Group: Vegas

Christoph Eick

COSC 4368

April 15th, 2021

AI Group Project: Reinforcement Learning in Transportation World

Introduction:

This group project is used to learn and demonstrate how reinforcement learning can be used to discover paths in a transportation world. Reinforcement learning is a strategy in artificial learning used to train machine learning models into making a sequence of decisions; the programmer designs an algorithm that, when introduced with a challenge, the machine will apply trial and error to solve such problems. In reinforcement learning, the feedback the machine receives is called either reward or penalty; for example, in a maze, the program is always trying to maximize the reward, and for every step taken, there is a small penalty until it finds the goal, which is when it is awarded. Naturally, this will take many trials and the past experiences (episodes) will be used to eventually reach the goal with maximum reward.

There are many other factors to reinforcement learning the programmer should always bear in mind. Input is what is known as the initial state where the model will start, output will be the exit state or solutions to a particular problem, training is the method where the model returns as a solution and the programmer will choose to either reward or penalize it depending on the output. Reinforcement learning is always learning, in other words, the output is dependent on the current input and the next input will be dependent on the previous output.

There are two widely used learning models: Markov Decision Process, and Q-Learning.

This project will be mainly focused on using the Q-Learning model.

Q-Learning/SARSA:

This project implements Q-learning and SARSA. Q-learning is an off-policy learning algorithm and geared towards optimal behavior. Although this might not be realistic to accomplish in practice as in most applications, policies are needed that allow for some exploration. SARSA, however, uses the actually taken action for the update and is, therefore, more realistic as it uses the employed policy. However, it has problems with convergence. Q-values for SARSA and Q-learning can be mathematically calculated using the following equations:

$$\text{SARSA: } Q(a,s) = Q(a,s) + \alpha\,[R(s) + \gamma * Q(a',s') - Q(a,s)]$$

$$\text{Q-learning: } Q(a,s) = (1-\alpha) * Q(a,s) + \alpha[R(s',a,s) + \gamma * \max_a (Q(a',s'))\,]$$

The term $\max_a (Q(a',s'))$ allows the agent to select an action with a maximum q-value, thereby following a greedy approach.

Program Description:

The agent-based system designed in this project consists of two pickup locations (3,5) and (4,2), four drop-off locations (1,1), (1,5), (3,3,) and (5,5), and normal locations to move items from pickup to drop-off. The purpose of this task is to train the model using reinforcement learning such that over time, the system efficiently transports blocks from pickup to drop-off locations using fewer iterations. Five experiments were conducted, each using different policies including PRANDOM, PEXPLOIT, PGREEDY, and SARSA Q-learning. The program prints the steps taken to reach the respective location, and the number of blocks remaining at the pickup location or deposited at the drop-off location, along with the respective Q-tables. The efficiency of each run is determined by the number of times the agent reaches terminal state within a given number of steps. The performance is measured by the number of operators, the number of steps, applied to reach terminal state.

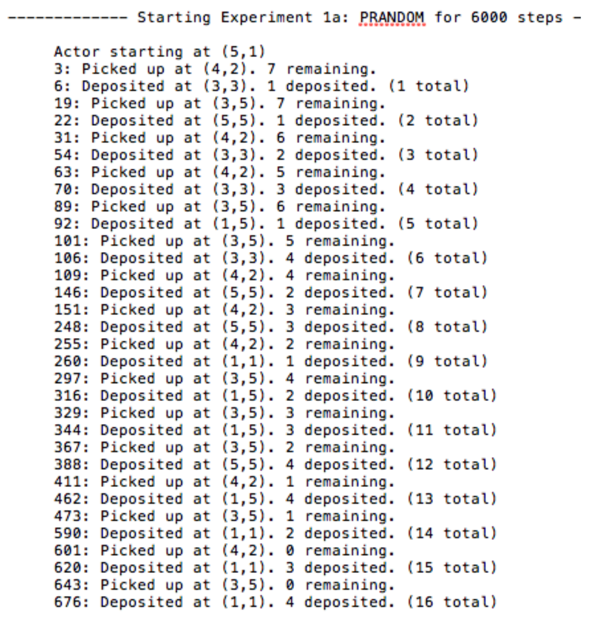
The agents start at (5,1). Each pickup location consists of 8 blocks while there are zero blocks at the drop-off location towards the start of the experiment. The program makes use of six different operators: North, South, East, West, Pickup, and Drop-off. North, South, East, and West are applicable movement directions for the agent in each state except for those that would allow the agent to leave the grid. Pickup is only applicable if the agent is in a pickup cell that contains at least one block and if the agent doesn't already carry a block. Similarly, drop-off is only applicable if the agent is already in a drop-off cell that contains less than four blocks and the agent carries a block. Rewards are assigned to every operator in order to determine the q-value. The reward to pick up or drop off a block in the correct state +13. When the operators North, South, East, and West are applied, a reward of -1 is applied. The performance of the agent is measured by the total reward and the number of operators applied to reach the terminal state from the initial state. The q-values of the paths taken by the agent may decrease as soon as the particular pickup location runs out of blocks or the particular drop-off location cannot store any more blocks. Therefore, an optimal result can be achieved when the reward is maximized and the number of operators used to reach the terminal state is minimum.

Experiments:

Experiment 1-a: Traditional Q-Learning algorithm is used in this experiment where alpha = 0.3 and gamma = 0.5. The algorithm runs for 6000 steps. Initially, PRANDOM policy is used for

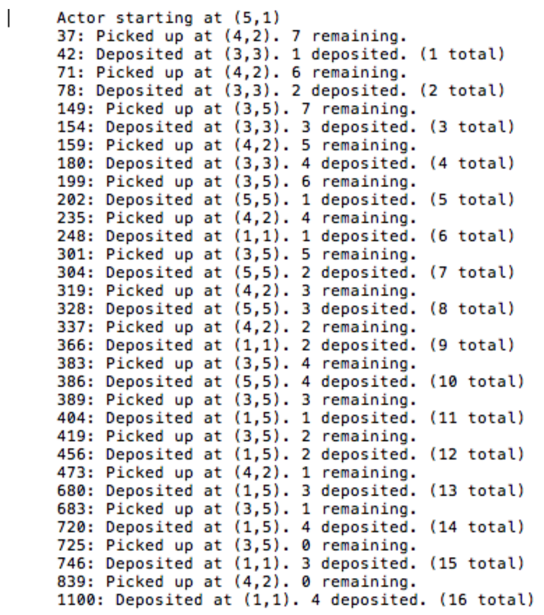
500 steps, followed by PRANDOM policy for 5500 more steps.

Run 1



```
------------- Starting Experiment 1a: PRANDOM for 6000 steps -

     Actor starting at (5,1)
     3: Picked up at (4,2). 7 remaining.
     6: Deposited at (3,3). 1 deposited. (1 total)
     19: Picked up at (3,5). 7 remaining.
     22: Deposited at (5,5). 1 deposited. (2 total)
     31: Picked up at (4,2). 6 remaining.
     54: Deposited at (3,3). 2 deposited. (3 total)
     63: Picked up at (4,2). 5 remaining.
     70: Deposited at (3,3). 3 deposited. (4 total)
     89: Picked up at (3,5). 6 remaining.
     92: Deposited at (1,5). 1 deposited. (5 total)
     101: Picked up at (3,5). 5 remaining.
     106: Deposited at (3,3). 4 deposited. (6 total)
     109: Picked up at (4,2). 4 remaining.
     146: Deposited at (5,5). 2 deposited. (7 total)
     151: Picked up at (4,2). 3 remaining.
     248: Deposited at (5,5). 3 deposited. (8 total)
     255: Picked up at (4,2). 2 remaining.
     260: Deposited at (1,1). 1 deposited. (9 total)
     297: Picked up at (3,5). 4 remaining.
     316: Deposited at (1,5). 2 deposited. (10 total)
     329: Picked up at (3,5). 3 remaining.
     344: Deposited at (1,5). 3 deposited. (11 total)
     367: Picked up at (3,5). 2 remaining.
     388: Deposited at (5,5). 4 deposited. (12 total)
     411: Picked up at (4,2). 1 remaining.
     462: Deposited at (1,5). 4 deposited. (13 total)
     473: Picked up at (3,5). 1 remaining.
     590: Deposited at (1,1). 2 deposited. (14 total)
     601: Picked up at (4,2). 0 remaining.
     620: Deposited at (1,1). 3 deposited. (15 total)
     643: Picked up at (3,5). 0 remaining.
     676: Deposited at (1,1). 4 deposited. (16 total)
```

Run 2

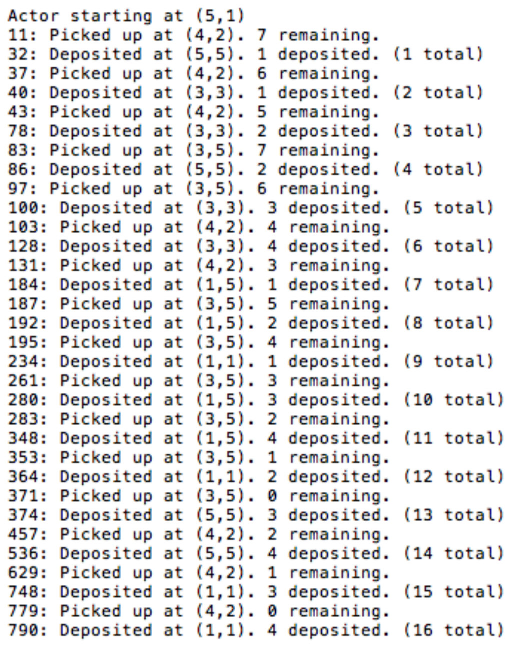


```
|   Actor starting at (5,1)
    37: Picked up at (4,2). 7 remaining.
    42: Deposited at (3,3). 1 deposited. (1 total)
    71: Picked up at (4,2). 6 remaining.
    78: Deposited at (3,3). 2 deposited. (2 total)
    149: Picked up at (3,5). 7 remaining.
    154: Deposited at (3,3). 3 deposited. (3 total)
    159: Picked up at (4,2). 5 remaining.
    180: Deposited at (3,3). 4 deposited. (4 total)
    199: Picked up at (3,5). 6 remaining.
    202: Deposited at (5,5). 1 deposited. (5 total)
    235: Picked up at (4,2). 4 remaining.
    248: Deposited at (1,1). 1 deposited. (6 total)
    301: Picked up at (3,5). 5 remaining.
    304: Deposited at (5,5). 2 deposited. (7 total)
    319: Picked up at (4,2). 3 remaining.
    328: Deposited at (5,5). 3 deposited. (8 total)
    337: Picked up at (4,2). 2 remaining.
    366: Deposited at (1,1). 2 deposited. (9 total)
    383: Picked up at (3,5). 4 remaining.
    386: Deposited at (5,5). 4 deposited. (10 total)
    389: Picked up at (3,5). 3 remaining.
    404: Deposited at (1,5). 1 deposited. (11 total)
    419: Picked up at (3,5). 2 remaining.
    456: Deposited at (1,5). 2 deposited. (12 total)
    473: Picked up at (4,2). 1 remaining.
    680: Deposited at (1,5). 3 deposited. (13 total)
    683: Picked up at (3,5). 1 remaining.
    720: Deposited at (1,5). 4 deposited. (14 total)
    725: Picked up at (3,5). 0 remaining.
    746: Deposited at (1,1). 3 deposited. (15 total)
    839: Picked up at (4,2). 0 remaining.
    1100: Deposited at (1,1). 4 deposited. (16 total)
```

In both runs, the agent reached the terminal state 7 times on average. Additionally, it can be said that about 800 to 850 steps are needed to successfully place the blocks in their place. This run shall serve to compare how other algorithms run and should be used as a basis for comparison. The random policy is moving around the state space until it finds a pickup cell, and eventually drops it off in a corresponding cell. Lastly, PRANDOM can be said to be somewhat efficient. This can be due to a reduced state space (only 25 cells).

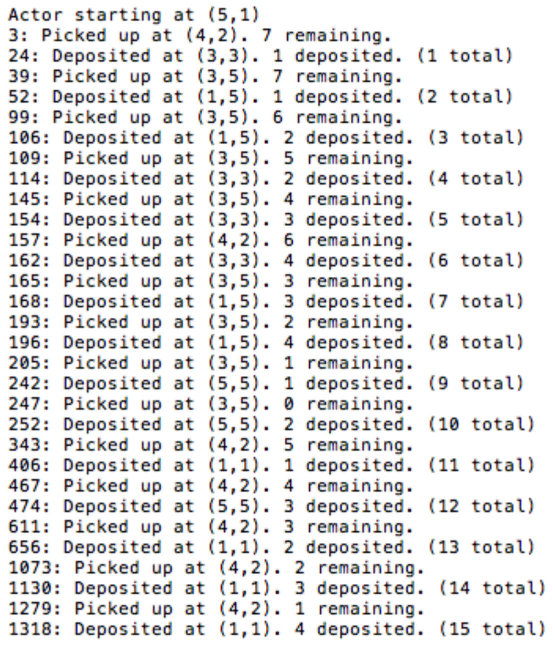
Experiment 1-b: Traditional Q-learning algorithm is used in this experiment where alpha = 0.3 and gamma = 0.5. The algorithm runs for 6000 steps. Initially, PRANDOM policy is used for 500 steps, followed by PGREEDY policy for 5500 more steps.

Run 1



```
Actor starting at (5,1)
11: Picked up at (4,2). 7 remaining.
32: Deposited at (5,5). 1 deposited. (1 total)
37: Picked up at (4,2). 6 remaining.
40: Deposited at (3,3). 1 deposited. (2 total)
43: Picked up at (4,2). 5 remaining.
78: Deposited at (3,3). 2 deposited. (3 total)
83: Picked up at (3,5). 7 remaining.
86: Deposited at (5,5). 2 deposited. (4 total)
97: Picked up at (3,5). 6 remaining.
100: Deposited at (3,3). 3 deposited. (5 total)
103: Picked up at (4,2). 4 remaining.
128: Deposited at (3,3). 4 deposited. (6 total)
131: Picked up at (4,2). 3 remaining.
184: Deposited at (1,5). 1 deposited. (7 total)
187: Picked up at (3,5). 5 remaining.
192: Deposited at (1,5). 2 deposited. (8 total)
195: Picked up at (3,5). 4 remaining.
234: Deposited at (1,1). 1 deposited. (9 total)
261: Picked up at (3,5). 3 remaining.
280: Deposited at (1,5). 3 deposited. (10 total)
283: Picked up at (3,5). 2 remaining.
348: Deposited at (1,5). 4 deposited. (11 total)
353: Picked up at (3,5). 1 remaining.
364: Deposited at (1,1). 2 deposited. (12 total)
371: Picked up at (3,5). 0 remaining.
374: Deposited at (5,5). 3 deposited. (13 total)
457: Picked up at (4,2). 2 remaining.
536: Deposited at (5,5). 4 deposited. (14 total)
629: Picked up at (4,2). 1 remaining.
748: Deposited at (1,1). 3 deposited. (15 total)
779: Picked up at (4,2). 0 remaining.
790: Deposited at (1,1). 4 deposited. (16 total)
```

Run 2

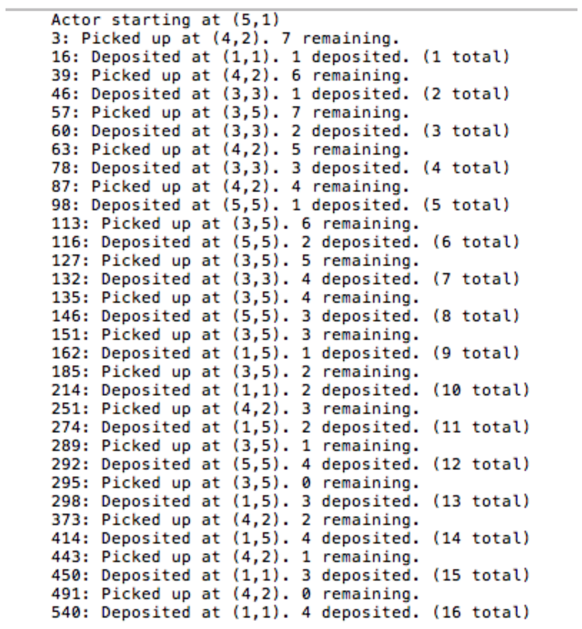


```
Actor starting at (5,1)
3: Picked up at (4,2). 7 remaining.
24: Deposited at (3,3). 1 deposited. (1 total)
39: Picked up at (3,5). 7 remaining.
52: Deposited at (1,5). 1 deposited. (2 total)
99: Picked up at (3,5). 6 remaining.
106: Deposited at (1,5). 2 deposited. (3 total)
109: Picked up at (3,5). 5 remaining.
114: Deposited at (3,3). 2 deposited. (4 total)
145: Picked up at (3,5). 4 remaining.
154: Deposited at (3,3). 3 deposited. (5 total)
157: Picked up at (4,2). 6 remaining.
162: Deposited at (3,3). 4 deposited. (6 total)
165: Picked up at (3,5). 3 remaining.
168: Deposited at (1,5). 3 deposited. (7 total)
193: Picked up at (3,5). 2 remaining.
196: Deposited at (1,5). 4 deposited. (8 total)
205: Picked up at (3,5). 1 remaining.
242: Deposited at (5,5). 1 deposited. (9 total)
247: Picked up at (3,5). 0 remaining.
252: Deposited at (5,5). 2 deposited. (10 total)
343: Picked up at (4,2). 5 remaining.
406: Deposited at (1,1). 1 deposited. (11 total)
467: Picked up at (4,2). 4 remaining.
474: Deposited at (5,5). 3 deposited. (12 total)
611: Picked up at (4,2). 3 remaining.
656: Deposited at (1,1). 2 deposited. (13 total)
1073: Picked up at (4,2). 2 remaining.
1130: Deposited at (1,1). 3 deposited. (14 total)
1279: Picked up at (4,2). 1 remaining.
1318: Deposited at (1,1). 4 deposited. (15 total)
```

In this experiment, the PRANDOM and PGREEDY are used in conjunction. First, the agent explores the state space to obtain utility for 500 steps using PRANDOM. Then, the agent switches to PGREEDY, which tries to maximize immediate reward. The first run suggests that the random policy allowed the agent to exercise 13 blocks. Then, after some 290 steps, the agent was able to complete its task. The data indicates two pieces of information from run 1. The first one is that the random policy is fairly good at exploring. The second key takeaway from run 1 is that the greedy policy lacks the ability to learn, but does accomplish the task. On run two, the agent found itself unable to find the last block. This could indicate that a path to pick and drop off should be implemented since a reinforced path to pick-up cell is inherently opposite to a reinforced path to drop-off cell. Having said this, it was found that the agent was able to complete its tasks in other randomly-seeded runs. Actually, the agent would finish its task more times than it wouldn't. PGREEDY did not perform well and will usually just reach its terminal state only once per 6000 steps (on average). Lastly, the greedy policy, which tries to maximize a path, is in itself doomed to fail about a path change. To increase the success rate, PGREEDY should have two different q-tables, one for a path towards a pick-up cell and one towards a drop-off cell.

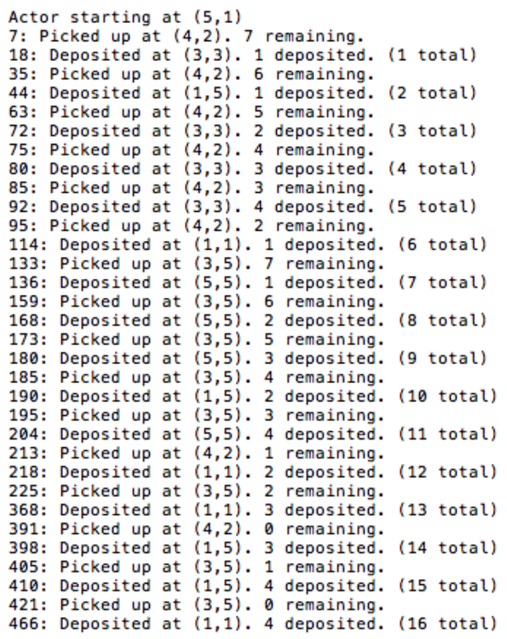
Experiment 1-c: Traditional Q-Learning algorithm is used in this experiment where alpha = 0.3 and gamma = 0.5. The algorithm runs for 6000 steps. Initially, PRANDOM policy is used for 500 steps, followed by PEXPLOIT policy for 5500 more steps.

Run 1



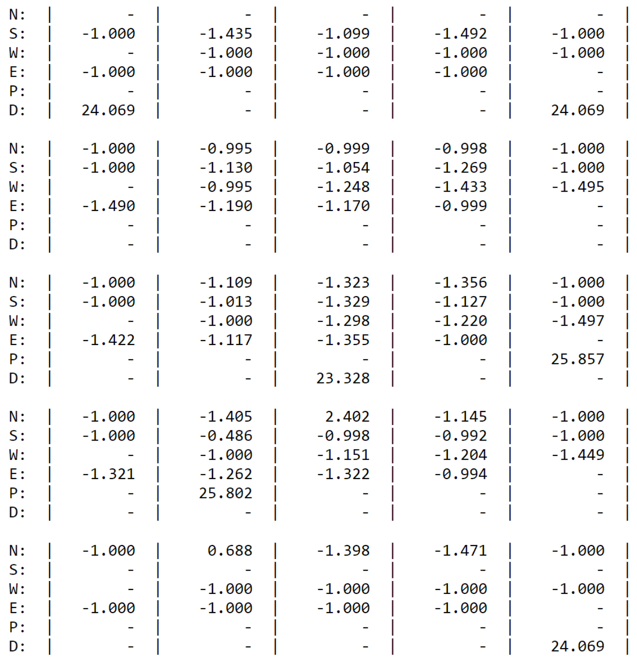
```
Actor starting at (5,1)
3: Picked up at (4,2). 7 remaining.
16: Deposited at (1,1). 1 deposited. (1 total)
39: Picked up at (4,2). 6 remaining.
46: Deposited at (3,3). 1 deposited. (2 total)
57: Picked up at (3,5). 7 remaining.
60: Deposited at (3,3). 2 deposited. (3 total)
63: Picked up at (4,2). 5 remaining.
78: Deposited at (3,3). 3 deposited. (4 total)
87: Picked up at (4,2). 4 remaining.
98: Deposited at (5,5). 1 deposited. (5 total)
113: Picked up at (3,5). 6 remaining.
116: Deposited at (5,5). 2 deposited. (6 total)
127: Picked up at (3,5). 5 remaining.
132: Deposited at (3,3). 4 deposited. (7 total)
135: Picked up at (3,5). 4 remaining.
146: Deposited at (5,5). 3 deposited. (8 total)
151: Picked up at (3,5). 3 remaining.
162: Deposited at (1,5). 1 deposited. (9 total)
185: Picked up at (3,5). 2 remaining.
214: Deposited at (1,1). 2 deposited. (10 total)
251: Picked up at (4,2). 3 remaining.
274: Deposited at (1,5). 2 deposited. (11 total)
289: Picked up at (3,5). 1 remaining.
292: Deposited at (5,5). 4 deposited. (12 total)
295: Picked up at (3,5). 0 remaining.
298: Deposited at (1,5). 3 deposited. (13 total)
373: Picked up at (4,2). 2 remaining.
414: Deposited at (1,5). 4 deposited. (14 total)
443: Picked up at (4,2). 1 remaining.
450: Deposited at (1,1). 3 deposited. (15 total)
491: Picked up at (4,2). 0 remaining.
540: Deposited at (1,1). 4 deposited. (16 total)
```

Run 2



```
Actor starting at (5,1)
7: Picked up at (4,2). 7 remaining.
18: Deposited at (3,3). 1 deposited. (1 total)
35: Picked up at (4,2). 6 remaining.
44: Deposited at (1,5). 1 deposited. (2 total)
63: Picked up at (4,2). 5 remaining.
72: Deposited at (3,3). 2 deposited. (3 total)
75: Picked up at (4,2). 4 remaining.
80: Deposited at (3,3). 3 deposited. (4 total)
85: Picked up at (4,2). 3 remaining.
92: Deposited at (3,3). 4 deposited. (5 total)
95: Picked up at (4,2). 2 remaining.
114: Deposited at (1,1). 1 deposited. (6 total)
133: Picked up at (3,5). 7 remaining.
136: Deposited at (5,5). 1 deposited. (7 total)
159: Picked up at (3,5). 6 remaining.
168: Deposited at (5,5). 2 deposited. (8 total)
173: Picked up at (3,5). 5 remaining.
180: Deposited at (5,5). 3 deposited. (9 total)
185: Picked up at (3,5). 4 remaining.
190: Deposited at (1,5). 2 deposited. (10 total)
195: Picked up at (3,5). 3 remaining.
204: Deposited at (5,5). 4 deposited. (11 total)
213: Picked up at (4,2). 1 remaining.
218: Deposited at (1,1). 2 deposited. (12 total)
225: Picked up at (3,5). 2 remaining.
368: Deposited at (1,1). 3 deposited. (13 total)
391: Picked up at (4,2). 0 remaining.
398: Deposited at (1,5). 3 deposited. (14 total)
405: Picked up at (3,5). 1 remaining.
410: Deposited at (1,5). 4 deposited. (15 total)
421: Picked up at (3,5). 0 remaining.
466: Deposited at (1,1). 4 deposited. (16 total)
```

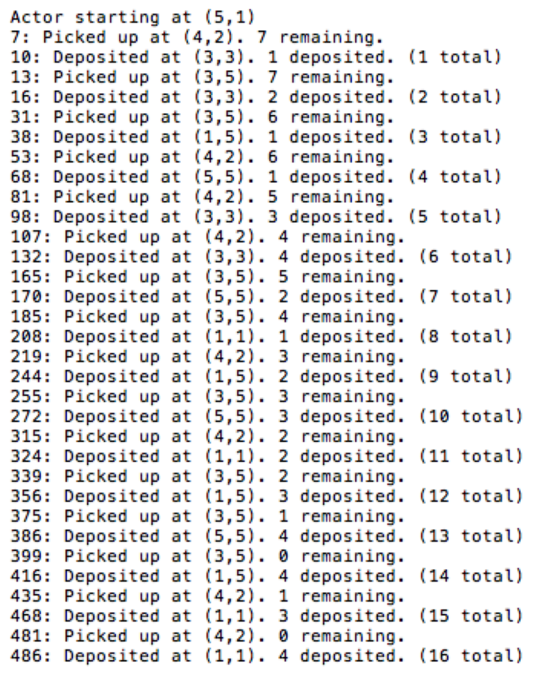
The figure below is the Q-table for experiment 1c after reaching the terminal state three times. The interpretation of this table shall start. Firstly, PRANDOM again seems to understand the state space, exploring and gathering utility values. This fortifies the idea that the state space is small. If more research is to be done, a bigger state-space should be explored. In this world, the agent starts on cell (5,1) or the bottom left-most cell; the discussion shall start there. The best path, according to the exploit policy, was found where the agent preferentially picks a path from a pick-up to a drop-off until that location is full, i.e. the cell has 4 blocks. Then, since PEXPLOIT allows for a 20% exploration path, some other path will be explored and possibly "learned" a better route. Having stated this, the Q-table allows one to see that the agent will preferentially move north, then east to pick up a block. From that cell, the agent will move south. Here a conflicting path appears, but this is due to the overlapping going to a drop-off and returning to a pick-up cell. This is why cells (4,2) and (5,2) have rebounding best utility. In other words, cell (4,2) indicates that the best move is south, similarly, cell (5,2) indicates that the best move is north. Carefully examining this scenario is the key to understanding this policy. The path the agent learned was to pick up a block in cell (4,2) and to drop it at (5,5). This creates a two-way route, yet since the agent was able to successfully complete the task, the policy must have forced the agent out of its "learned way" to a new path. Thus, this shows that the agent will also try to drop off at cell (3,3) and also pick up at cell (3,5). The exploit policy is a great way to maximize results but avoid adversarial behavior in the paths. The erudite agent experiences optimal paths, but is also allowed to explore. This resembles real-life where students who only learn what is taught will ultimately fail to explore new ideas. In summary, the Q-table does not explicitly show the best routes because these can be conflicted for picking-up and dropping-off, yet knowing this resolves the issues of antagonistic utility *best* routes.



```
N:  |        -  |        -  |        -  |        -  |        -  |
S:  |   -1.000  |   -1.435  |   -1.099  |   -1.492  |   -1.000  |
W:  |        -  |   -1.000  |   -1.000  |   -1.000  |   -1.000  |
E:  |   -1.000  |   -1.000  |   -1.000  |   -1.000  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |   24.069  |        -  |        -  |        -  |   24.069  |

N:  |   -1.000  |   -0.995  |   -0.999  |   -0.998  |   -1.000  |
S:  |   -1.000  |   -1.130  |   -1.054  |   -1.269  |   -1.000  |
W:  |        -  |   -0.995  |   -1.248  |   -1.433  |   -1.495  |
E:  |   -1.490  |   -1.190  |   -1.170  |   -0.999  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |        -  |

N:  |   -1.000  |   -1.109  |   -1.323  |   -1.356  |   -1.000  |
S:  |   -1.000  |   -1.013  |   -1.329  |   -1.127  |   -1.000  |
W:  |        -  |   -1.000  |   -1.298  |   -1.220  |   -1.497  |
E:  |   -1.422  |   -1.117  |   -1.355  |   -1.000  |        -  |
P:  |        -  |        -  |        -  |        -  |   25.857  |
D:  |        -  |        -  |   23.328  |        -  |        -  |

N:  |   -1.000  |   -1.405  |    2.402  |   -1.145  |   -1.000  |
S:  |   -1.000  |   -0.486  |   -0.998  |   -0.992  |   -1.000  |
W:  |        -  |   -1.000  |   -1.151  |   -1.204  |   -1.449  |
E:  |   -1.321  |   -1.262  |   -1.322  |   -0.994  |        -  |
P:  |        -  |   25.802  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |        -  |

N:  |   -1.000  |    0.688  |   -1.398  |   -1.471  |   -1.000  |
S:  |        -  |        -  |        -  |        -  |        -  |
W:  |        -  |   -1.000  |   -1.000  |   -1.000  |   -1.000  |
E:  |   -1.000  |   -1.000  |   -1.000  |   -1.000  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |   24.069  |
```

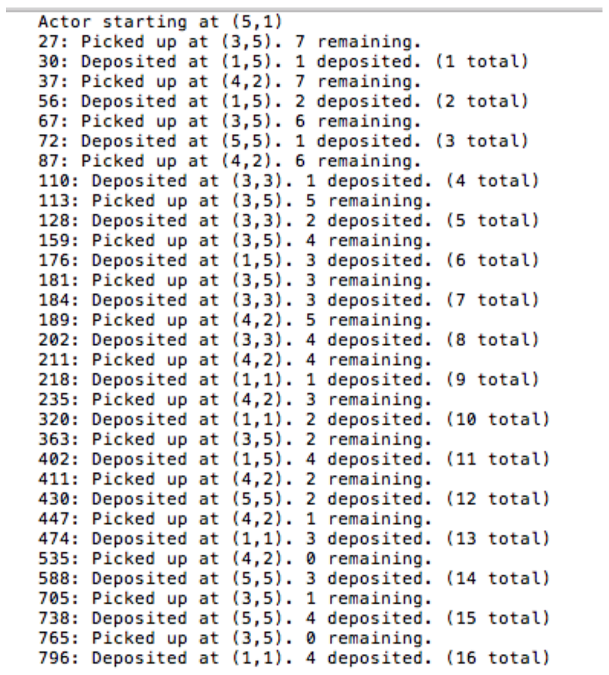
Experiment 2: SARSA Q-learning algorithm is used in this experiment where alpha = 0.3 and gamma = 0.5. The algorithm runs for 6000 steps.

Run 1



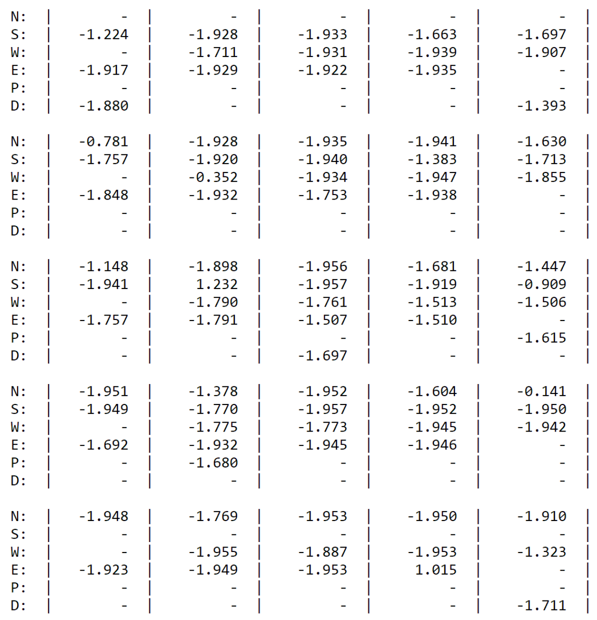
```
Actor starting at (5,1)
7: Picked up at (4,2). 7 remaining.
10: Deposited at (3,3). 1 deposited. (1 total)
13: Picked up at (3,5). 7 remaining.
16: Deposited at (3,3). 2 deposited. (2 total)
31: Picked up at (3,5). 6 remaining.
38: Deposited at (1,5). 1 deposited. (3 total)
53: Picked up at (4,2). 6 remaining.
68: Deposited at (5,5). 1 deposited. (4 total)
81: Picked up at (4,2). 5 remaining.
98: Deposited at (3,3). 3 deposited. (5 total)
107: Picked up at (4,2). 4 remaining.
132: Deposited at (3,3). 4 deposited. (6 total)
165: Picked up at (3,5). 5 remaining.
170: Deposited at (5,5). 2 deposited. (7 total)
185: Picked up at (3,5). 4 remaining.
208: Deposited at (1,1). 1 deposited. (8 total)
219: Picked up at (4,2). 3 remaining.
244: Deposited at (1,5). 2 deposited. (9 total)
255: Picked up at (3,5). 3 remaining.
272: Deposited at (5,5). 3 deposited. (10 total)
315: Picked up at (4,2). 2 remaining.
324: Deposited at (1,1). 2 deposited. (11 total)
339: Picked up at (3,5). 2 remaining.
356: Deposited at (1,5). 3 deposited. (12 total)
375: Picked up at (3,5). 1 remaining.
386: Deposited at (5,5). 4 deposited. (13 total)
399: Picked up at (3,5). 0 remaining.
416: Deposited at (1,5). 4 deposited. (14 total)
435: Picked up at (4,2). 1 remaining.
468: Deposited at (1,1). 3 deposited. (15 total)
481: Picked up at (4,2). 0 remaining.
486: Deposited at (1,1). 4 deposited. (16 total)
```

Run 2



```
Actor starting at (5,1)
27: Picked up at (3,5). 7 remaining.
30: Deposited at (1,5). 1 deposited. (1 total)
37: Picked up at (4,2). 7 remaining.
56: Deposited at (1,5). 2 deposited. (2 total)
67: Picked up at (3,5). 6 remaining.
72: Deposited at (5,5). 1 deposited. (3 total)
87: Picked up at (4,2). 6 remaining.
110: Deposited at (3,3). 1 deposited. (4 total)
113: Picked up at (3,5). 5 remaining.
128: Deposited at (3,3). 2 deposited. (5 total)
159: Picked up at (3,5). 4 remaining.
176: Deposited at (1,5). 3 deposited. (6 total)
181: Picked up at (3,5). 3 remaining.
184: Deposited at (3,3). 3 deposited. (7 total)
189: Picked up at (4,2). 5 remaining.
202: Deposited at (3,3). 4 deposited. (8 total)
211: Picked up at (4,2). 4 remaining.
218: Deposited at (1,1). 1 deposited. (9 total)
235: Picked up at (4,2). 3 remaining.
320: Deposited at (1,1). 2 deposited. (10 total)
363: Picked up at (3,5). 2 remaining.
402: Deposited at (1,5). 4 deposited. (11 total)
411: Picked up at (4,2). 2 remaining.
430: Deposited at (5,5). 2 deposited. (12 total)
447: Picked up at (4,2). 1 remaining.
474: Deposited at (1,1). 3 deposited. (13 total)
535: Picked up at (4,2). 0 remaining.
588: Deposited at (5,5). 3 deposited. (14 total)
705: Picked up at (3,5). 1 remaining.
738: Deposited at (5,5). 4 deposited. (15 total)
765: Picked up at (3,5). 0 remaining.
796: Deposited at (1,1). 4 deposited. (16 total)
```

This experiment compares how an agent does using an on-policy SARSA algorithm and simultaneously can be compared to the previous first experiment. It is well established that SARSA will converge while Q-Learning will fail to do so *sometimes*. This means SARSA approaches a *true* value faster. The following Q-table is from experiment 2 after reaching the terminal state five times. As one can see, since SARSA is not greedy, it will not reinforce preferential paths. This ultimately leads to conflicting paths but is inherently better to mitigate drop-off locations that are full. In other words, since a drop-off cell can only hold a maximum of 4 blocks, and a path was learned before the 4 blocks were placed, the agent discovered a path to take during the non-filled cell. A switch in the learning happens since that path is not desired. As



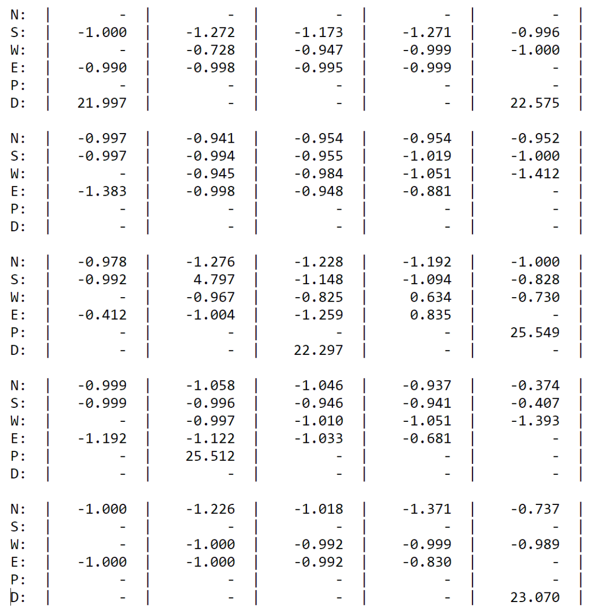
```
N:  |      -  |      -  |      -  |      -  |      -  |
S:  | -1.224  | -1.928  | -1.933  | -1.663  | -1.697  |
W:  |      -  | -1.711  | -1.931  | -1.939  | -1.907  |
E:  | -1.917  | -1.929  | -1.922  | -1.935  |      -  |
P:  |      -  |      -  |      -  |      -  |      -  |
D:  | -1.880  |      -  |      -  |      -  | -1.393  |

N:  | -0.781  | -1.928  | -1.935  | -1.941  | -1.630  |
S:  | -1.757  | -1.920  | -1.940  | -1.383  | -1.713  |
W:  |      -  | -0.352  | -1.934  | -1.947  | -1.855  |
E:  | -1.848  | -1.932  | -1.753  | -1.938  |      -  |
P:  |      -  |      -  |      -  |      -  |      -  |
D:  |      -  |      -  |      -  |      -  |      -  |

N:  | -1.148  | -1.898  | -1.956  | -1.681  | -1.447  |
S:  | -1.941  |  1.232  | -1.957  | -1.919  | -0.909  |
W:  |      -  | -1.790  | -1.761  | -1.513  | -1.506  |
E:  | -1.757  | -1.791  | -1.507  | -1.510  |      -  |
P:  |      -  |      -  |      -  |      -  | -1.615  |
D:  |      -  |      -  | -1.697  |      -  |      -  |

N:  | -1.951  | -1.378  | -1.952  | -1.604  | -0.141  |
S:  | -1.949  | -1.770  | -1.957  | -1.952  | -1.950  |
W:  |      -  | -1.775  | -1.773  | -1.945  | -1.942  |
E:  | -1.692  | -1.932  | -1.945  | -1.946  |      -  |
P:  |      -  | -1.680  |      -  |      -  |      -  |
D:  |      -  |      -  |      -  |      -  |      -  |

N:  | -1.948  | -1.769  | -1.953  | -1.950  | -1.910  |
S:  |      -  |      -  |      -  |      -  |      -  |
W:  |      -  | -1.955  | -1.887  | -1.953  | -1.323  |
E:  | -1.923  | -1.949  | -1.953  |  1.015  |      -  |
P:  |      -  |      -  |      -  |      -  |      -  |
D:  |      -  |      -  |      -  |      -  | -1.711  |
```

opposed to a greedy approach, SARSA policy can understand the "change in path" necessary to translate blocks to a different drop-off cell. This translates well to show that even when previously gained knowledge (from previously reached terminal states) the agent can adapt and explore a new path towards a successful task. By using the average of terminal states reached, it can be said that SARSA is better performing than PEXPLOIT. PEXPLOIT has more deviation on the steps required to finish moving the blocks. In other words, SARSA does converge naturally, while the traditional Q-Learning (especially using random approach) will sometimes converge (faster), similarly, or not converge to attain a successful run. The traditional Q-Learning algorithms may be faster, but there is a trade-off in which the routes learned are not optimal.

<u>Experiment 3:</u> Traditional Q-Learning algorithm is used in this experiment where alpha = 0.3, 0.15 and 0.45, and gamma = 0.5. The algorithm runs for 6000 steps. Initially, PRANDOM policy is used for 500 steps, followed by PEXPLOIT policy for 5500 more steps.

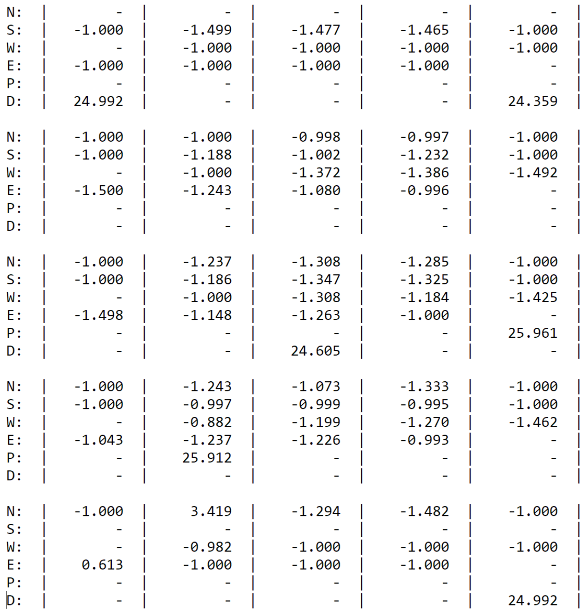
The following three figures show the Q-tables developed by changing the alpha value. In order of appearance, the alpha values are 0.15, 0.3, and 0.45. Respectively, the number of terminal states reached in 6000 steps were six, four, and three. To illustrate how the alpha value affects Q-Learning tables, this experiment should suffice. Alpha is labeled as the learning rate. This means that increasing alpha is desired to a certain point because an alpha too low will ultimately lead to an agent *not* learning a path, while an alpha near one will lead to overlearning;

not understanding the correct pattern.



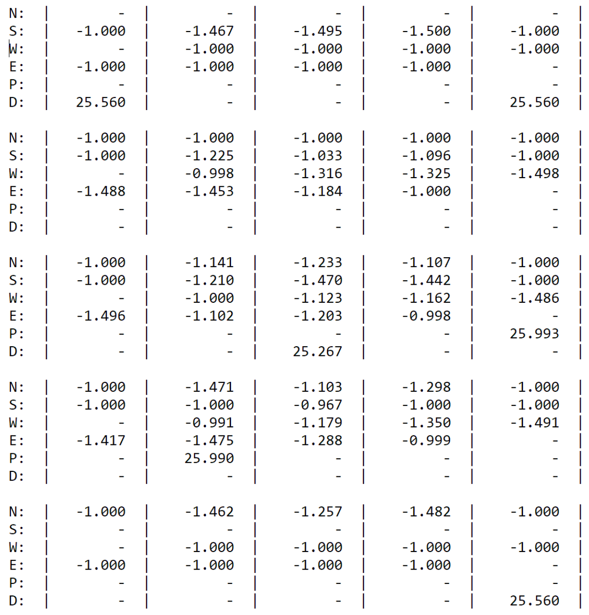
```
N:  |        -  |        -  |        -  |        -  |        -  |
S:  |   -1.000  |   -1.272  |   -1.173  |   -1.271  |   -0.996  |
W:  |        -  |   -0.728  |   -0.947  |   -0.999  |   -1.000  |
E:  |   -0.990  |   -0.998  |   -0.995  |   -0.999  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |   21.997  |        -  |        -  |        -  |   22.575  |

N:  |   -0.997  |   -0.941  |   -0.954  |   -0.954  |   -0.952  |
S:  |   -0.997  |   -0.994  |   -0.955  |   -1.019  |   -1.000  |
W:  |        -  |   -0.945  |   -0.984  |   -1.051  |   -1.412  |
E:  |   -1.383  |   -0.998  |   -0.948  |   -0.881  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |        -  |

N:  |   -0.978  |   -1.276  |   -1.228  |   -1.192  |   -1.000  |
S:  |   -0.992  |    4.797  |   -1.148  |   -1.094  |   -0.828  |
W:  |        -  |   -0.967  |   -0.825  |    0.634  |   -0.730  |
E:  |   -0.412  |   -1.004  |   -1.259  |    0.835  |        -  |
P:  |        -  |        -  |        -  |        -  |   25.549  |
D:  |        -  |        -  |   22.297  |        -  |        -  |

N:  |   -0.999  |   -1.058  |   -1.046  |   -0.937  |   -0.374  |
S:  |   -0.999  |   -0.996  |   -0.946  |   -0.941  |   -0.407  |
W:  |        -  |   -0.997  |   -1.010  |   -1.051  |   -1.393  |
E:  |   -1.192  |   -1.122  |   -1.033  |   -0.681  |        -  |
P:  |        -  |   25.512  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |        -  |

N:  |   -1.000  |   -1.226  |   -1.018  |   -1.371  |   -0.737  |
S:  |        -  |        -  |        -  |        -  |        -  |
W:  |        -  |   -1.000  |   -0.992  |   -0.999  |   -0.989  |
E:  |   -1.000  |   -1.000  |   -0.992  |   -0.830  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |   23.070  |
```

To fully comprehend how alpha shapes utility values around the state space present, several runs were completed. The runs that set alpha to 0.3 seemed to be the most stable and predictable; the number of terminal states reached was four times (always for all the times ran). Having varying levels of performance with different alpha values, demonstrates how its value affects Q-Learning. Yet a few patterns were noted. Firstly, let's recall that PEXPLOIT on average reached terminal state three times (alpha =0.3). Using this as a benchmark metric, the nature of how alpha shapes Q-tables can be uncovered. Firstly, one can notice that lower alpha leads to lower rewards in pick-up and drop-off cells. A reasonable assumption is that lower alpha values will tend to converge (slow).



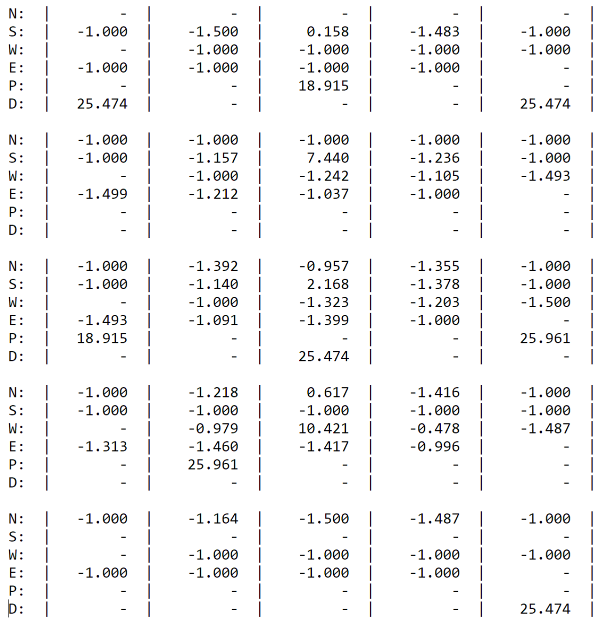
```
N:  |        -  |        -  |        -  |        -  |        -  |
S:  |   -1.000  |   -1.499  |   -1.477  |   -1.465  |   -1.000  |
W:  |        -  |   -1.000  |   -1.000  |   -1.000  |   -1.000  |
E:  |   -1.000  |   -1.000  |   -1.000  |   -1.000  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |   24.992  |        -  |        -  |        -  |   24.359  |

N:  |   -1.000  |   -1.000  |   -0.998  |   -0.997  |   -1.000  |
S:  |   -1.000  |   -1.188  |   -1.002  |   -1.232  |   -1.000  |
W:  |        -  |   -1.000  |   -1.372  |   -1.386  |   -1.492  |
E:  |   -1.500  |   -1.243  |   -1.080  |   -0.996  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |        -  |

N:  |   -1.000  |   -1.237  |   -1.308  |   -1.285  |   -1.000  |
S:  |   -1.000  |   -1.186  |   -1.347  |   -1.325  |   -1.000  |
W:  |        -  |   -1.000  |   -1.308  |   -1.184  |   -1.425  |
E:  |   -1.498  |   -1.148  |   -1.263  |   -1.000  |        -  |
P:  |        -  |        -  |        -  |        -  |   25.961  |
D:  |        -  |        -  |   24.605  |        -  |        -  |

N:  |   -1.000  |   -1.243  |   -1.073  |   -1.333  |   -1.000  |
S:  |   -1.000  |   -0.997  |   -0.999  |   -0.995  |   -1.000  |
W:  |        -  |   -0.882  |   -1.199  |   -1.270  |   -1.462  |
E:  |   -1.043  |   -1.237  |   -1.226  |   -0.993  |        -  |
P:  |        -  |   25.912  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |        -  |

N:  |   -1.000  |    3.419  |   -1.294  |   -1.482  |   -1.000  |
S:  |        -  |        -  |        -  |        -  |        -  |
W:  |        -  |   -0.982  |   -1.000  |   -1.000  |   -1.000  |
E:  |    0.613  |   -1.000  |   -1.000  |   -1.000  |        -  |
P:  |        -  |        -  |        -  |        -  |        -  |
D:  |        -  |        -  |        -  |        -  |   24.992  |
```

This suggests a hypothesis that PEXPLOIT and SARSA are similar if alpha is low for the agent to learn a path sufficiently well to converge. On the other hand, the last part of experiment three has a high alpha value. While its value is still relatively low, it is exemplary on how increasing alpha will tend to overestimate rewarding cells. Since the state space is small, this could lead to faster convergence. Yet, its performance metric varied much more than the previous trials. Notice how the Q-tables change values, especially on the drop-off and pick-up cells. This indicates that as alpha increases, a preferential path will be chosen by the agent even if it is not the *best*. In other words, if a path happens to be chosen randomly (20% chance of happening) multiple times, that route will be reinforced more with a higher alpha, which may eventually lead to an agent failing (or taking longer steps). Similarly, if the agent happens to randomly choose the *best* path, such path will be reinforced and lead to fewer steps for the event to converge. This explains the high deviance of the last part of this experiment.



```
N: |        - |        - |        - |        - |        - |
S: |   -1.000 |   -1.467 |   -1.495 |   -1.500 |   -1.000 |
W: |        - |   -1.000 |   -1.000 |   -1.000 |   -1.000 |
E: |   -1.000 |   -1.000 |   -1.000 |   -1.000 |        - |
P: |        - |        - |        - |        - |        - |
D: |   25.560 |        - |        - |        - |   25.560 |

N: |   -1.000 |   -1.000 |   -1.000 |   -1.000 |   -1.000 |
S: |   -1.000 |   -1.225 |   -1.033 |   -1.096 |   -1.000 |
W: |        - |   -0.998 |   -1.316 |   -1.325 |   -1.498 |
E: |   -1.488 |   -1.453 |   -1.184 |   -1.000 |        - |
P: |        - |        - |        - |        - |        - |
D: |        - |        - |        - |        - |        - |

N: |   -1.000 |   -1.141 |   -1.233 |   -1.107 |   -1.000 |
S: |   -1.000 |   -1.210 |   -1.470 |   -1.442 |   -1.000 |
W: |        - |   -1.000 |   -1.123 |   -1.162 |   -1.486 |
E: |   -1.496 |   -1.102 |   -1.203 |   -0.998 |        - |
P: |        - |        - |        - |        - |   25.993 |
D: |        - |        - |   25.267 |        - |        - |

N: |   -1.000 |   -1.471 |   -1.103 |   -1.298 |   -1.000 |
S: |   -1.000 |   -1.000 |   -0.967 |   -1.000 |   -1.000 |
W: |        - |   -0.991 |   -1.179 |   -1.350 |   -1.491 |
E: |   -1.417 |   -1.475 |   -1.288 |   -0.999 |        - |
P: |        - |   25.990 |        - |        - |        - |
D: |        - |        - |        - |        - |        - |

N: |   -1.000 |   -1.462 |   -1.257 |   -1.482 |   -1.000 |
S: |        - |        - |        - |        - |        - |
W: |        - |   -1.000 |   -1.000 |   -1.000 |   -1.000 |
E: |   -1.000 |   -1.000 |   -1.000 |   -1.000 |        - |
P: |        - |        - |        - |        - |        - |
D: |        - |        - |        - |        - |   25.560 |
```

Experiment 4: Traditional Q-Learning algorithm is used in this experiment where alpha = 0.3 and gamma = 0.5. The algorithm runs for 6000 steps. Initially, PRANDOM policy is used for 500 steps, followed by PEXPLOIT policy for 5500 more steps. However, after the agent reaches a terminal state the third time, pick-up locations are changed to (3,1) and (1,3); the drop off locations and the Q-table remain unchanged; finally, the program continues to run PEXPLOIT with the "new" pickup locations until the agent reaches a terminal state the sixth time.



```
N: |        - |        - |        - |        - |        - |
S: |   -1.000 |   -1.500 |    0.158 |   -1.483 |   -1.000 |
W: |        - |   -1.000 |   -1.000 |   -1.000 |   -1.000 |
E: |   -1.000 |   -1.000 |   -1.000 |   -1.000 |        - |
P: |        - |        - |   18.915 |        - |        - |
D: |   25.474 |        - |        - |        - |   25.474 |

N: |   -1.000 |   -1.000 |   -1.000 |   -1.000 |   -1.000 |
S: |   -1.000 |   -1.157 |    7.440 |   -1.236 |   -1.000 |
W: |        - |   -1.000 |   -1.242 |   -1.105 |   -1.493 |
E: |   -1.499 |   -1.212 |   -1.037 |   -1.000 |        - |
P: |        - |        - |        - |        - |        - |
D: |        - |        - |        - |        - |        - |

N: |   -1.000 |   -1.392 |   -0.957 |   -1.355 |   -1.000 |
S: |   -1.000 |   -1.140 |    2.168 |   -1.378 |   -1.000 |
W: |        - |   -1.000 |   -1.323 |   -1.203 |   -1.500 |
E: |   -1.493 |   -1.091 |   -1.399 |   -1.000 |        - |
P: |   18.915 |        - |        - |        - |   25.961 |
D: |        - |        - |   25.474 |        - |        - |

N: |   -1.000 |   -1.218 |    0.617 |   -1.416 |   -1.000 |
S: |   -1.000 |   -1.000 |   -1.000 |   -1.000 |   -1.000 |
W: |        - |   -0.979 |   10.421 |   -0.478 |   -1.487 |
E: |   -1.313 |   -1.460 |   -1.417 |   -0.996 |        - |
P: |        - |   25.961 |        - |        - |        - |
D: |        - |        - |        - |        - |        - |

N: |   -1.000 |   -1.164 |   -1.500 |   -1.487 |   -1.000 |
S: |        - |        - |        - |        - |        - |
W: |        - |   -1.000 |   -1.000 |   -1.000 |   -1.000 |
E: |   -1.000 |   -1.000 |   -1.000 |   -1.000 |        - |
P: |        - |        - |        - |        - |        - |
D: |        - |        - |        - |        - |   25.474 |
```

The Q-table above shows the utility values after reaching the terminal state six times (average). This experiment is suggesting how the agent “re-learns” to find new pick-up locations. To determine the effects of changing the pickup location, and thereby the rewards, this experiment’s results will be compared to that of the experiment 1c. For the first three terminal states reached, the number of operations applied, the number of steps, were very similar to experiment 1c. However, in trying to reach the terminal state with different pickup locations in run 4 with the same q-table, the agent makes significantly more operator applications to get to the terminal state. This increase in the number of steps is reasonable as the agent has to unlearn and relearn ideal paths to high rewards and it coincides with PEXPLOIT’s 20% chance of exploration. Once the locations change, the agent has a 20% chance to deviate from the previously found optimal path and explore a potentially better path. In the next few runs, the agent’s ability to adapt to the new pickup location is exemplified. After having been able to successfully complete one runthrough with the new pickup locations, the agent is able to adapt it’s path to make fewer operator applications before reaching a terminal state. On average, the number of steps taken to reach the sixth terminal state in this experiment is similar to the number of steps in experiment 1c. This suggests that PEXPLOIT adapts to changes in the environment relatively well.

Conclusion:

The experiments with the best results were PRANDOM, then PEXPLOIT with alpha as 0.15, then PEXPLOIT as 0.30, and SARSA. The PRANDOM result is surprising but can be attributed to a small state space, thus this policy won’t perform well in larger state space. The PEXPLOIT algorithm was very stable in performance, yet the alpha value of 0.15 achieved great results even when conflicting paths arose. This means PEXPLOIT is able to discover new routes, and still learn appealing paths (supported by findings in experiment 4). Lastly, SARSA was consistent, meaning that the results did not really deviate much. On the other hand, PGREEDY was not successful in some runs and thus unable to adapt to a changing world.

While it is hard to identify attractive paths for the agent to complete its task successfully, there are a few noteworthy things that could be distinguished throughout the analysis. Firstly, it was noted that the starting cell did have a preferential behavior. The agent could translate itself north or east without much utility. Then the pick-up location at (4,2) suggests having a preferential path to drop-off cells in a circular manner. Yet, these circular paths have an adversarial direction. This may be the reason why it is hard to distinguish paths. Moreover, the pick-up location (3,5) has direct adversarial path utilities. This means that a cell is in between the drop-off cells. The agent will want to pick up and drop with as little movement as possible, thus conflicting paths will arise in the neighboring cells of such pick-up location. Putting all the appealing paths together will eventually lead to a mess, and hard to determine an appealing path. To mitigate this, a path analysis for each time an agent has a block from a certain cell could shed light on appealing paths. As also mentioned, having different Q-tables for going and return to pick-up should lead to more clarity on appealing paths.

The purpose of experiment 1 is to explore different applications of Q-Learning algorithms and the balance of exploration vs. exploitation. PRANDOM, which is mainly exploration of the state space was found to be the most efficient in terms of operator application and the number of terminal states able to be achieved within the 6000 iteration limit. This was stated to be most likely due to the small space in which the agent needs to explore. On the other hand, PGREEDY was found to be the worst algorithm. Upon closer analysis of the steps taken, the algorithm was unable to find a drop-off location with a pure PGREEDY search due to its lack of exploration.

PEXPLOIT uses a combination of exploration and exploitation to explore the world and get rewards. Similarly, in experiment 2, the use of the SARSA algorithm changes only how the Q-values get updated in the table. These two experiments both reach the goal state four times within the given 6000 step iterations illustrating the effects of exploration and exploitation used in conjunction. However, PEXPLOIT has varying degrees of success when looking at performance based on the number of operators applied. Some runs perform as well as PRANDOM search while others in the same world can take up to four times as long. With the SARSA algorithm, performance is much more consistent. Between these two experiments, the goal state can be reached the same number of times within the given number of iterations but SARSA offers more performance consistency while Q-Learning performance has a chance to give better results.

Experiments 3 and 4 looked at how different factors can affect the performance of Q-Learning. Experiment 3 sheds light upon the effects of alpha values. It was determined that lower alpha values tend to be able to successfully converge. There are a few implications to keep in mind. The first one is that, since the random policy is used for 500 steps, Q-values will not gravitate towards an undesired path when alpha is low. The other important factor to keep in mind is that high alpha values could set the agent on successful paths sometimes, but not most of the time, faster than lower alpha values in Q-tables. Similarly, experiment 4 elucidate about how the agent re-learns new paths. While the state space is small and Q-tables difficult to interpret due to the conflicting path arising from the reduced state space, it can be stated that the agent is able to learn a new path. This is supported by the idea that all runs were successful.

Contributors:

Joann Pham: 1589075

Max Broekhuis: 1808005

Pablo Bribiesca R.: 1597268

Samita Damani: 1980918

Muaz Bourhani: 1869692