Project2 COSC 6368 [Fall 2016]

Learning Paths from Feedback Using Q-Learning

University of Houston, Oct.2016 [Group H]

Priyal Kulkarni (priyalkulkarni1@gmail.com) ID: 1520207

Sarthak Sharma (ssarthak15@gmail.com) ID: 1520229

Xiaoyang LI (betty93635@gmail.com) ID: 1456424

1 Pick-up and Drop-off (PD) World

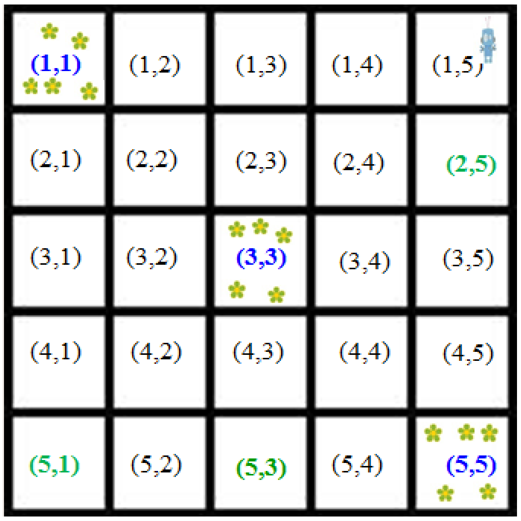


Figure 1 PD-World matrix

Agent start from initial state (1,5), and pickup cells contain 5 blocks (denoted in ✿) and terminates when drop off cells contain 5 blocks each, in which pick-up cells (blue): (1,1), (3,3) , (5,5) and drop-off cells (green) are (5,1), (5,3), (2,5).

In this delivery transport problem, as Fig.1 shown, a courier (agent) carries one block each time, traveling through the PD-world where it would only move north/south/west/east ward. During the travel, it will pick up the blocks from pick-up cells and drop them off at drop-off cells. Our goal is to design a route from the agent so that it could use the least steps to send all the blocks to drop-off cells.

Initial state of the PD-World: Each pickup cell contains 5 blocks and drop-off cells contain 0 blocks; the agent always starts in position (1, 5).

Terminal state of the PD-World: All the 15 blocks has been delivered, i.e. pickup cell contains 0 blocks and drop-off cells contain 5 blocks.

2 Reinforcement Learning

2.1 Basic Reinforcement Learning Concepts

Reinforcement learning is the problem faced by an agent that must learn behavior through trial and error interactions with a dynamic environment [1], i.e. learn optimal policy with a priori unknown environment. To solve reinforcement learning problems, we use statistical approach and dynamic programming, especially Q-learning, to estimate the utility of taking actions in the states of the world. To clarify the problem clearly, we simplify the PD-world to Fig.2, in which P stands for pick-up cells and D stands for drop-off cells.

P				
				D
		P		
D		D		P

Figure 2 Simplified PD-World matrix

In order to describe how we use reinforcement learning to solve PD-world problems, some basic definitions are proposed as follows:

Experience Tuple $< \boldsymbol{s}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{s'} >$ a tuple to summarize a single transition in the environment.

- ***State s:*** Agents state before the transition
- ***Action a :*** Choice of action
- ***Reward r:*** Instantaneous reward it receives, reflects immediate or short-term consequences of executing actions. In our project,
    - Picking up a block from a pickup state: +13
    - Dropping off a block in a drop-off state: +13
    - Applying north, south, east, west: -1
- ***State s':*** Agents state after the transition

Utility on the contrary to reward, is to present the long-term consequences. Intuitively, the utility of taking action in some state is the expected immediate reward for that action plus the sum of the long-term rewards over the rest of the agent's lifetime, assuming it acts using the best policy [2].

Q- learning: the following update formula is used every time an agent reaches states' from s using actions a:

$$Q(s,a) \coloneqq Q(s,a) + \alpha(r + \gamma \max_{a'} Q(s',a') - Q(s,a))$$

Utility value $\boldsymbol{Q(s,a)}$***:*** Measure the utility of using action a in state s. We assume that Q values are initialized with 0 at the beginning of the experiment.

Learning rate α: Determines to what extent the newly acquired information will override the old information. It controls the speed of updating utilities. In our project, we set it for $a = 0.3$ and $a = 0.5$ for different experiments.

Discount factor (rate) γ: Determines the importance of future rewards. A factor of 0 will only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward. In our project, we set it to be γ= 0.3 to have a decent discount rate.

2.2 Design of Reinforcement Learning

Blocks: The 6×6 matrix showing the current number of blocks in the PD world. If there are any blocks at those coordinate, then blocks[$\mathbf{i}, \mathbf{j}$] =number of blocks it has otherwise 0. After pickup, the block number in the position will decrease by 1, drop-off will increase by 1.

Operators: North(N), South(S), West(W), East(E) are applicable in each state, and move the agent to the cell in that direction except leaving the grid is not allowed.

SSpace= ($\mathbf{i}, \mathbf{j}, \mathbf{x}$***):*** The agents state space, in which(i, j), is the horizontal and vertical coordinates of agent ($\mathrm{i}, \mathrm{j} \in \{1 \ldots 6\}$); x is the Boolean vector indicating whether an agent carries a block $(\mathrm{x} = 1)$ or not $(\mathrm{x} = 0)$.

Qtab: the Q table store the learned result of Q values. A full Q table would contains sSpace and Q learned values of each operators, overall 25 x 2 rows and 6 columns. However, the Q values for the drop-off and pickup operators do not necessarily have to be reported, so we will make a separate analysis for Q values of states with x=1 and of states with x=0.

isDropOff : The Boolean vector that only returns true if the agent is in a drop-off cell that contains less than 5 blocks and if the agent carries a block.

isPickUp: The Boolean vector that only returns true if the agent is in a pickup cell that contain at least one block and if the agent does not already carry a block.

Policies specify what action to take for every possible state s. In our experiment, the following 3 policies will be used:

> ***PRandom:*** If pickup and drop-off is applicable, choose this operator; otherwise, choose an operator randomly.

PExploit1: If pickup and drop-off is applicable, choose this operator; otherwise, apply the applicable operator with the highest q-value (break ties by rolling a dice for operators with the same q-value) with probability 0.65 and choose an applicable operator randomly with probability 0.35, as shown in left side of Fig 3.

PExploit2: If pickup and drop-off is applicable, choose this operator; otherwise, apply the applicable operator with the highest q-value (break ties by rolling a dice for operators with the same q-value) with probability 0.9 and choose an applicable operator randomly with probability 0.1, as shown in right side of Fig 3.

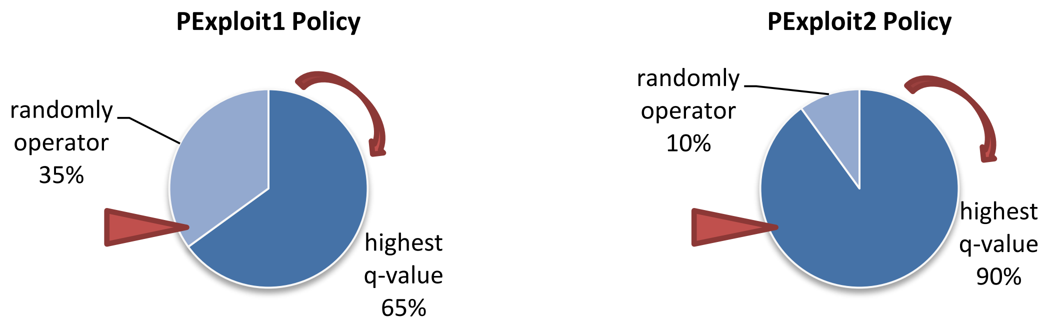


Figure 3 Policy of PExploit1 and PExploit2

3 Experiments Implementation and Analysis

To implement the Q-learning on PD world, we experiment six ways to interpret the performance of Q -learning algorithm, the parameters setting is shown as Table 3.

Table 1 Parameter Setting of 6 experiments

Experiment	Learning rate	Policy	Learning steps	Actions to terminal states
1	0.3	PRandom	10000	reset PD world to initial state
2	0.3	PRandom PExploit1	1~100 101~10000*	reset PD world to initial state
3	0.3	PRandom PExploit2	1~100 101~10000*	reset PD world to initial state
4	0.5	PRandom PExploit2	1~100 101~10000*	reset PD world to initial state
5	0.5	PRandom PExploit2	1~100 101~10000	1st : reset PD world to initial state 2nd: swap pickup and drop-off locations.
6	0.5	PRandom PExploit1	1~100 101~10000	1st : reset PD world to initial state 2nd: change the pickup locations to: (2, 2), (4, 4), (1, 5).

* If a terminal state is reached the fourth time it terminates the experiment prematurely; in this case Q-learning runs less than 10,000 steps.

3.1 Visualization and Evaluations

3.1.1 Visualization

For each experiment, we implemented visualization techniques to interpret the result, which inspired by [3]. In terms of Q values, we displace them into arrow form for each action, as Figure 4 shows.

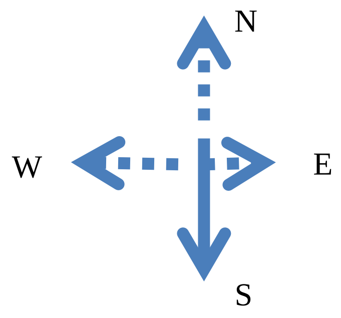


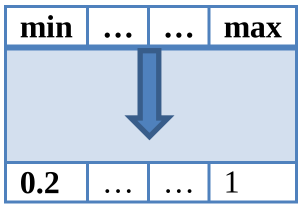
Figure 4 Visualization of actions

Basically, the actions have larger Q values are supposed to have longer length of arrow. Before that, some preprocessing procedure as down as follows:

1) Assignment the illegal actions to -1000

   Illegal actions are the defined as the actions go beyond the P-D world, e.g. North actions for rows of cells that are 1, i.e. $(i, j) = (1, n), \forall n = 1 \dots 5$. In arrow visualization, we will not show any illegal actions.

2) Normalize the Q-values using feature scaling



min	**…**	**…**	**max**
0.2	…	…	1

   Considering the restriction that the length of rows could only be positive and within 1, we normalize the Q-values over each cell according to the adapted version of feature scaling formula:

$$Q_{\text{normalized}} = \frac{q - min(q, cell)}{(max(q) - min(q, cell))} * 0.8 + 0.2$$

3) Learning pruning

To showing the legal moves, there are some scenarios that even though some of the q-value has been learned, the agent will not visit this cell, especially in the beginning of the iteration. The actions are inhibited by setting all the length of arrows to zero the following cases for 4 q values in each cell:

- all are zeros;
- negative 1 zero
- 2 negative 2 zero
- 1 negative 3 zero

The arrows that have the largest q-values are regarded as ***attractive paths;*** the South action in Fig.4 shows an example of attractive path in a single cell.

Noted that the entire processing are only implement for displaying, they will not may any change for Q –value itself.

3.1.2 Performance Measuring

To evaluate the performance of each experiment, we observe their performance changes over following three aspects:

Bank account: Considering the fact that agent must pay money (indicating rewards are negative) for every move it attempt to go and only earn money (indicating rewards are positive) when it arrives a pick up/ drop off location, bank account is designed to be a indicator for accumulated rewards at each step of the agent, and it will reset to zero after a termination has reached. In real experiments, the back account would frequently go to negatively large because the agent trials on different actions and have comparably low possibility to reach pick up/ drop off locations, while it also occasionally go up when resets have set or pick up/ drop off locations has reached.

Interval Rewards: The rewards received over number of operators applied over 50 steps of operator applications. It indicates the bank account changing speed to some extent.

Delivered Blocks: The number of block has delivered over number of operators applied over 50 steps of operator applications. This value increases from 0 to 15 and reset to 0 after every resets.

For each experiment, we run them for 2 times with different random seeds each time, and compare their performances over the 3 measures above.

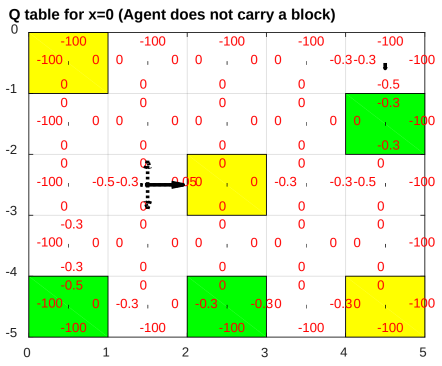
3.2 Experiments Results

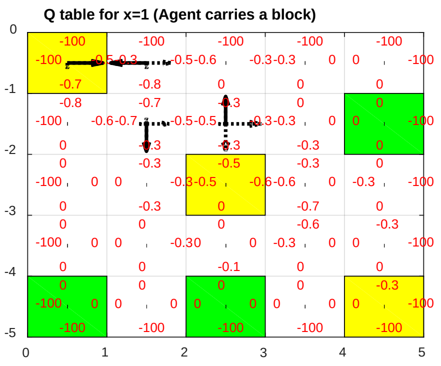
In Experiment 1 we use α =0.3, and run the Q-learning algorithm for 10000 steps with policy PRANDOM. If the agent reaches a terminal state, reset the PD world to the initial state, but do not change the Q-table.

For Q table, depicted on the First row of Fig.5, we display separately for Agent carrying and not carrying a block(x = 0 and x = 1). In each cell, the action is shown as the arrow in Fig 4, all the legal actions are shown in imaginary line arrows, in which only the attractive paths are displayed in full line arrows. The length of arrows are various depending on their q- values. Furthermore, we display the string of value for each action right on the 4 positions of cells. For Full states, depicted on the 2nd row of Fig.5, we simply display the number of blocks for each Pick up /drop off location.

Q-Table and full states of P-D world are visualized in the following 4 crucial sceneries:

a. 100 steps





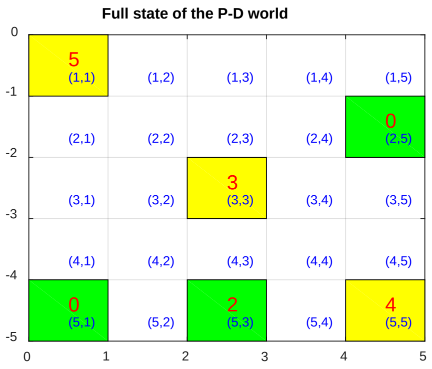
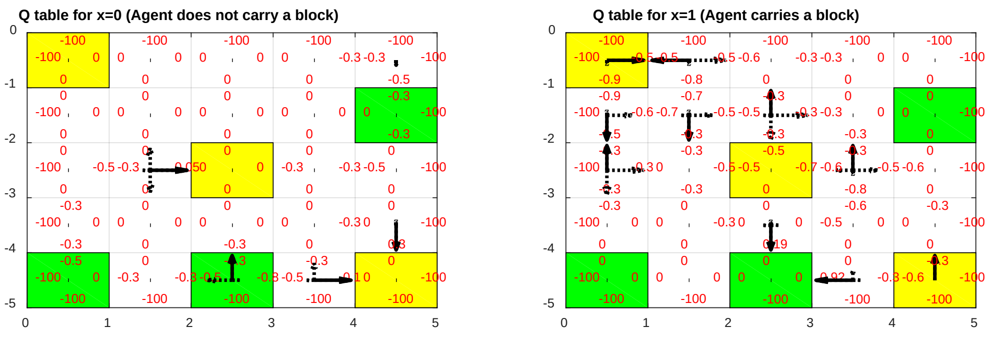


Figure 5 Q tables and Full State visualization on 100th steps for Experiment 1

The pickup and drop off locations are noted in yellow and green squares respectively.

The 100th step is just started for exploration on PD world, so most of the spaces haven't been visited yet. While the tendency of approaching to pick up location for agent not carrying a block and approaching to drop off location for agent carrying a block are already seen from the graph.

b. When the first drop-off location is filled (the fifth block has been delivered to it)



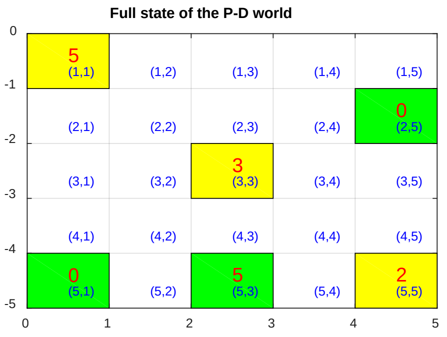
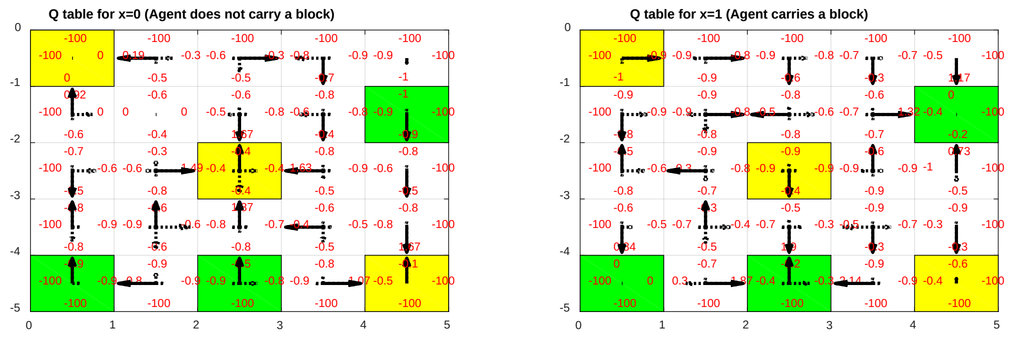


Figure 6 Q tables and Full State visualization when the first drop-off location cell is filled

As Full State shows, the first drop-off location cell (5,3) is filled, correspondingly approaching tendency to cell (5,3) in Q table x=1 and escaping tendency in Q table x=0 are obvious.

c. When the first terminal state is reached.



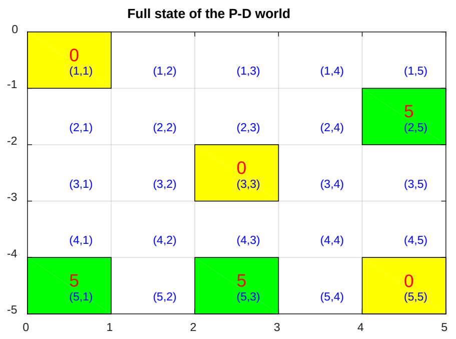
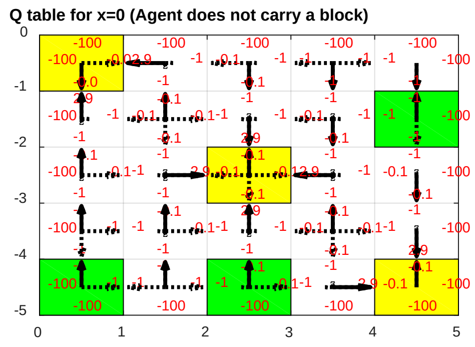
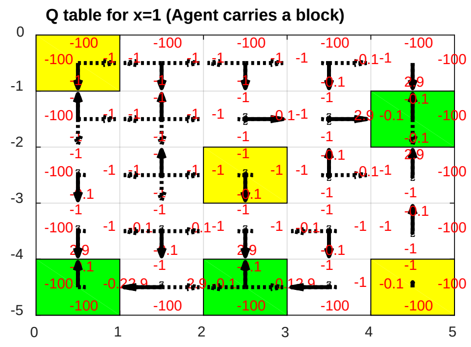


Figure 7 Q tables and Full State visualization when first terminal state is reached

As Full State shows, the all the drop-off location cell are filled and all the pick –up location are empty, indicating a terminate loop of experiment. Arrows pointing to nearest pickup/ drop off locations are shown as expectedly in Q table.

d. When the last terminal state is reached.





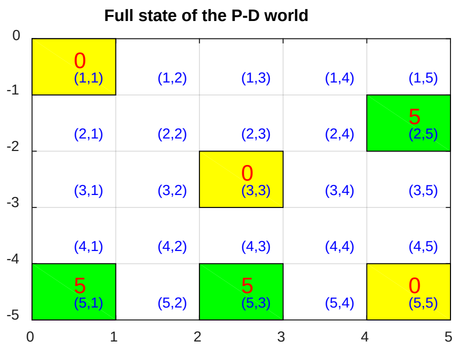
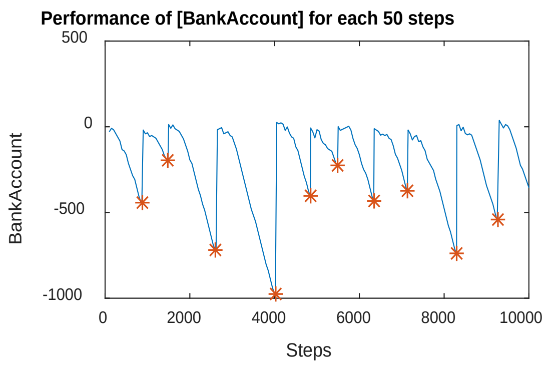


Figure 8 Q tables and Full State visualization when last terminal state is reached

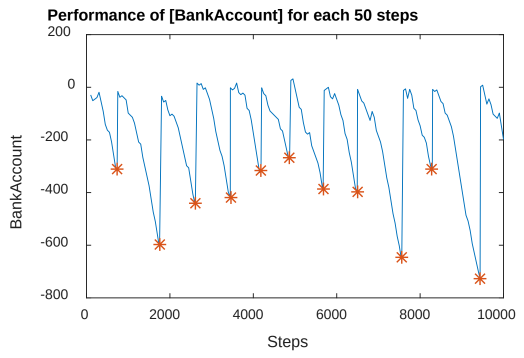
In the end of Q- learning, an optimal attractive paths are supposed to be determined. Compared to the Q- table in the 1st terminal state that some path are still hazard, the final state are optimal in all the cell location.

The Screen Shot of Q table and Full State of experiments 2~6 can be seen in our attachment.

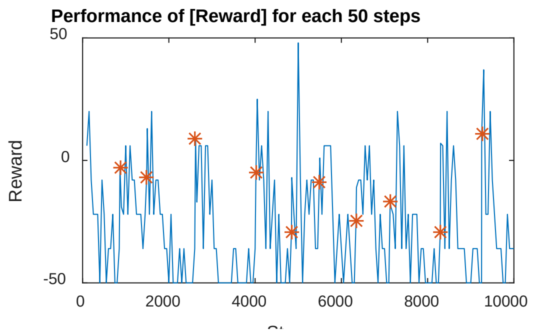
Performance Measures



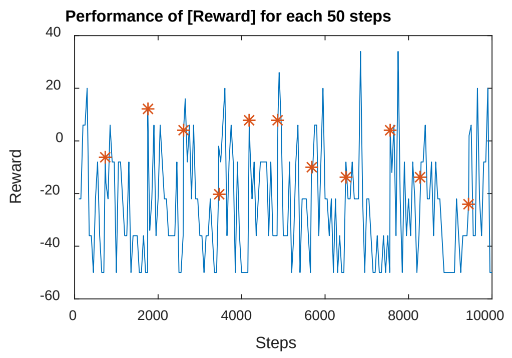
a.1 Bank Account for Run 1



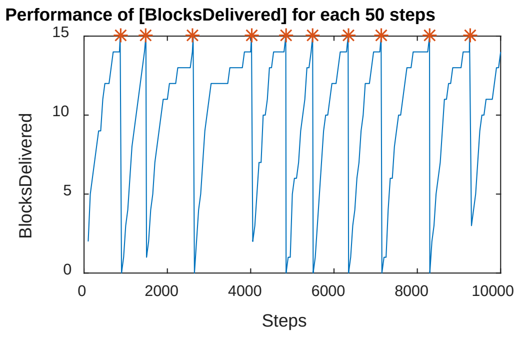
a.2 Bank Account for Run 2



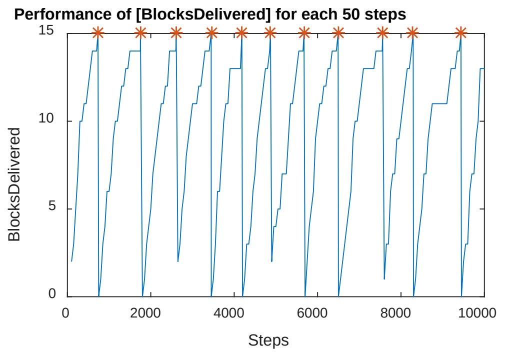
b.1 Interval Rewards for Run 1



b.2 Interval Rewards for Run 2



c.1 Blocks Delivered for Run 1



c.2 Blocks Delivered for Run 2

Figure 6 Performance Results of Experiment 1

Three performance measurements are overserved in every 50 step of Q- learning. Orange stars indicate the steps when PD world is reset to initial state as experiment required.

As Figure 5 shows, bank accounts are set to 0 after every resets. Bank accounts and Interval Rewards graphs of Fig 5.a and Fig 5.a illustrates that the bank consuming speeds are high, i.e. bank account decease fast and in a large scale because the parameter setting for experiment 1 is not wise enough that agent speed much steps to explore the PD world. Number of Blocks

Delivered are supposed to set to 0 at the end of terminates steps and set to initial 15 after every resets. The reason why Fig 5. c.1 and c.2 is unable to reach 0 precisely is that we down-sample the steps for 50 intervals, accepting some critical terminal steps are not capture/ sampled within the intervals.

The Performance Measures of experiments 2~6 can be seen in Chap 6 Appendix Figure A to E.

3.3 Discussions on the results

As we defined in Chap 2.1, learning rate is set to control the sight of learning, determining to what extent the newly acquired information will override the old information. It controls the speed of updating utilities. In our project, we set it for $\alpha = 0.3$ in Experiment 1,2,3 and $\alpha = 0.5$ in Experiment 4,5,6.The policy PEXPLOIT2 combined with the learning rate that has a value of 0.3 or 0.5 turns out to be too harsh for the selected state-space.

All experiment that use PEXPLOIT2 do not always find a terminal state. We run the experiment 5 for 100,000 steps separately to check the effect of swapping the pick-up and drop-off locations. This output is separately put in the folder ‘output 5’.

When we compare experiment 1 and 2 by running them with same seed values, we notice that for 2nd run while the first drop-off location is reached earlier however, the steps taken to reach the termination state is much higher. This is because in the first 100 steps (which are random) the agent learns a promising path to one of the pick-up/drop-off state. But, after some operations, this pick-up/drop-off state is empty/full, and so the agent has to unlearn this change.

For experiments 3, 4 and 5, the graphical analysis appears to be similar because all of them use PEXPLOIT2. The difference of value of alpha between 3 and 4, 5 is only 0.2 which is insignificant to show up on graphical analysis for only 10,000 steps.

4 Conclusion

The project focuses on solving a problem in the Pickup Drop World using reinforcement learning. The given system setup is run for 6 different experiments that use different learning rates and policies. It is observed that experiment 1, 2 and 6 were able to get to the terminating state 4 times within 10,000 steps. Experiment 3, 4 and 5 could only reach the terminating state once which was mainly due to the choice policy.

5 Acknowledgements

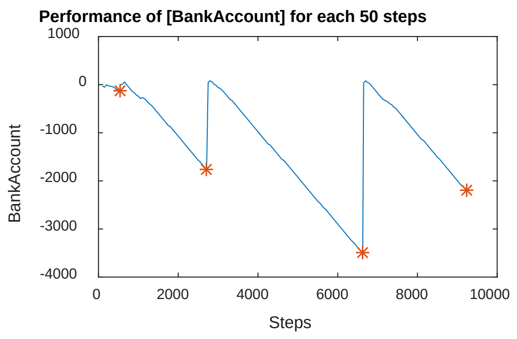
We would like to appreciate the much needed help provided to us by Dr Eick and Nguyen Pham regarding this project. The project couldn't have been completed without Nguyen's help.

We would also like to thank the Professor specifically for adding the three of us to the same group. It was one amazing adventure. Thanks!

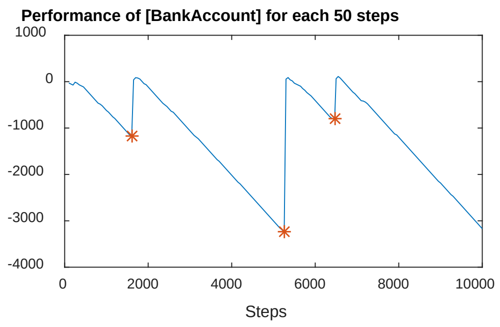
6 References

[1] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." Journal of artificial intelligence research4 (1996): 237-285.
[2] https://web.eecs.umich.edu/~baveja/RLMasses/node3.html
[3] http://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

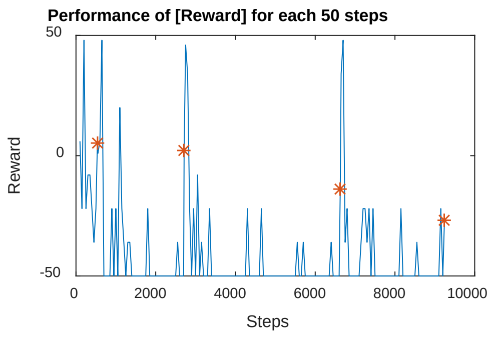
7 Appendix



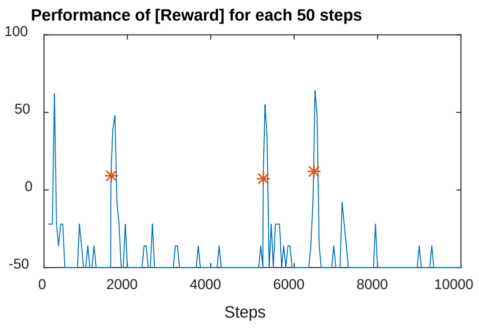
a.1 Bank Account for Run 1



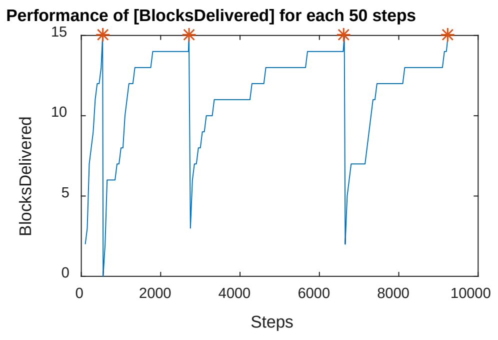
a.2 Bank Account for Run 2



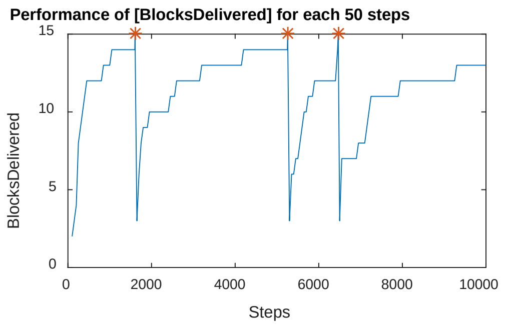
b.1 Interval Rewards for Run 1



b.2 Interval Rewards for Run 2



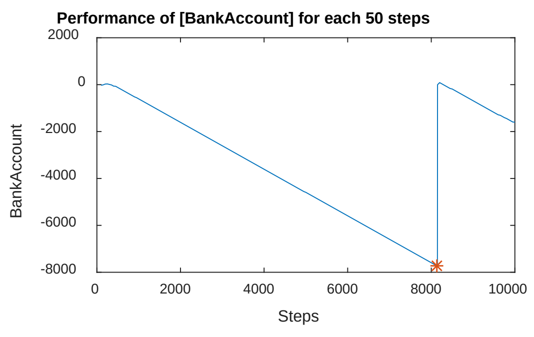
c.1 Blocks Delivered for Run 1



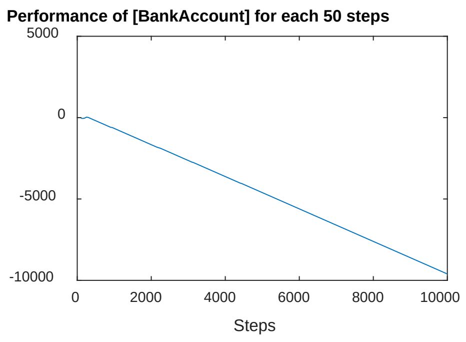
c.2 Blocks Delivered for Run 2

Figure A Performance Results of Experiment 2

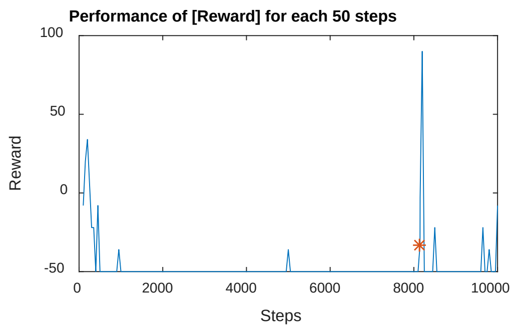
Three performance measurements are overserved in every 50 step of Q- learning. Orange stars indicate the steps when PD world is reset to initial state as experiment required.



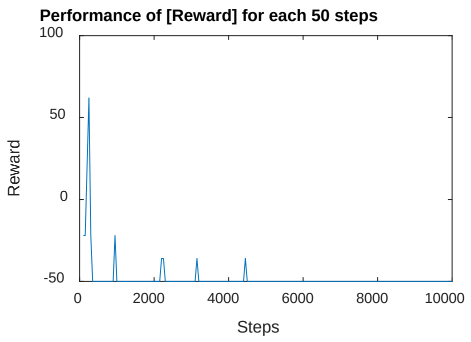
a.1 Bank Account for Run 1



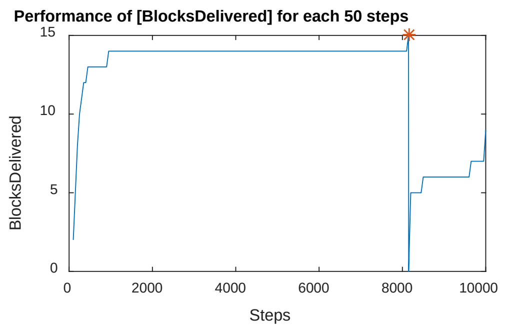
a.2 Bank Account for Run 2



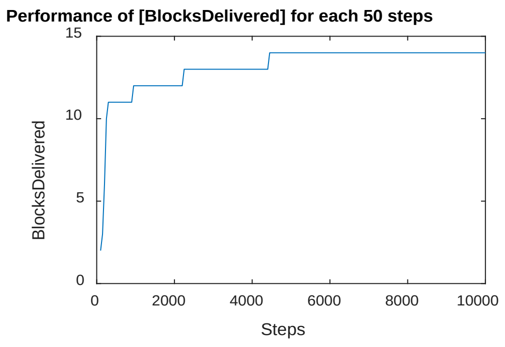
b.1 Interval Rewards for Run 1



b.2 Interval Rewards for Run 2



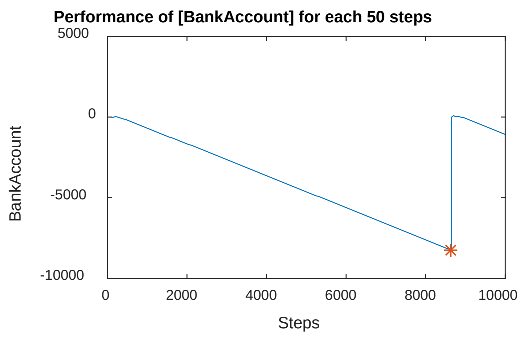
c.1 Blocks Delivered for Run 1



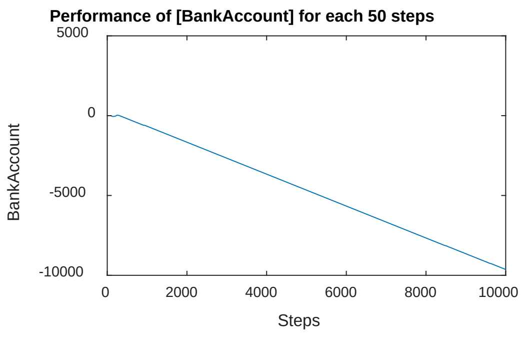
c.2 Blocks Delivered for Run 2

Figure B Performance Results of Experiment 3

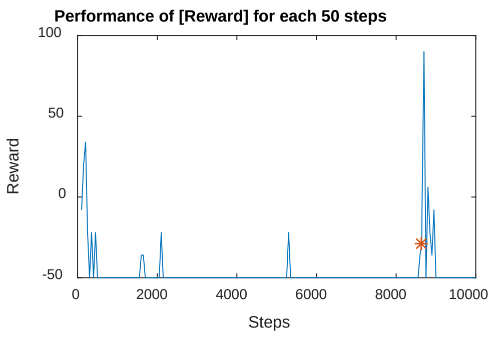
Three performance measurements are overserved in every 50 step of Q- learning. Orange stars indicate the steps when PD world is reset to initial state as experiment required.



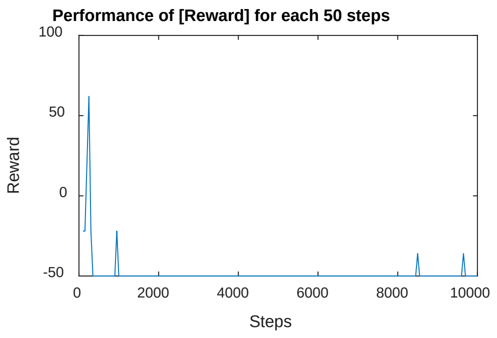
a.1 Bank Account for Run 1



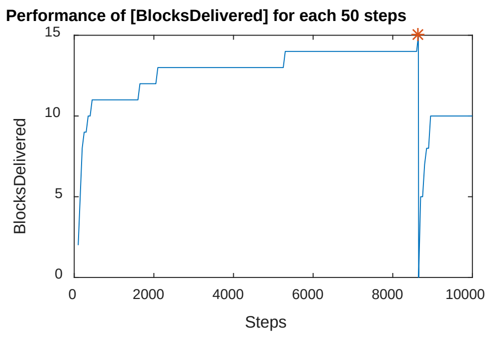
a.2 Bank Account for Run 2



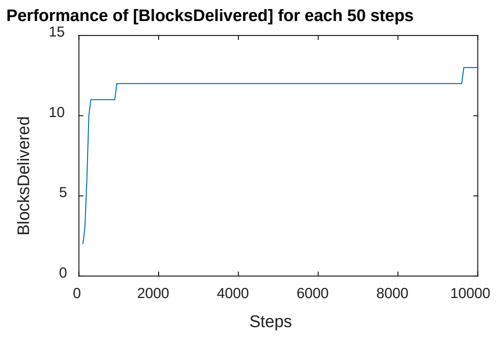
b.1 Interval Rewards for Run 1



b.2 Interval Rewards for Run 2



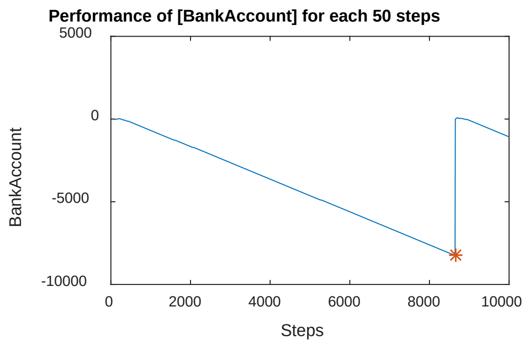
c.1 Blocks Delivered for Run 1



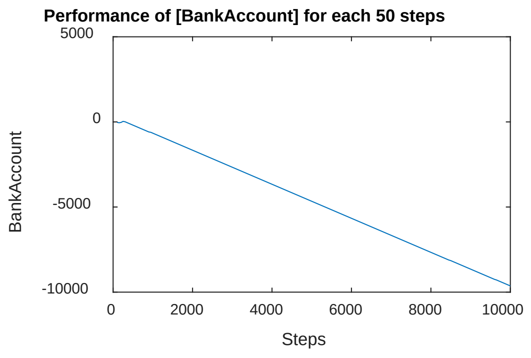
c.2 Blocks Delivered for Run 2

Figure C Performance Results of Experiment 4

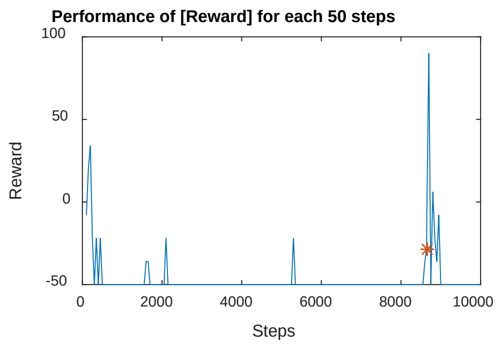
Three performance measurements are overserved in every 50 step of Q- learning. Orange stars indicate the steps when PD world is reset to initial state as experiment required.



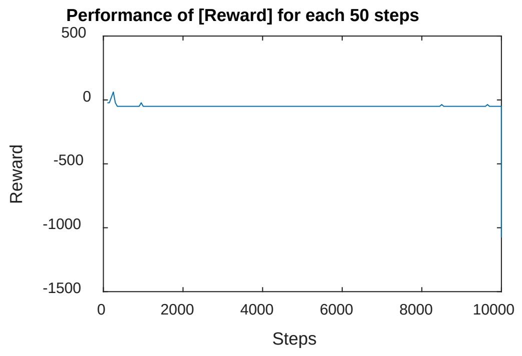
a.1 Bank Account for Run 1



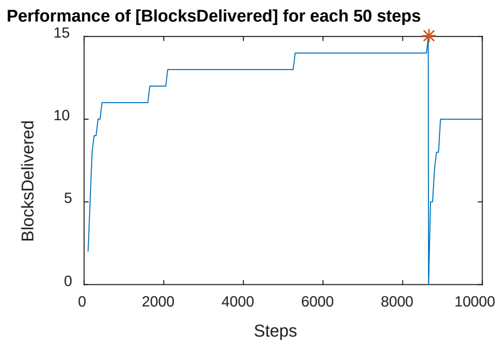
a.2 Bank Account for Run 2



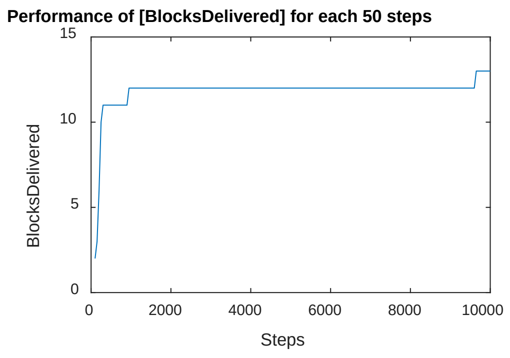
b.1 Interval Rewards for Run 1



b.2 Interval Rewards for Run 2



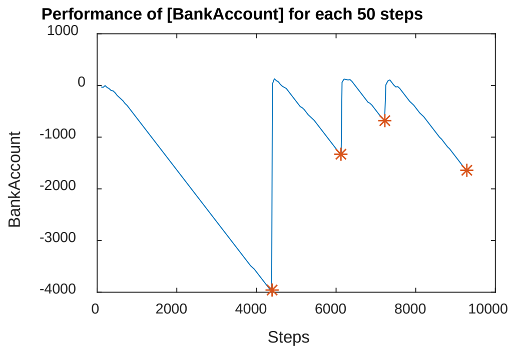
c.1 Blocks Delivered for Run 1



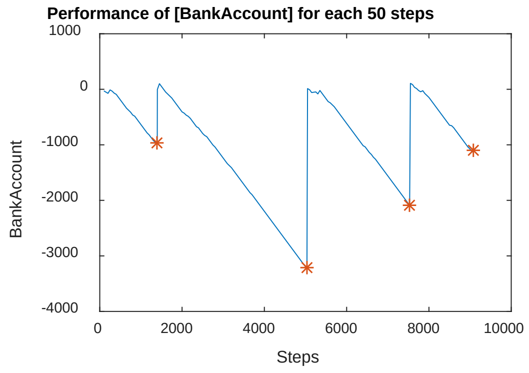
c.2 Blocks Delivered for Run 2

Figure D Performance Results of Experiment 5

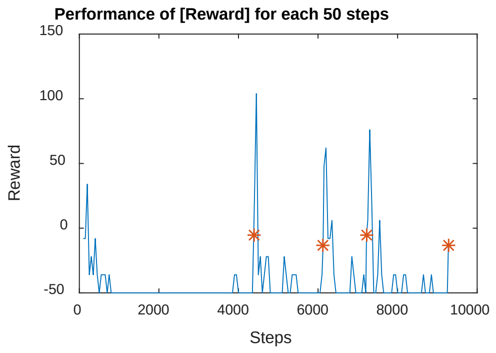
Three performance measurements are overserved in every 50 step of Q- learning. Orange stars indicate the steps when PD world is reset to initial state as experiment required.



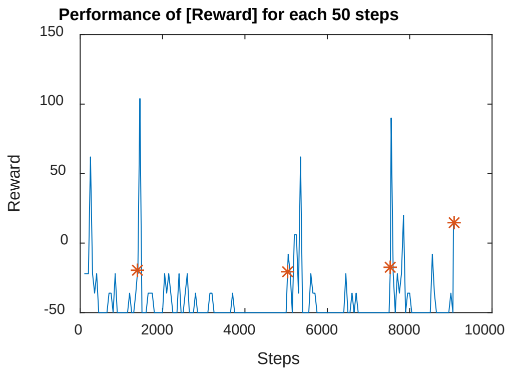
a.1 Bank Account for Run 1



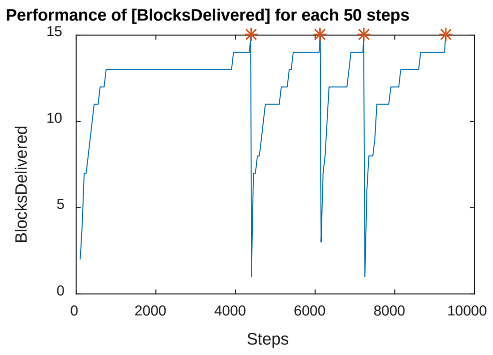
a.2 Bank Account for Run 2



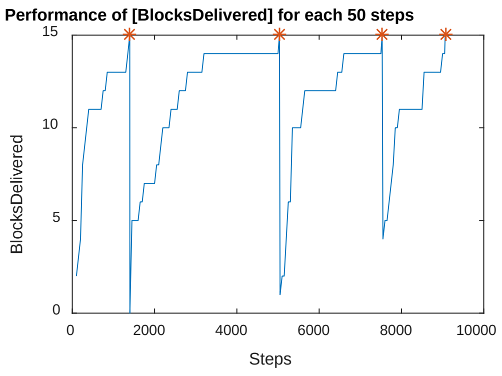
b.1 Interval Rewards for Run 1



b.2 Interval Rewards for Run 2



c.1 Blocks Delivered for Run 1



c.2 Blocks Delivered for Run 2

Figure E Performance Results of Experiment 6

Three performance measurements are overserved in every 50 step of Q- learning. Orange stars indicate the steps when PD world is reset to initial state as experiment required.