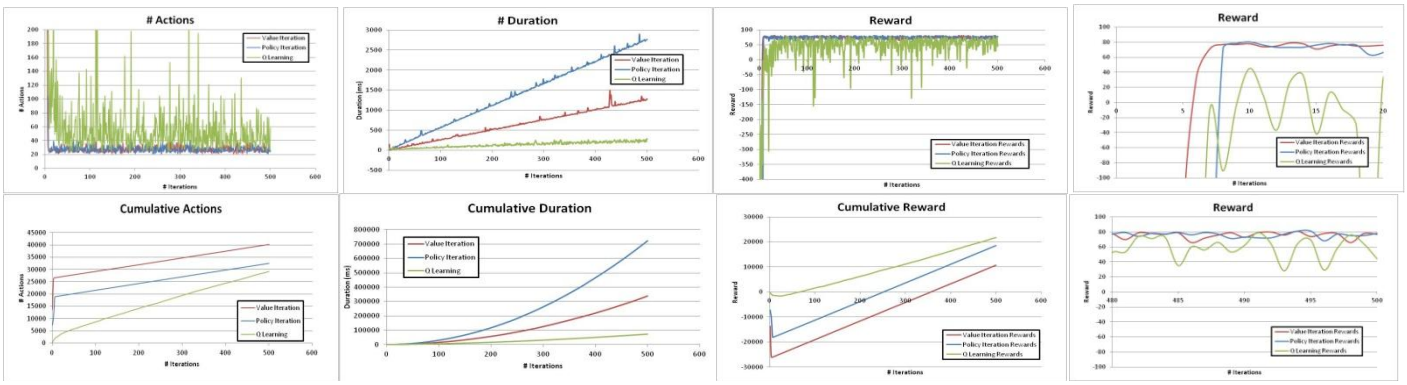
Figure 18: Modified Ikea Domain



The two designed MDPs on the surface appeared different, but the results were very similar. A modified domain was designed of the Ikea MDP for further insight into the differences in the algorithms. Improvements from the original Ikea MDP are as follows: position the agent and terminal location as far apart as possible to encourage the agent to navigate more of the grid; create opportunities for a larger set of possible policies (notice 1: the "direct" approach along the left & bottom, 2: the "open" approach along the top and right, 3: the "obstacle" approach in the middle).

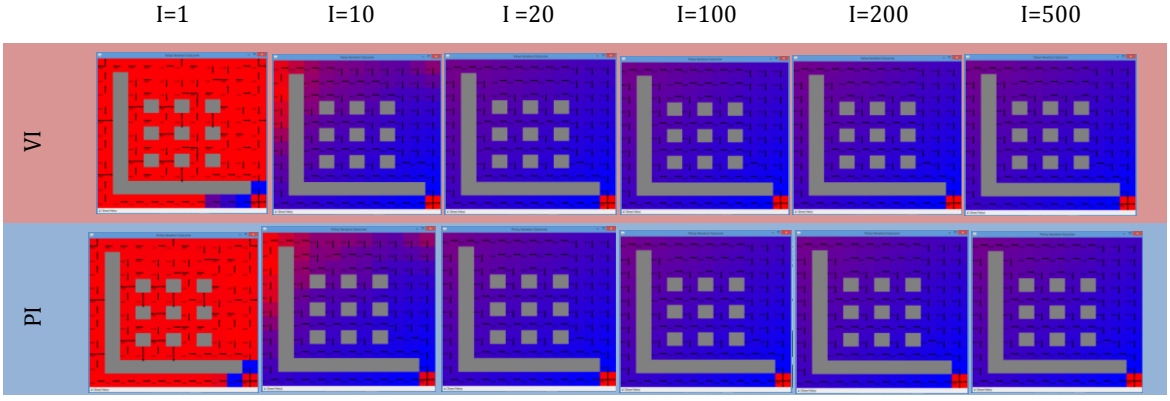
Similarities with the original Ikea MDP include: the number of (95), the linear increase in time per iteration, and the relative time performance between VI/PI/QL, the exploratory fashion of QL, the relative speed of convergence and the relative reward. Notice that VI converges before PI, and QL continues to explore well past VI/PI convergence. At I=500, VI/PI perform consistently with a standard policy, whereas QL can perform between half and equally as well.

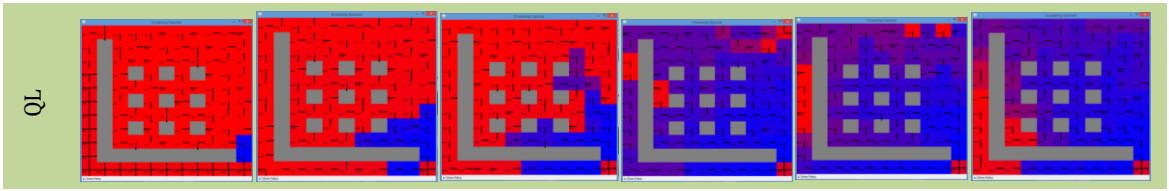
Figure 19: Summary of Modified Ikea Domain



This new MDP makes the policies more interesting. While it was expected that the Path 1 would perform best because of its simplicity, VI and PI have determined that the first action should to the right (Path 2/3). In hindsight, this makes sense because of the probabilistic actions, in which when the agent pursues the Path 1, there is a 13.3% chance it will hit a wall and not progress, and 6.7% it will take a step back. Taking the Path 2/3, the agent generally makes progress when it advances in 2 of the 4 directions (down or right), increasing chance of progress 86.7% versus 80% in Path 1. It is observed that Path 1 is not suitable for QL due to its limitation to explore, and the large space of possible actions in Path 2/3 result in varying policies between I = 100, 200 and 500. While QL does not perform as well per iteration, the cumulative results tell a different story. QL performs extremely well in the first 5 iterations relative to VI and PI as seen by the number of actions (300 vs. >5000). This has a large impact on the cumulative reward, which is higher for QL even at I=500. It appears as though the cumulative reward for VI/PI will surpass QL in finite time.

Figure 20: Policy Development of Modified Ikea Domain



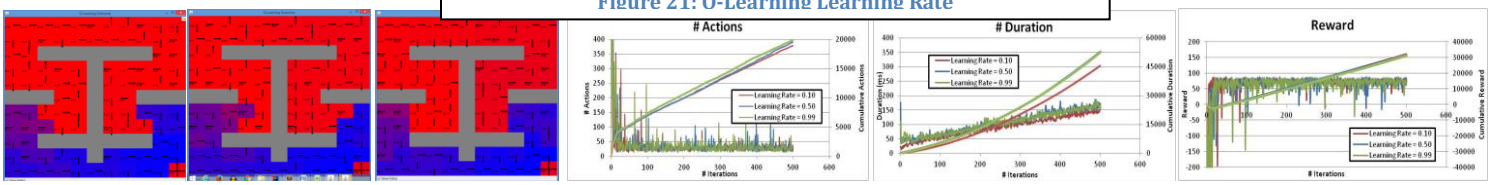


Q-Learning Parameters: Learning Rate & Exploration (Epsilon) and Q_Initial

For the majority of this project, the default values for QL were selected to observe the general behavior of the 3 algorithms. However, for further study, the default values were changed to assess how they affect the results.

The learning rate, L, controls the level at which new information overrides existing information. When L=0, the agent does not learn from new information whereas when L=1, the agent only considers new information. A constant learning rate was used for this MDP using ConstantLR.java, where the value of L=0.99 was initially used for experiments, however L=0.50 and L=0.10 were explored on the Ikea MDP. The results demonstrate that the behavior is similar, shown by the policy and the per-iteration and cumulative graphs below. While the performance is similar, it appears L=0.10 performs best. This suggests that new information is less valuable and supports the notion that the clear path along the bottom favors exploitation over exploration. Further study may explore the effect of using ExponentialDecayLR.

Figure 21: Q-Learning Learning Rate

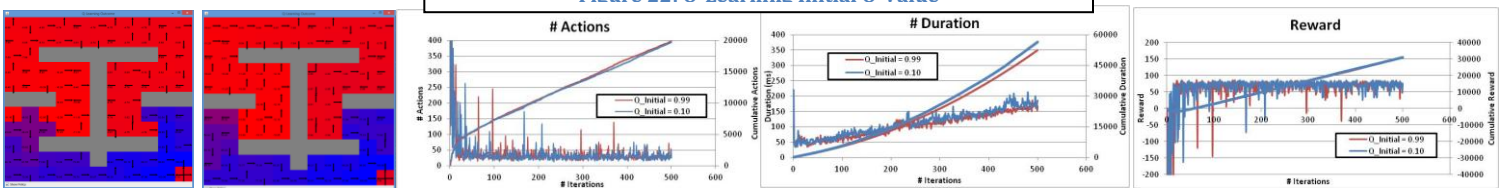


L=0.10 L=0.50 L=0.99

It was observed that QL continued to explore after it had converged on the optimal policy (as compared with VI and PI). This is controlled by the GreedyEpsilon strategy and default value of ε=0.1. For further study, it will be interesting to evaluate RandomPolicy or Boltzmann Policy, using values of epsilon between 0.1 and 0.99. It is expected that Boltzmann will perform best, as the high temperature dictates a high level of exploration, which cools down over time to prioritize exploitation.

To initialize the algorithm, QL applies an initial Q-value to all states prior to iteratively updating the values. There are generally two approaches to Q_Initial: use high initial values (0.99) to create optimistic conditions and encourage exploration; or to use low initial values (0.10) to create conservative/realistic conditions to encourage exploitation.

Figure 22: Q-Learning Initial Q-Value



Q_Initial=0.10 Q_Initial=0.99

CONCLUSION

To conclude, both MDPs demonstrate that Value Iteration performs better than Policy Iteration because they converge to the same policy but Value Iteration converges faster and computes faster per iteration. Q-Learning computes the fastest of the 3 algorithms, but with the epsilon greedy strategy the ability to explore was not advantageous for the QL converged reward. However, in the first several iterations, Q-Learning performs significantly better than Value Iteration and Policy Iteration by several magnitudes (-200 QL Reward vs. -20,000 PI Reward and -30,000 VI Reward).

The analysis on discount factor demonstrates a quantitative measure of exploration, however use of other MDPs may better highlight the impact of discount factor. As an extension to this paper, it is recommended to explore different grid designs and reward/cost structures. In addition, MDPs may be applied outside of the 2D GridWorld such as MountainCar (tutorial in BURLAP that models momentum and gravity to climb a mountain) or Real-Time Bidding in online advertising as used in Facebook ads or Google Adwords (an interesting study combining the Knapsack problem in Random Optimization with MDPs [5]).

BIBLIOGRAPHY

[1] CS7641 Assignment 4 BURLAP Extension https://github.com/stormont/cs7641-assignment-4
[2] Brown-UMBC Reinforcement Learning and Planning (BURLAP) java code library http://burlap.cs.brown.edu/
[3] Kaelbling, Littman, Moore. Reinforcement Learning: A Survey. https://arxiv.org/pdf/cs/9605103.pdf
[4] Ibrahim, M.F. (2002) The Importance of Entertainment in the Shopping Centre experience. Journal of Real Estate Portfolio Management, Vol. 8, No. 3, pp 239-254
[5] Duijndam (2011). Adwords Bid Optimization https://www.few.vu.nl/en/Images/stageverslag-duijndam_tcm244-234413.pdf