CS7641 ML- Project 4 report

Chenxi Yu

Saturday, Apr 22, 2017

1. Introduction

In this final project, I have chosen the grid world task as the main Markov Decision Processes (MDP) topic to apply value iteration (VI), policy iteration (PI) and Q-learning to develop policies for our agents to reach the goal state. In order to develop make comparisons for the two different processes with distinct state sizes and different difficulties in the goal seeking task, I manage to keep the processes as similar and interpretable as possible. This is why I decided to develop two different difficulties of the same game, where the agent is supposed to traverse a grid-like world from the same starting location to the same goal location in the map's edge. There is only one optimal path to the goal state from the starting point, whereas the difficulties lay in the different branches of routes available, as the (15*15) grid world does have longer and more misleading routes than the (7*7) grid world for the agent to traverse before reaching the goal. Though both tasks are obvious to human, they are not so to a computer. However, these problems may get optimal results by applying breadth first search approach in terms of computational time and space complexity. Yet, for the sake of studying MDP and reinforcement based on simplicity and interpretability, I will stick with exploring this classic problem.

Why do I choose grid world? Grid world seems to be a trivial problem with no real world application, yet it is not the case when considering its relevancies to the construction in automated robotics. In the real world setting, an automated robot or machine serves no use if not adapting to its environment and surroundings. Adaptation requires the robot to avoid obstacles to proceed its task without being explicitly programmed. A self-driving vehicle generates and processes signals via observations and makes decisions per second to achieve its goal of delivering its passengers to the destination efficiently and safely. This is very similar to the grid world's case, there is a starting location and a destination that are placed on a map. Within the map, there are obstacles where an agent in the grid need to avoid bumping into, and where meaning fatal to a self-driving vehicle if bumped into. With the help of pixel conversion, in Google map, the non-driving area, "the obstacles" include buildings, parks, pedestrian area, opposite paths and so on. With the application of a reward function that assigns severe penalties to entering states like those, an automated vehicle is made possible.

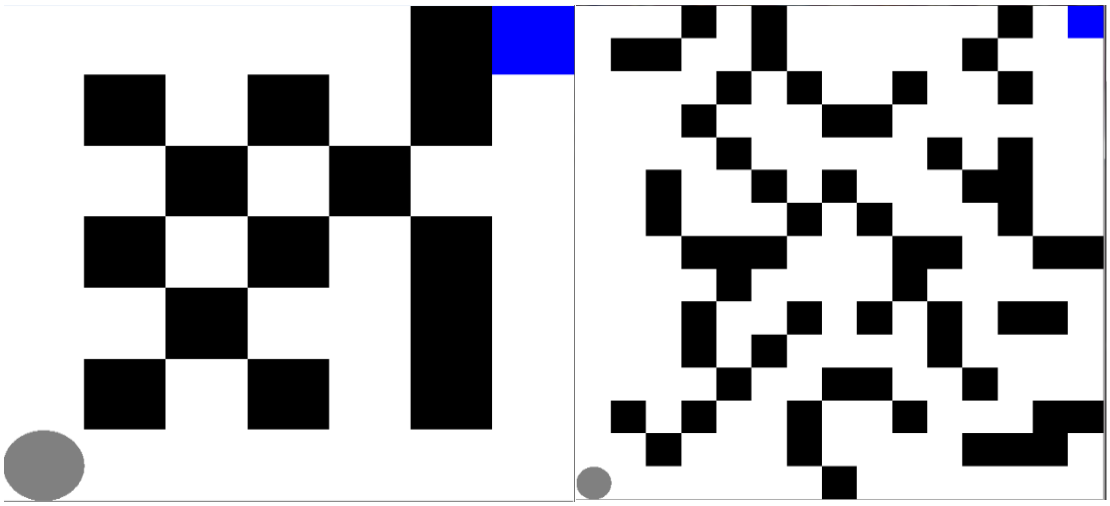
In the Grid world, the agent is designed to decide 1 of 4 possible actions to proceed in every step. In terms of a vehicle, the agent would have to decide many other factors, such as the rate of acceleration, the speed at which to remain, and the proper degree angle to turn the wheels of the vehicle in order to reach the next desired state. Like Grid world, the agent could be presumed with measurable probabilities of success in switching states due to the possibility of deterioration of the vehicle over time. This means that agent is compensated for these deteriorations

like that in Grid world is so for its possibility of success.

The bottom line here is that a demonstration of understanding this classic application will benefit my further exploration of RL and AI in the later research and work. Also, hopefully after being introduced to BURLAP applied in Grid world, I will further study other optimization algorithms to refine my understanding in RL.

Intuitively, after studying and watching lectures from our class and Berkley's CS188, given that VI and PI both require the proper transition probabilities and reward functions to obtain optimal policies, it was fairly reasonable to have results that these two algorithms typically converged to the optimal policy in fewer iterations than Q Learning, which had to derive that information from experience. Additionally, VI and PI, given their almost identical paradigms, typically converged to the same optimal policy in similar number of iterations. While VI and PI were clearly extremely similar, Q Learning deviates a lot in its algorithmic paradigm and results. More specifically, while VI and PI are given a lot of information to assist the searching upfront, Q Learning must derive this information by experimenting and interacting with the process. However, in general, the problem with VI and PI is that they must be assign with proper transition probabilities and reward functions, being that by grid search approach, domain knowledge or simply look ahead bias. I guess this is why they are just simple algorithm foundation for more practical framework like Q learning. The struggle with Q learning itself is that given both transition probabilities and reward functions are unknown, more iterations and running time are required to converge to an optimal policy like the ones calculated by VI and PI. This performance shortcoming gets obvious if the searching states available in the space increase merely from a (7*7) grid world to a (15*15) grid world. In short, while it is clear that with a higher number of states, there is an exponential increase in number of iterations required for Q Learning to converge on an optimal policy, it is important to note that this algorithm is converging without a transition probability matrix or reward function given up front. If a set of reasonable transition probability matrix and reward function is available up front, it is better to use either Value Iteration or Policy Iteration to calculate an optimal policy. Otherwise, approaches like Q learning may do better if the searchable space is big, versatile and dynamically changing. Thus, in the real world, I believe Q learning is more adapted to a real world application if proper algorithms is added to improve its speed performance or if enough domain knowledge constraints is given to limit its hypothetic searching space.

2. MDP problem Description:



The Grid world landscape consists of an agent (gray ellipse), walls (black squares), and a goal (blue rectangle), or terminal state. The objective of this MDP is for the agent to traverse the space and reach the terminal state in the least number of steps (actions). Above is the generated visual representations of the two variants. On the left is the "easy" version where there are only 32 possible states and only 2 "long" misleading routes (never leads to the goal), while on the right represents the "hard" scenario where there are 167 possible states and 7 possible "long" misleading routes with possible infinite states re-explored (meaning exploring a small space as there is a black in its center) as shown in the map.

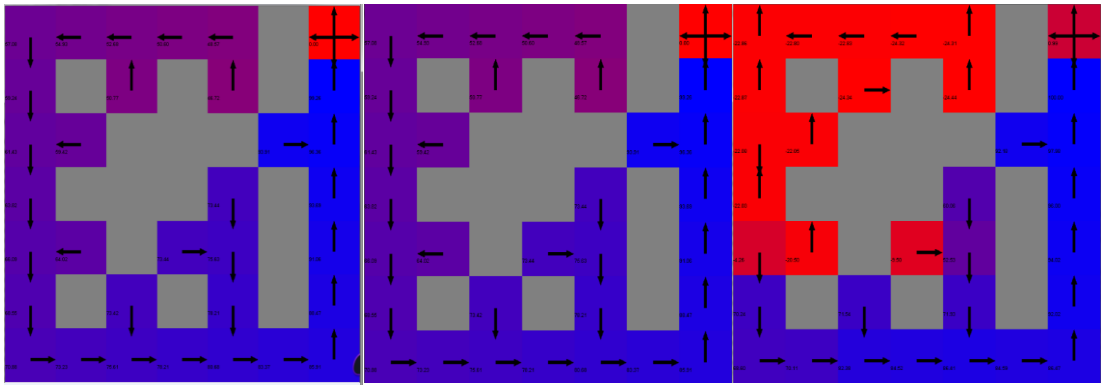
Reward function, or the cost of each additional action determines the agent's incentive to reaching the terminal state in the least number of steps. Like the setting in the literature, the agent loses 1 point for every action that does not immediately land it on the terminal state whereas if the agent's action immediately lands it on the terminal state, the agent is awarded 100 points. In a true MDP, there requires a chance that the actions performed bearing inherent probabilities of success and failure. This is made possible by having a transition matrix that dictates the probability of success that the agent will be able to move successfully in the direction that it intended. In every action, there are a maximum of four possible actions by movements north, east, south, and west. A successful movement in the direction intended has a probability of 80% while a movement in any other direction has a probability of 6.67%.

Now I will describe the 3 approaches VI, PI and Q learning in details.

3. Low Difficulty Grid World:

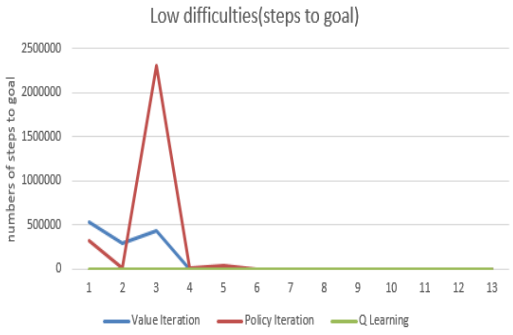
This section will focus on discussing the similarities and differences between VI, PI and Q learning and below shows their respective optimal policies. It is not surprising to find that both VI and PI yields the same result. They both initiates an arbitrary policy, meanwhile the former updates a policy based on neighbors and latter updates a policy based on getting policy improvement after each step's evaluation. Q learning's result looks optimal if the agent moves east as an initial action. However, if the agent initially moves north, it may be trapped in the middle

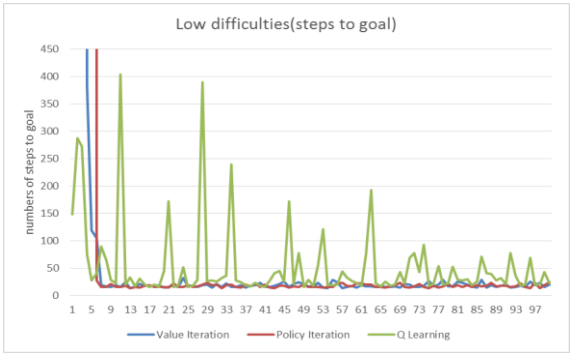
of the first column. This means that if the agent unluckily moves north, it is likely to get trapped in that "misleading" zone. Also, in the "misleading" zone, it is easier for the agent to bump into the boundaries of the world or the wall (obstacle) in the Q learning's while not the same case for the other two. Thus, it is self-evident that the optimal policy after 100 iteration in Q learning is less optimal than the VI and PI. It could be the fact that Q learning has yet converge after 100 iterations.



(From left to right are value iteration, policy iteration, Q learning on optimal policies after 100 iterations)

A. Steps to goal in iteration





Iteration	Value Ite	Policy It	Q Learnin
1	528309	318337	148
2	295531	2591	288
3	426890	2303862	272
4	382	12976	75
5	119	39475	27
6	105	28	40
7	19	15	89
8	17	16	65
9	15	21	29
10	19	15	22
11	15	16	404
12	24	18	20
13	13	13	33
14	15	16	17
15	21	14	31

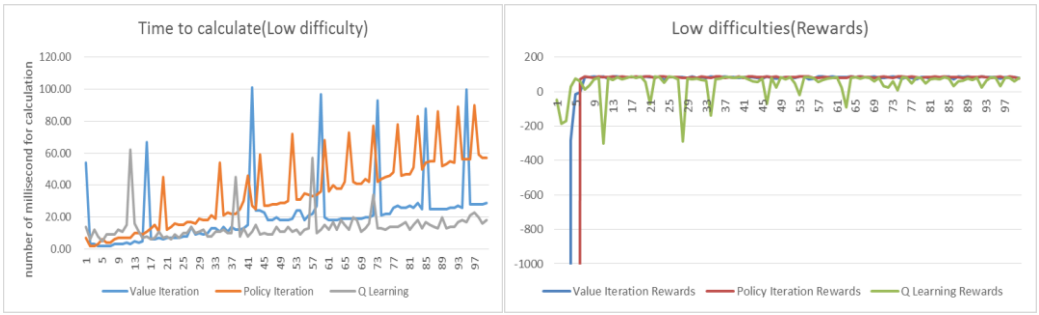
93	16	17	36
94	18	22	18
95	16	16	20
96	25	13	69
97	19	23	20
98	23	13	20
99	16	19	43
100	20	24	22

(the upper left shows the first 13 iterations while the second graph emphasized the iteration after 13 iteration, the bottom two data chart shows steps to reach the goal)

It is clear to find that both VI and PI converge around 7 to 9 whereas this is not as a strictly convergence in the Q learning after 100 iterations as shown in the data chart and graphs above. Overall, PI and VI converges at 13th iteration by 13 steps to goal. It seems that Q learning has not converged by 100th iteration and its convergence speed is slow as expected. But this setting has favored PI as there are not many states available to

search and thus policy evaluation has not hindered performance as obviously in slightly more complex scenarios like the one we have later in this report.

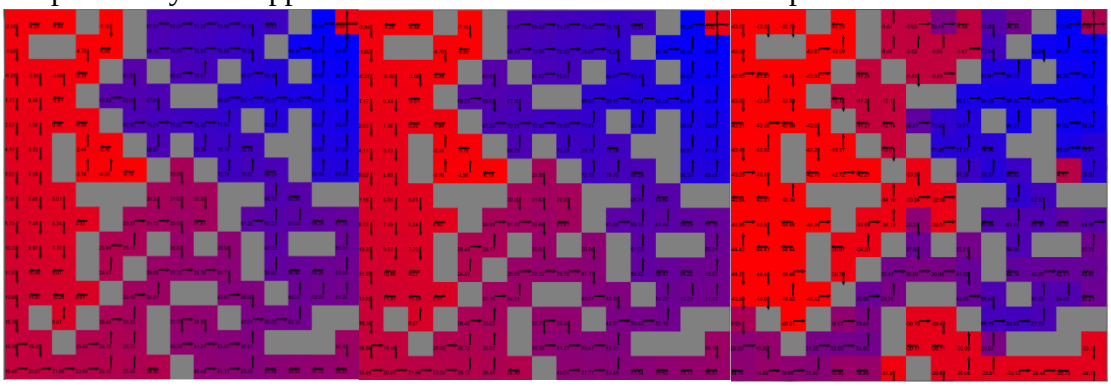
B. Time complexity and reward chart:



The time complexity chart shown in the left above indicates that the PI may take the longest occasionally than the others, given that the PI's maximum iteration number has not been set with a cap. The slow performance of PI is most likely because it takes a policy generation step that occurs during each iteration while VI does not. By avoiding the policy generation step, Value Iteration does run a little bit faster than Policy Iteration. Thus, this also complies with the searching mechanism of VI compared with PI since VI's space search cover some repetitive steps that PI avoids. Between PI and VI, VI sacrifices computational space due to its greedy approach while PI computes relatively slower due to the policy evaluation step. The rewards graph also indicates similar trend of VI and PI in line with the smallest convergence iteration stated previously, so is the Q learning's graph shape. Nevertheless, there exhibits a linear pattern for all three in time required for calculation. In term of the spike in the time graphs, I believe that has to do with the trap path set in the grid world. Interestingly, Q Learning appears to run at a constant rate of 12-18 milliseconds regardless of the number of iterations. The reason why the algorithm runs is probably due to the fact that the algorithm, during the iterations, is not actually performing any computation like the PI and VI, but hashing the actions taken and the rewards achieved until a policy comes out in the end.

4. High Difficulty Grid World:

As stated earlier, I have modified the map to include 167 possible states and 7 possible "long" misleading routes with possible infinite states re-exploration possibility. It is apparent that the difficulties has leveled up.

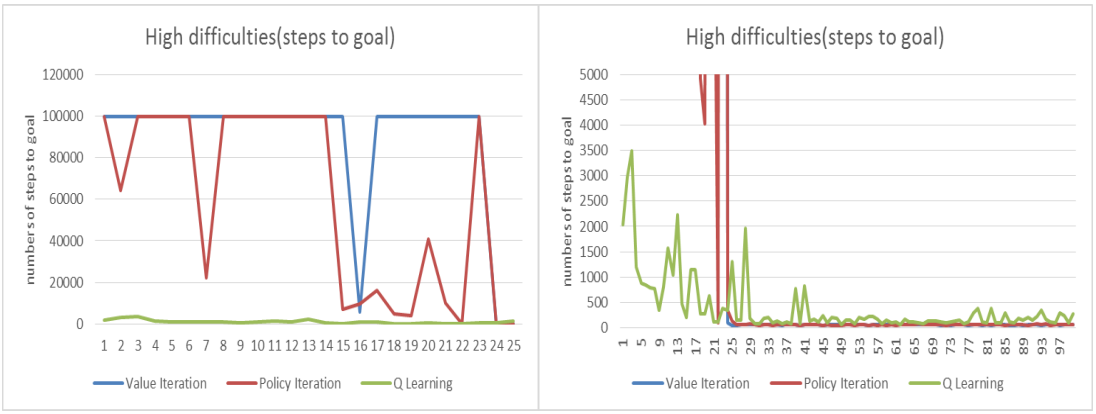


(From left to right are value iteration, policy iteration, Q learning on optimal policies after 100

iterations)

The result looks similar to the previous easy grid world in the optimal policy maps, except that this time witnesses a worse result on Q learning since there has not been an optimal policy which leads the agent to the goal state, while the PI and VI have both achieved optimality and there has not been a bumping-in wall scenario in either approach. Another key observation from the comparison is that the negative expected reward distributed in PI and VI have no dramatic low value while Q learning has. Also, interestingly, Q learning agent has marked some key tipping states well as if it could identify traps region in the map.

A. Steps to goal in iteration:



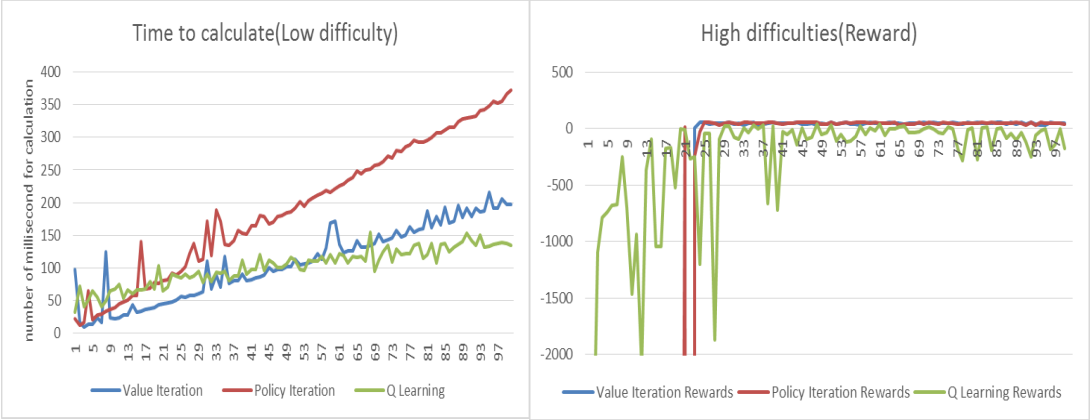
Iteration	Value Ite	Policy It	Q Learning
9	100000	100000	349
16	5898	9837	1150
17	100000	16064	1147
18	100000	4916	275
19	100000	4032	270
20	100000	40862	624
21	100000	10118	104
22	100000	91	110
23	100000	100000	371
24	92	345	349
25	43	127	1302
26	48	49	146
27	61	50	145
28	58	51	1973
29	51	68	191
30	52	53	82
31	50	47	76
32	61	56	182
33	56	63	194
34	69	48	98
35	56	49	134
36	46	58	72

88	59	56	190
89	48	57	146
90	63	47	202
91	49	55	138
92	62	68	221
93	48	49	350
94	63	69	161
95	72	45	117
96	73	53	102
97	45	57	282
98	51	52	244
99	53	53	100
100	58	62	274

(the upper left shows the first 25 iterations while the second graph emphasized the iteration after 25 iteration, the bottom two data chart shows steps to reach the goal)

The above steps to goal shows similar result to the low difficulties grid. Both VI and PI are converging faster than Q learning, yet this time VI has converged faster than PI. More specifically, VI converges at 25th iteration by 43 steps to goal while PI does so at 41th iteration. However, there is one noticeable detail in the above data chart. I have limited the max steps to 100000, thus we could have had the presented results. Otherwise, there could have been a run time error every time it proceeded through PI. I assumed this is due to the fact that the agent may have over-explored the space and thus suffering from exploring a small area with a center block forever. According to the bottom right corner data chart, we find that Q learning has yet converged and thus explaining why the optimal policy after 100 iteration has not been successful to reach the goal.

B. Time complexity and reward chart:



Like in the low difficulties Grid world, the run time required to run the experiments, regardless of the algorithm, is linear. PI is the slowest comparatively throughout the whole process, with VI the second. However, it is also noticeable that Q learning have a larger run time overhead at the early iterations, while the time to generate a policy is constant regardless of the number of iterations. The curves in the reward graph complies with the previous analysis. Most importantly, it also indicates that the Q learning has yet converged after 100 iterations.

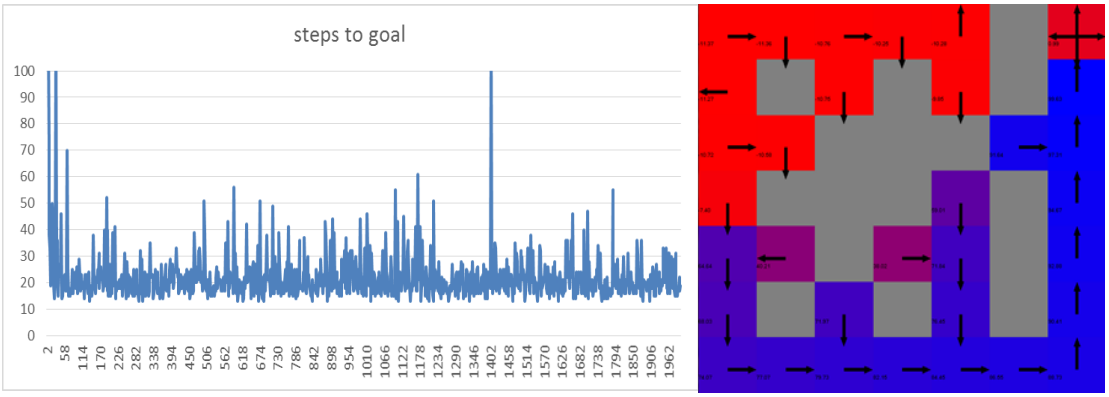
Overall, as for how the number of states increase from 32 to 167 affecting each algorithm's performance, it is obvious that this time PI converges slower. PI may keep exploring without the interference of the cap on the maximum steps in a larger states space. Q learning is in line with the intuition that it converges slower than any other algorithms. Yet, the spike appearing in the graph of the 167 states are not as observable as the 32 states. But it seems that with the setting of a maximum steps, PI performs better than VI in speed as the "time to calculate" graph shows. The later report also indicates that the time taken for Q learning convergence increases exponentially as states increases in the grid world problem.

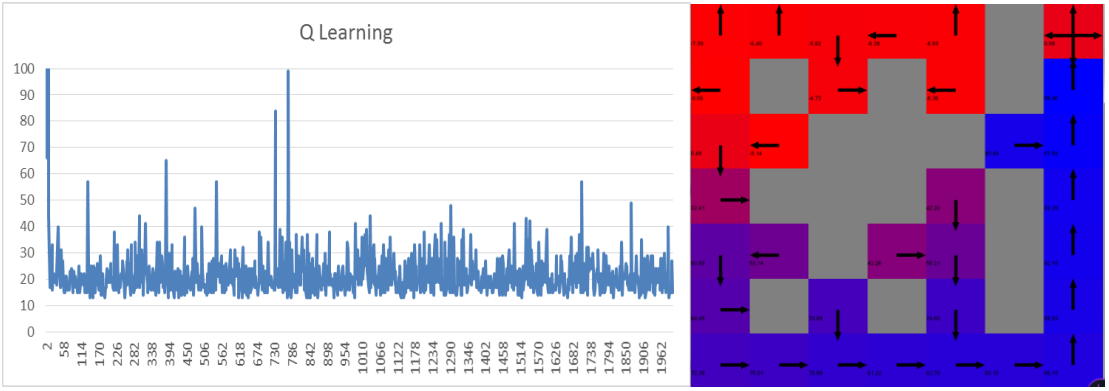
5. Grid Worlds: Q Learning Expanded

In both grid worlds, Q learning have yet reached an ideal optimal solution, in this section, I aim to further investigate these 2 cases.

A. Low difficulties grid world: Q learning

Let me continue with the easy grid world experiment. Upon the 136th iteration, indeed, Q learning does reach the optimal policy. However, it is not obvious to observe through the graph shown in the upper left below. Now, I applied the greedy Q policy pre-written in BURLAP, I found though the agent reach the goal on the 140th iteration, the spike shown in the graph indicates more stability over that under the general policy.

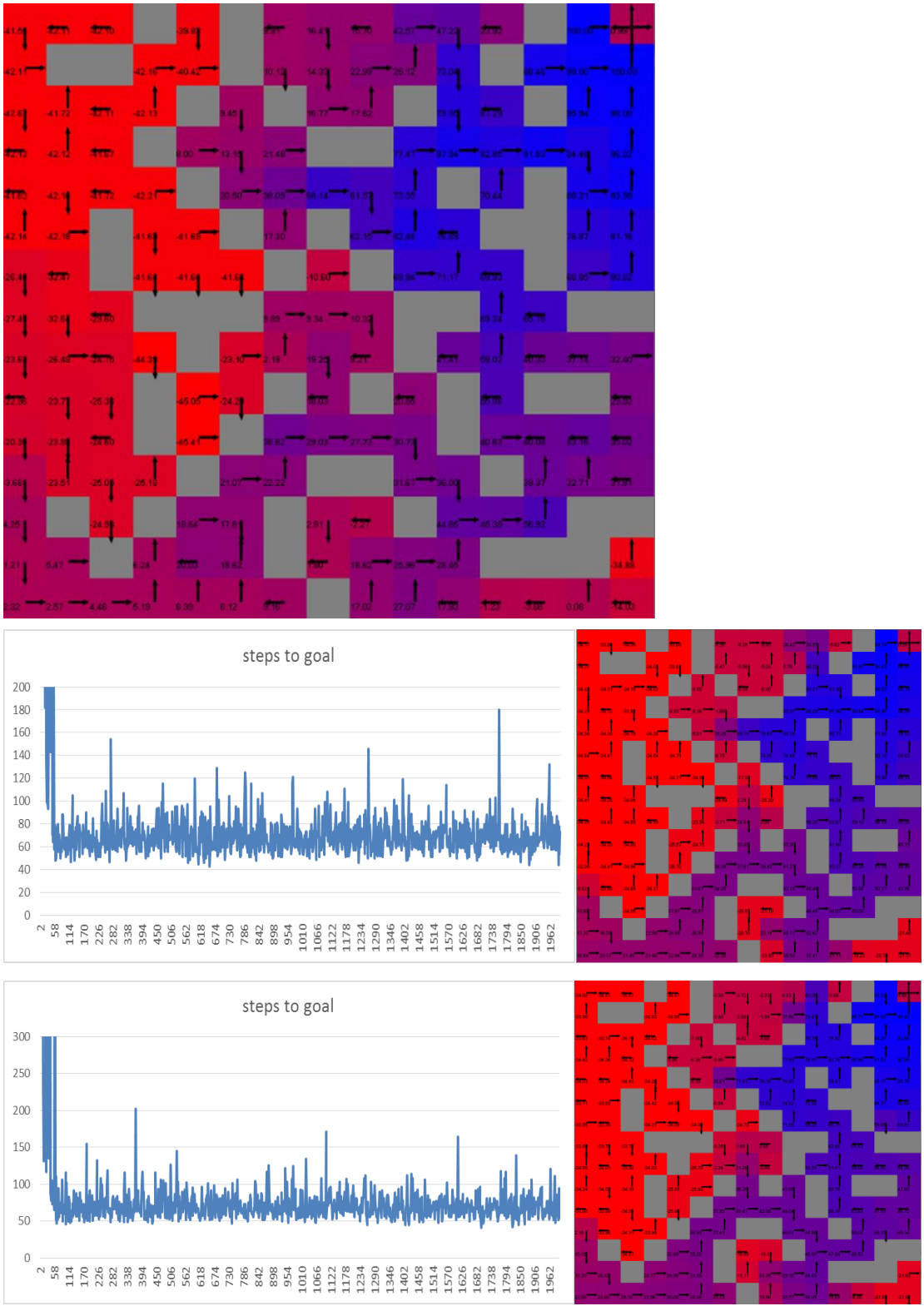




(On the upper graph pair is the general policy's result, on the lower graph pair is the greedy Q policy's result)

B High difficulties grid world: Q learning

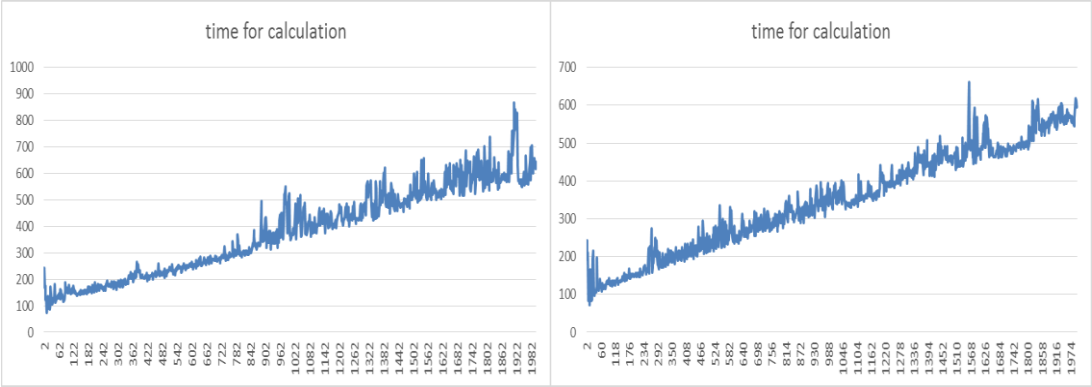
The high difficult grid world is further studied under a learning rate of 0.5 instead of the previously .99. Because an optimal policy is not reachable even after 10000th iteration shown as below. On the 672th iteration, Q learning achieves the optimal policy which takes 43 steps to reach the goal. Next, the greedy Q policy is applied again and surprisingly the optimal policy obtained takes only 41 steps after 1700th iteration. Yet, I assume that is due to the odd that 2 fewer unintended steps were taken since the optimal count by hand has a result of 39 steps. Thus, a learning rate of .5 is a better strategy than that plus the greedy Q policy since steps equal to 41 is still not the ideal solution.



(On the upper graph is the optimal policy after 10000 iterations at a rate of .99, on the middle graph pair is the policy at a rate of .50, on the lower graph pair is the greedy Q policy's result at a learning rate of .50)

Though essentially the greedy approach does not help the process, the adjustment of the learning rate to .5 works to achieve a sub-optimal policy at steps required by 43. However, this result is still far worse than the one achieved by VI and PI who converges

within 100 steps. Anyway, I learned that adjusting the learning rate could be of great importance to speed up the convergence of a complex grid world problem.
In terms of time complexity, it seems that the pure 0.5 learning rate does not push the agent to explore a lot until later iterations, whereas that under the greedy Q policy does and therefore saving time from exploring in the later iterations.



This is not a stochastic environment as the grid landscape has yet changed. But treating it as a somewhat in the middle of a deterministic and a stochastic environment does help the agent converge to the optimal policy. This is probably because the high difficult grid world has embedded many traps that mislead the agent. Since the solution appears to be quite zip-zagged, the greedy approach does not help. Thus, this also tells the importance of balancing exploration and exploitation in Q learning. One that always explore may does worse than one that only explore where the scenario is filled with sufficient uncertainties

I believe the Q learning agent could benefit a lot in performance fundamentally if the agent is able to scan the surrounding space by a reasonable size “look ahead” window. This could significantly avoid small traps in a complex system.

6. Conclusion:

Given the fact that Value Iteration and Policy Iteration both require the transition probabilities and reward functions in order to calculate optimal policies, it is not uncommon to find these two algorithms typically converge to the optimal policy in fewer iterations than Q Learning, which had to derive that information from experience and experiments. Because those inputs are different between the former 2 and the latter, the problems they are trying to solve are inherently different. As we find that Q learning could indeed converge without knowing or preconditioning the transition probabilities matrix and reward functions, though an exponential increase in iteration is expected as the anticipated states increases. Therefore, if the reasonable transition probabilities and reward functions are provided, value iteration and policy iteration prevail. Otherwise, it is better to conduct Q learning and optimize its performance after sufficient consideration in the trade-off between exploration and exploitation.