Project 2 Reinforcement Learning
Mohit G Kaduskar 1544187

Introduction

The most basic concept on which reinforcement learning stands is reinforcement or **reward**. Every **state** has various **actions** associated with it. Hence every action with respect to the states have rewards tagged with it this serves as feedback to either encourage or discourage the agent from taking similar actions or not taking some actions. The **agent** moves through different states by taking actions generated by **policies**. This give various utility values which are updated over time as states change. Utility is calculated using some evaluation function given either by Q Learning or SARSA:

1. Q Learning: $Q(a,s) \leftarrow (1- \alpha)*Q(a,s) + \alpha *[R(s',a,s)+ \gamma*\max_{a'}Q(a',s')]$
2. SARSA: $Q(a,s) \leftarrow Q(a,s) + \alpha [R(s) + \gamma*Q(a',s') - Q(a,s)]$

where

- $Q(a,s)$ is the current utility if at state s we take action a, initially zero.
- $Q(a',s')$ is the current utility if at state s' we take action a'.
- α is the learning rate.
- s is the current state being evaluated.
- R is the immediate reward function for reaching state s' by taking action a in state s.
- γ is the discount rate.
- $\max_{a'}Q$ is the max utility obtainable from all valid actions after taking action a in state.

Q-Learning being a model-free reinforcement learning technique. An agent learns an action-value function to get an expected utility of taking an action in a given state on the basis of state selection policy.

Here we also use learning rate α which determines to what extent the newly acquired information will override the old information whereas discount rate γ determines the importance of future rewards. A factor of 0 will make the agent "myopic" (or short-sighted) by only considering current rewards, while a factor approaching 1 will make it strive for a long-term high reward.

In this project, I implemented a Q-learning and SARSA algorithm for the PD-World for 3 experiments using γ=0.5 and α=0.3 as given. I have used 2 performance variables a. Bank account of the agent also known as total rewards and b. Number of operators applied to reach a terminal state from the initial *state* which I have denoted as Operators Applied. Moreover every time an agent reaches terminal state I say iteration is complete and increment it.

Experiment 1

Run the Q-learning algorithm for 3000 steps with policy PRANDOM; then run PGreedy for 3000 steps. Display and interpret the Q-Table you obtained in the middle and the end of the experiment.

Performance Variables

Performance Variables for Seed 1

Current Pass	Iteration	Operators Applied	Total Rewards
734	1	734	-318
1890	2	1156	-740
2522	3	632	-216
3110	4	588	-172
3442	5	332	84
3722	6	280	136
3980	7	258	158
4352	8	372	44
4682	9	330	86
4870	10	188	228
5334	11	464	-48
5660	12	326	90

Performance Variables for Seed 2

Current Pass	Iteration	Operators Applied	Total Rewards
776	1	776	-360
1526	2	750	-334
2364	3	838	-422
3106	4	742	-326
3330	5	224	192
3694	6	364	52
4034	7	340	76
4386	8	352	64
4880	9	494	-78
5268	10	388	28
5746	11	478	-62

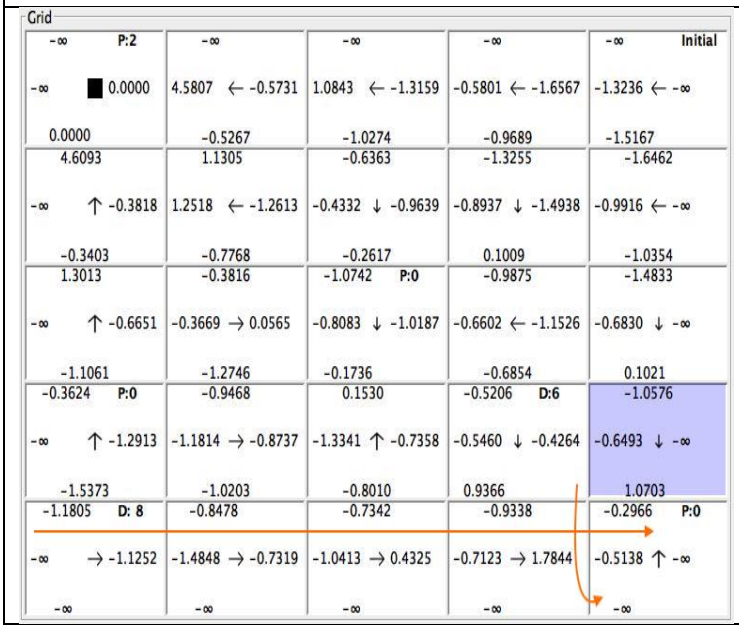
As we run the first 3000 iterations using prandom policy the agent explores most of the actions for each state quite well but the number of iterations completed during this time is considerably less. As you can see above only 3 iterations are completed (With bad rewards and large no. of operators applied) when the random policy is used. Never the less this turns out to be a good thing as we Maximize long-term wellbeing. I am saying this because for seed 1 iterations 5 6 7 we can observe the rewards increasing and operators applied decreasing.

Though this happens in pgreedy policy but one should note that this is due to Q values developed by maximum exploration. We observe that the agent finds the terminal state quite fast and with good rewards in the rest of the duration initially but later we can observe that in between sometimes for the pgreedy policy the agent is not doing well this certainly points out that we need to balance between exploration and exploitation as exploitation looks for maximum rewards and doesn't care about long term.

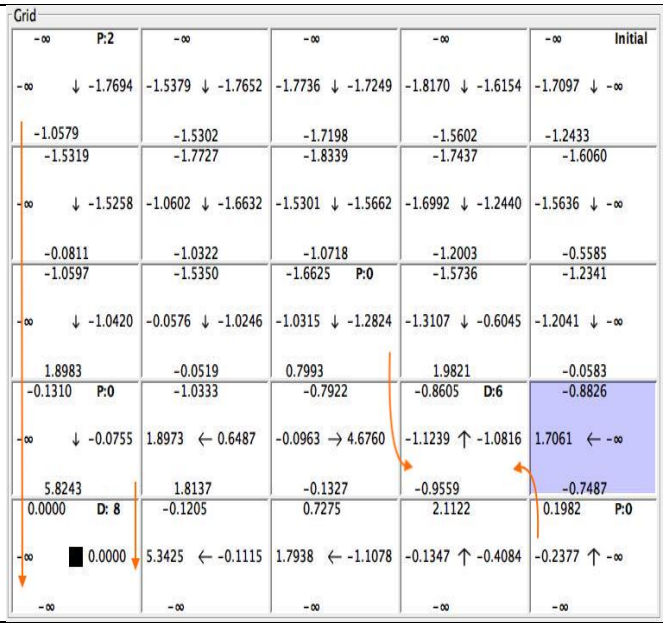
Q table after 3000 Iterations Seed 2

After 3000 iterations when the agent is looking for pickup locations we can observe the attractive paths the agent tries to go to the pickup spots. Similarly, when its looking for drop off locations the diagram on the right shows the attractive paths from pickups to drop offs.

X=0

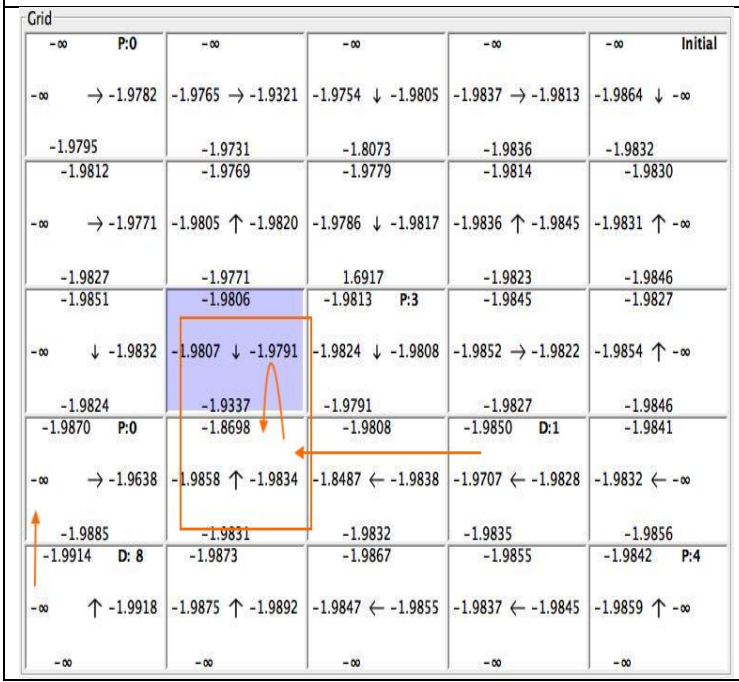


X=1

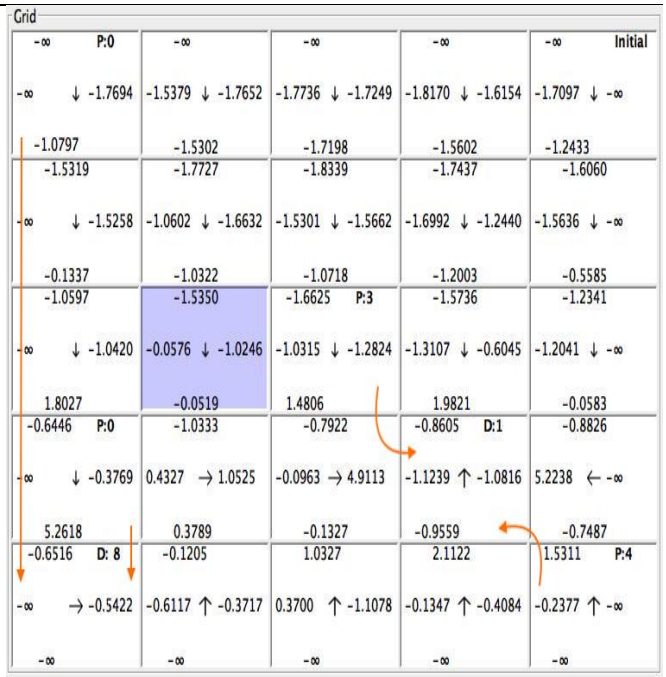


Q table after 6000 Iterations at the end of experiment Seed 2

X=0



X=1



At the end of experiment, I can observe that q values don't seem to give the best results in case the agent wants to pick up. They are quite close to each other hence in long run they may get in the loop and the performance may degrade over time as the agent wouldn't take the best paths later. As of the 6000 iteration the Q values when the agent wants to drop off looks fine as the agent tends to take the optimal paths to drop the package.

Experiment 2

Run the Q-learning algorithm for 6000 steps with policy PEXPLOIT—however, use policy PRANDOM for the first 200 steps of the experiment, and then switch to PEXPLOIT for the remainder of the experiment. Analyze the performance variables and summarize what was learnt by analyzing the q-table at different stages of the experiment.

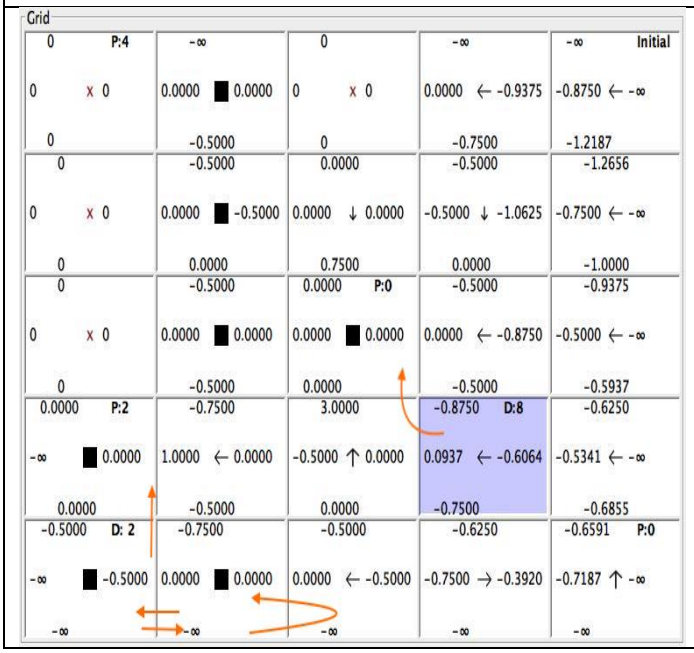
The experiment is not exploring to good extent hence as you can observe qvalues get worse over time

Q-Table Analysis
From the tables below we can clearly see how the pexploit policy converges to the best actions over time. The tables involve observation at 3 points in experiment
Point A: Q Table when first drop-off location is filled (the eighth block has been delivered to it)
Point B: When a terminal state is reached- I have taken when the terminal state is reached first time.
Point C: Final Q Table of the experiment
Analysis when the agent is looking for the drop off location
Here the main idea of agent is find the closest drop off location and in case the closest one is full it should move to next drop off location.
At Point A apart from Pickup location 1,1 the agent takes drop packages from pickup locations to closest drop offs but apparently it is not able move from one drop off to another. But being so early it has hardly learnt anything.
At Point B we see agent can carry blocks from pickups to the closest drop off
At Point C we can clearly see that the agent tries to balance between reaching the drop off from pick up as well as moving from one drop off to another.
But apparently the agents doesn't perform exceptionally well as it quite clear in this experiment that the agent has not explored the environment. It could do better had it explored more initially.

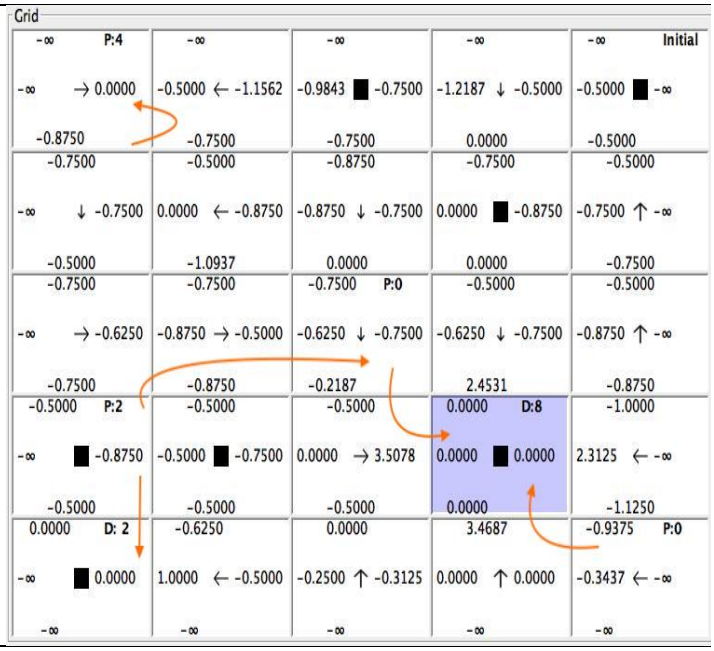
Analysis when the agent is looking for the pickup location
Here the main idea of agent is find the closes pickup location and in case the closest one is empty it should move to next pickup location.
At Point A the agent has not explored the environment as you can clearly see that 1 pickup location is not visited
At Point B though not optimally but the agent has learned to move towards all close pickup locations from the drop off locations but again as this is the first time the agent has reached terminal state it has not learnt much.
At Point C we see that the Q values are not showing that great path. This clearly is a result of over exploitation. There are some paths where the agent moves optimally to pickup location but in general the agent is not giving great output.

Q Table when first drop-off location is filled (the eighth block has been delivered to it) –Seed 1

X=0

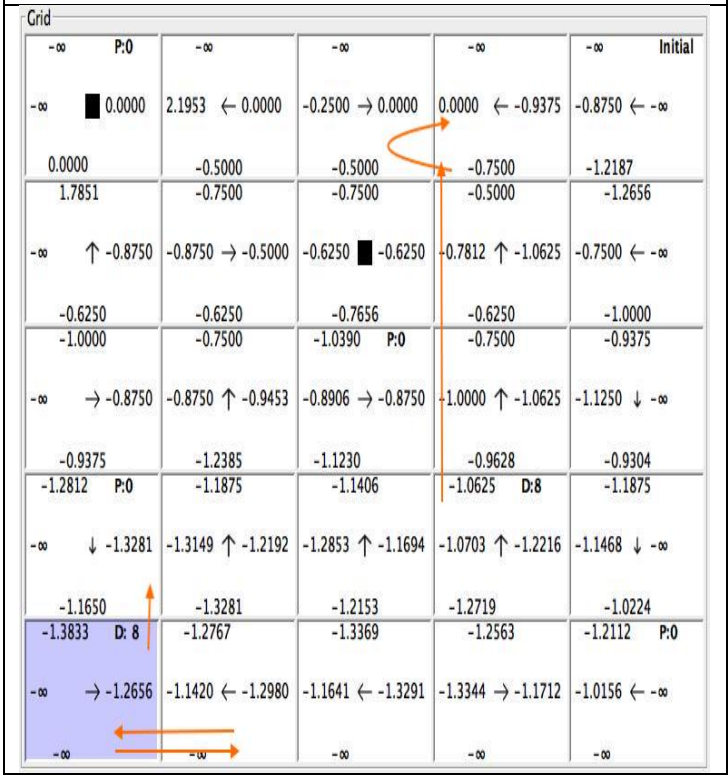


X=1

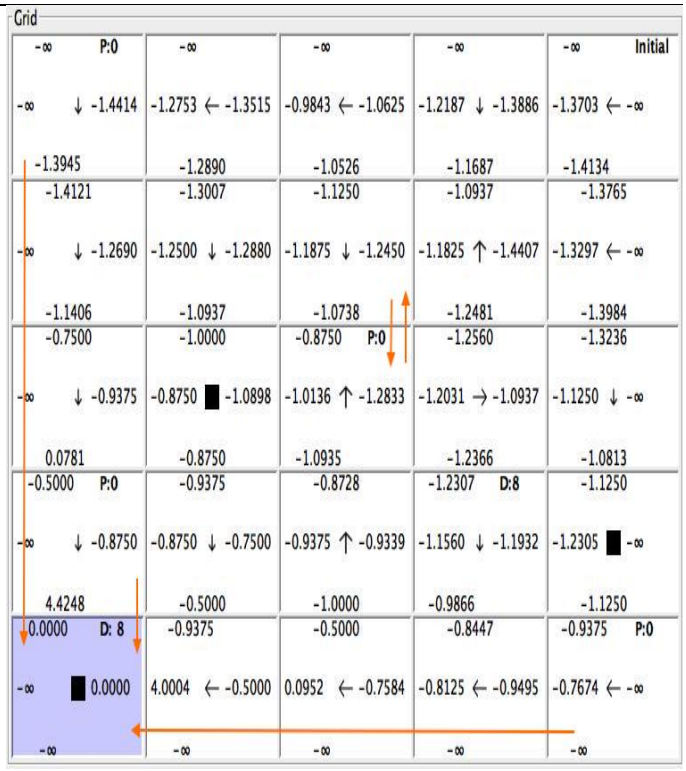


When a terminal state is reached –Seed 1

X=0

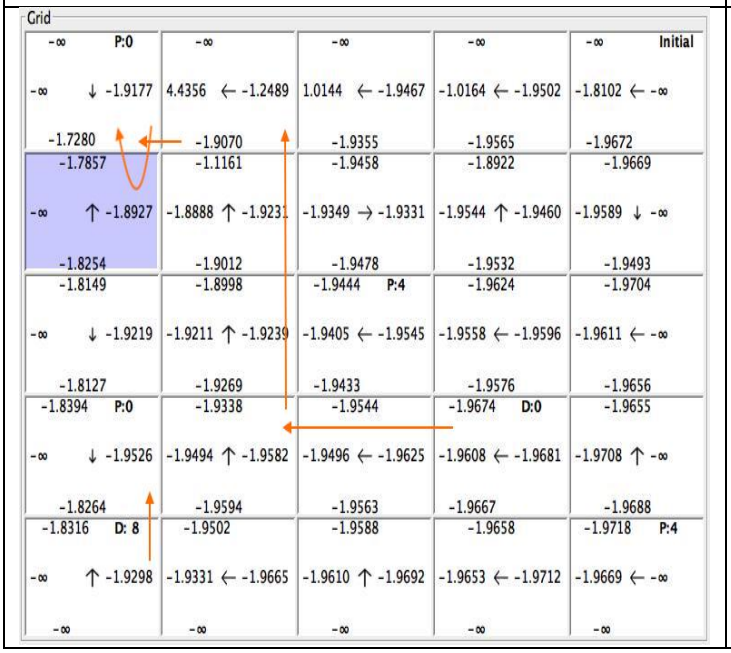


X=1

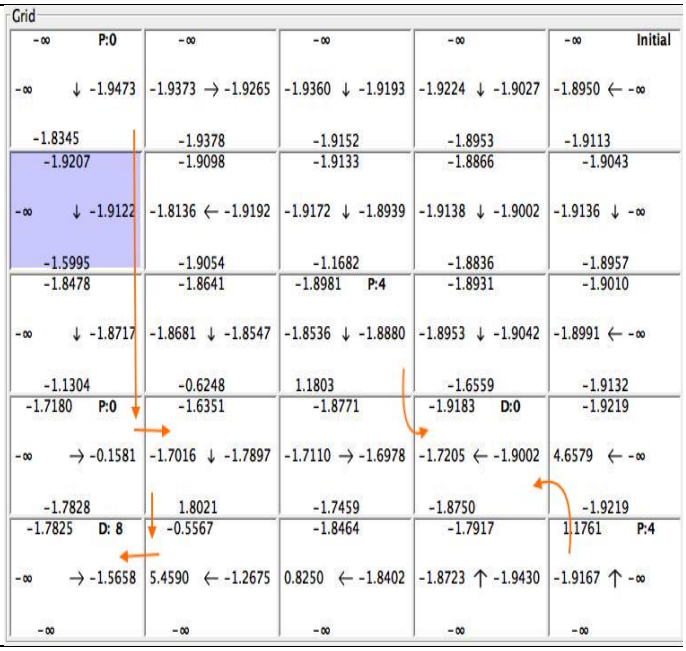


Final Q Table of the experiment –Seed 1

X=0



X=1



Performance Variables

Performance Variables for Seed 1

Current Pass	Iteration	Operators Applied	Total Rewards
548	1	548	-132
944	2	396	20
1520	3	576	-160
1912	4	392	24
2440	5	528	-112
2764	6	324	92
3036	7	272	144
3412	8	376	40
3982	9	570	-154
4436	10	454	-38
4990	11	554	-138
5394	12	404	12
5860	13	466	-50

Performance Variables for Seed 2

Current Pass	Iteration	Operators Applied	Total Rewards
592	1	592	-176
948	2	356	60
1426	3	478	-62
1920	4	494	-78
2566	5	646	-230
3120	6	554	-138
3670	7	550	-134
4186	8	516	-100
4824	9	638	-222
5830	10	1006	-590

Experiment 2 just explores for 200 iterations hardly does the agent explore and then we use pexploit policy which explores just with probability of 0.15. From the above 2 performances it is quite evident that the agent performs badly over time.

For seed 1 we can see that the agent does better sometimes and bad other times. Here the agent is not giving good results or bad results continuously like in the case of seed 2 the agent gives all

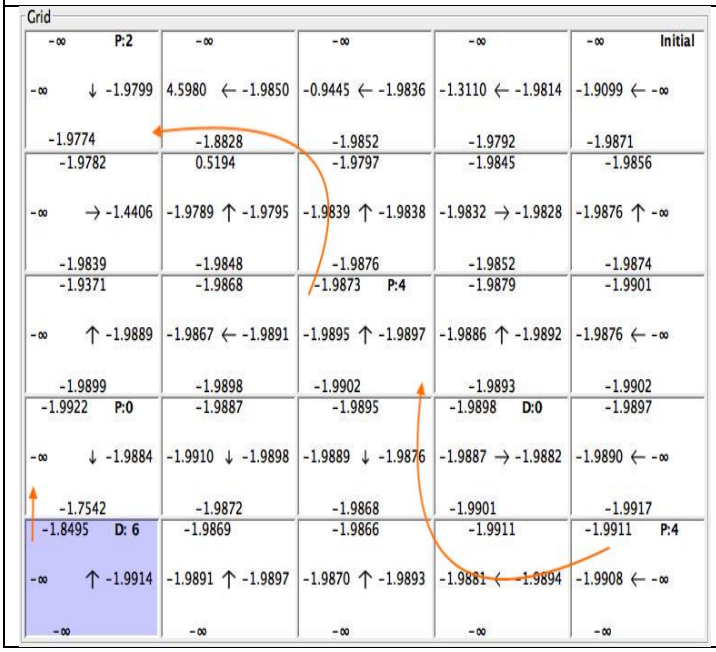
negative rewards with increasing operators applied value. Clearly summarizing that the agent is over exploiting.

Experiment 3

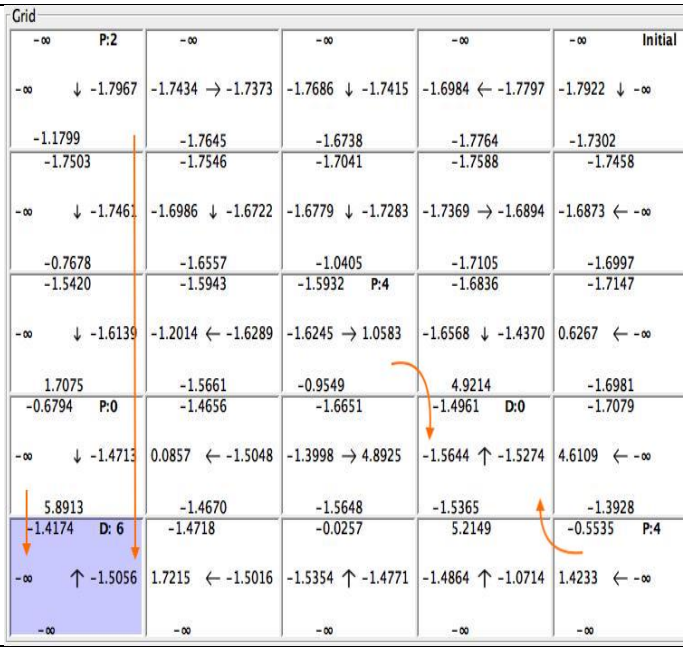
Run the SARSA q-learning variation for 6000 steps with policy PEXPLOIT—however, use policy PRANDOM for the first 200 steps of the experiment, and then switch to PEXPLOIT for the remainder of the experiment. When analyzing Experiment 3 center on comparing the performance of Q-learning and SARSA. Also report the final q-table of this experiment.

Final Q Table of the experiment-Seed 2

X=0



X=1



At the end of experiment, I can observe that q values don't seem to give the best results in case the agent wants to pick up. Among the 2 criteria discussed above the agent can very well move from one pick up to another but it cannot move from drop off to pick up. It only moves from drop off 5,1 to pick up 4,1.

On the other hand, the agent does exceptionally well incase its looking for drop off as you can see above. Here all the paths match with the best path possible.

Performance Variables

Performance Variables for Seed 1

Current Pass	Iteration	Operators Applied	Total Rewards
550	1	550	-134
912	2	361	54
1310	3	398	18
1600	4	290	126
1924	5	324	92
2340	6	416	0
2728	7	388	28
3296	8	568	-152
3686	9	390	26
4256	10	570	-154
4680	11	424	-8
5562	12	882	-466

Performance Variables for Seed 2

Current Pass	Iteration	Operators Applied	Total Rewards
532	1	532	-116
820	2	288	128
1194	3	374	42
1654	4	460	-44
2012	5	358	58
2312	6	300	116
2624	7	312	104
2878	8	254	162
3260	9	382	34
3760	10	500	-84
4242	11	482	-66
4652	12	410	6
5014	13	362	54
5454	14	440	-24
5934	15	480	-64

As we can clearly see that the number of iterations completed is better in SARSA compared to other experiments. Here the rewards and operators applied are quite stable and we don't get bad results like Experiment 2. We are using PExploit policy and PRandom for 200 passes. Had the agent moved in random passes and explored more probably it would had done better.

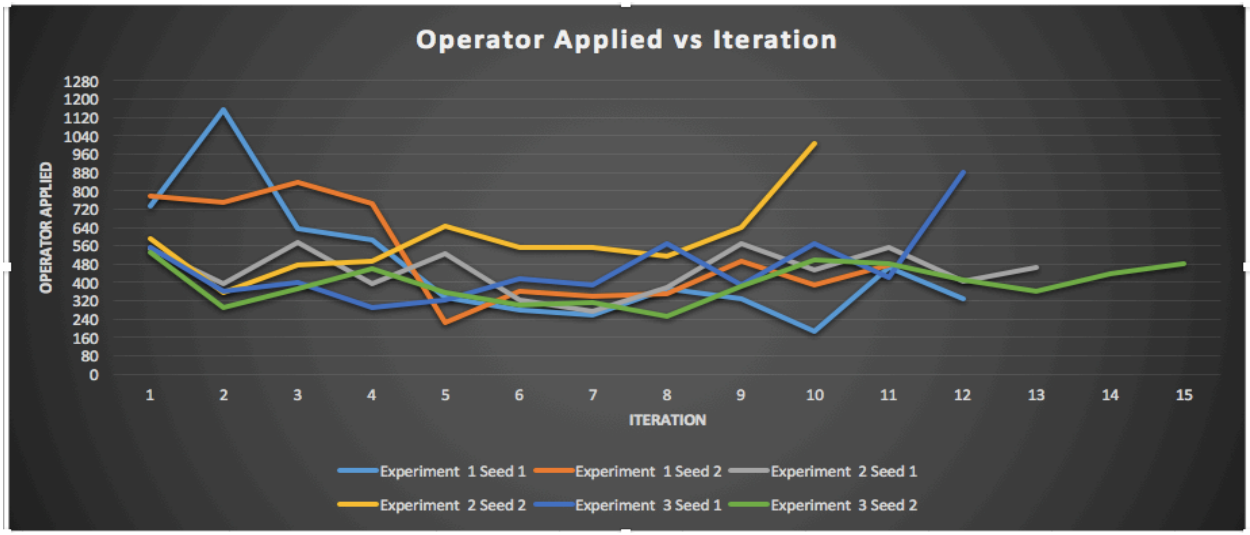
Comparison between Experiments 2 and 3

Experiment 2 implements Q-Learning whereas Experiment 3 we use SARSA Q-Learning. The main difference between the two is that in Q- Learning we update the Q value on the basis of greediness as we take $\max_{a'}Q$ on the other hand in SARSA we update the Q value on the basis of Policy.

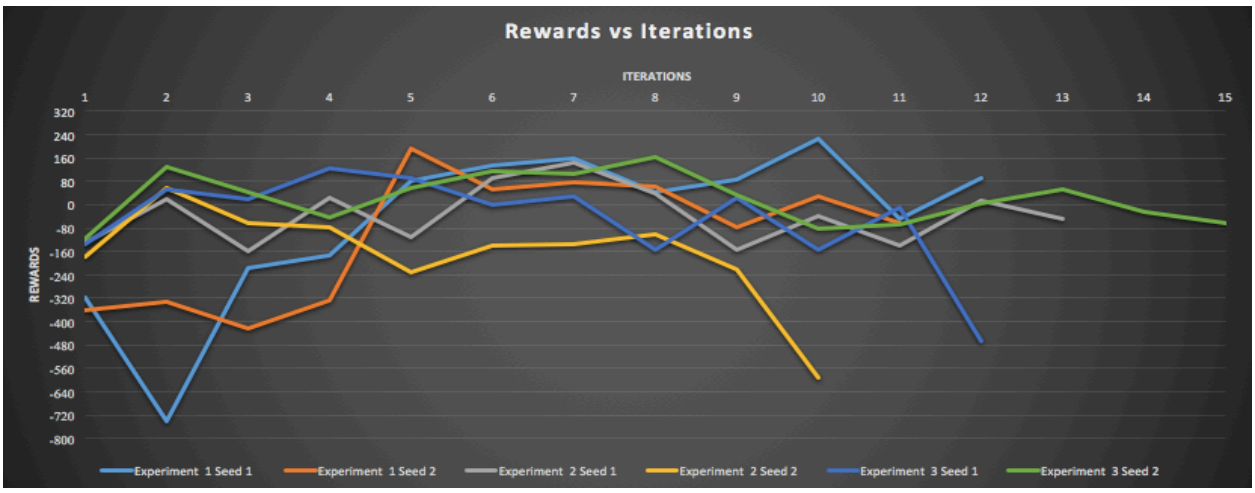
Between the two Experiment 3 does a better job as we can clearly see that the performance measures explained above i.e. the number of operators applied on average are less in Experiment 3 and the reward on average is greater in case of Experiment 3. Below are the graphs.

Moreover, the final actions with respect to Q values are significantly better in Experiment 3 when the agent tries to drop off. In case when the agent tries to pick up the actions in both the experiments seem to be ambiguous. But at least in experiment 3 the agent can move from some pickups to another in case of they are empty.

Conclusion



As per the performance measure operator applied, among the 3 Experiments, Experiment 1 gave the best output as we can observe above that there is significant drop in the operators applied overtime with iterations compared to Experiment 2 and 3. Moreover the average performance measure operator applied for experiment 1 for later iterations seems to be greater than that of experiment 2 and 3



Similarly, the average of performance measure rewards for Experiment 1 for later iterations seem to have higher than Experiment 2 and 3 as we can see above.

Experiment 1 runs the random policy for 3000 iterations thus it has explored the environment to great extent. Hence it has comparatively better performance and the trade off between exploration and exploitations seems to be balanced in experiment 1.