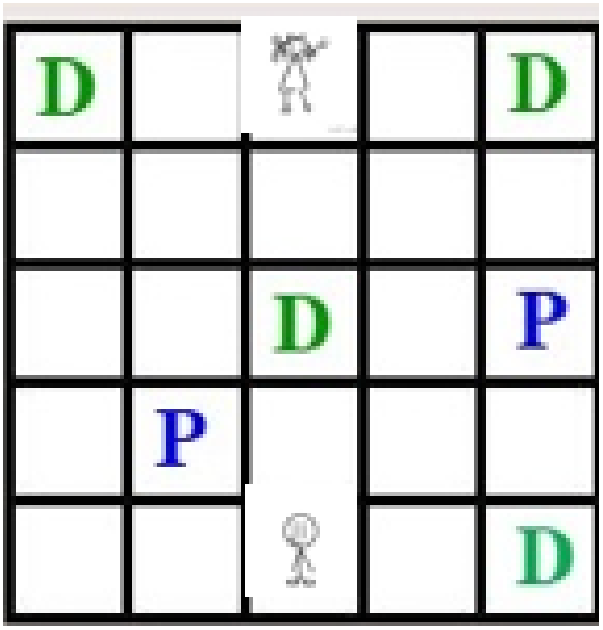
Using Reinforcement Learning To Discover Paths in a 2-Agent Transportation World

By

Tailer Nguyen, Christian Montemayor, Gilbert Lopez,

Kyle Stulen, Joshua Mang



Abstraction

To make good decisions in a PD world, agents must use reinforcement learning to generalize their new experiences, make use of feedback from reinforcement learning, and they must make use of their resources given to them, and ultimately make good decisions through action and observation. The actual reinforcement learning approach uses a simplified state space that aggregates multiple states of the actual state space into a single state in the reinforcement learning state space. These agents can make predictive and quantitative models different to each environment that are shown in the reinforcement learning. In this paper, we can observe multiple dissertations of how our two agent processes make their decisions through different policies.

Vocabulary

- Agent: It is an assumed entity that performs actions in an environment to gain some reward.
- Environment (e): A scenario that an agent has to face.
- Reward (R): An immediate return given to an agent when he or she performs specific action or task.
- State (s): State refers to the current situation returned by the environment.
- Policy (π): It is a strategy that is applied by the agent to decide the next action based on the current state.
- Value (V): It is expected long-term return with discount, as compared to the short-term reward.
- Value Function: It specifies the value of a state which is the total amount of reward. It is an agent which should be expected beginning from that state.

- Model of the environment: This mimics the behavior of the environment. It helps you to make inferences to be made and also determine how the environment will behave.
- Model-based methods: It is a method for solving reinforcement learning problems that use model-based methods.
- Q value or action value (Q): Q value is quite similar to value. The only difference between the two is that it takes an additional parameter as a current action.
    - Female Has Block Heatmap: the female agent's Q table when she is holding a block
    - Female No Block Heatmap: the female agent's Q table when her hands are empty
    - Male Has Block Heatmap: the male agent's Q table when he is holding a block
    - Male No Block Heatmap: the male agent's Q table when his hands are empty

Problem Definition and Algorithm

In the project, we were asked to implement a solution to the PD-World problem with the use of Q-Learning and SARSA. In the PD-World problem, we are given a board as seen below,

	1	2	3	4	5
1	D		F		D
2					
3			D		P
4		P			
5			M		D

Figure 1: PD World with pickup and dropoff states. Number of items picked up not shown.

where the ‘F’ cell at (1, 3) and ‘M’ cell at (5, 3) signify female and male agents, respectively. The goal of PD-World is to transport blocks from the pickup cells (3, 5) and (4, 2) to the drop-off cells at (1, 1), (1, 5), (3, 3), and (5, 5). To add to the constraints of the problem, each pick-up cell initially contains 10 blocks, while each drop-off cell can only hold 5 blocks. In our project, our agents have six operators, north, south, east, and west as the cardinal directions of movement. Additionally, the agent can also pick up a block if it lands in the pickup square to have a max capacity of 10. Also, the agent can drop off the blocks, only if the drop-off square has less than four if the agent is on top of the drop-off square. We use reinforcement learning strategies Q-Learning and policy-based reinforcement learning through SARSA to train the agents and develop a strategy for solving the problem of PD-World.

In State-Action-Reward-State-Action, a variation of the Q-learning algorithm, the agents will continue to adhere to the same policy throughout the process, whereas in Q-learning, there are greedy decisions being made along the entire process, thus giving SARSA the "on-policy" name, and making SARSA a deterministic process.

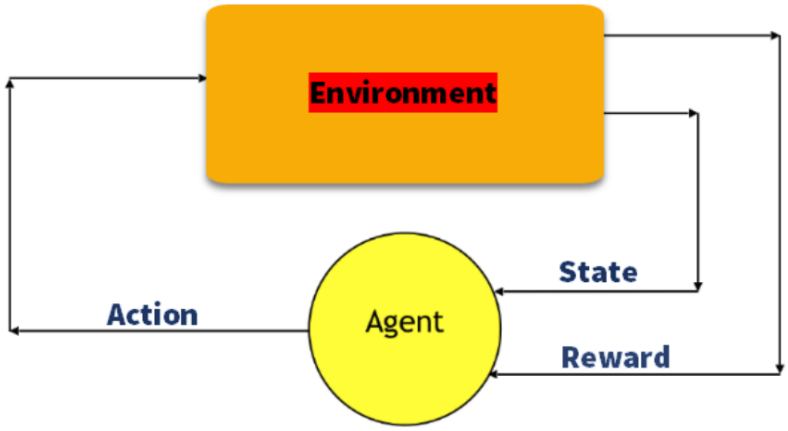


Figure 2: A visualization of the SARSA state to action to reward diagram.

Our original state space includes (**i, j, x, a, b, c, d, e, f)** where i and j are the agents initial positions, x is 0 or 1 if the agent has a block or not, and (a,b,c,d,e,f) are the number of blocks in the following cells: (1,1), (1,5), (3,3), (3,5), (4,2), (5,5).

Initial State: (5,1,0,0,0,0,10,10,0)

Terminal State: (*,*,0,5,5,5,0,0,5)

In the algorithm that the program will partake in, we have two agents working against each other in picking up and dropping off blocks. This takes place in a PD world state space where the algorithm initially learns paths between pickup and dropoff states, thus finding different paths for x = 0 or x = 1. For a number of steps, our agent chooses a policy and an action to take, then we update the q_values in the q_table. We first initialize the agent without q_tables all having the

initial values of 0, and then we update based on what actions the agent partakes in. By checking to see if the agent is holding a block, if the drop location is full then we cannot consider those Q values from being considered. Similarly, if the agent is not holding a block and the pickup location is full, then we must delete the Q values from being considered. The reward for choosing a cardinal direction is -1, and the reward for picking up or dropping off the block(s) is +13, thus incentivizing the agent (depending on which policy they partake in) to take actions that will further increase their q-values by the terminal state.

We then include a heatmap of the q-values to check the agents' future likelihood of moving around the PD world and the rewards given to them through their actions.

Experimental Evaluation and Methodology

Experiment 1: POLICIES

Three policies are defined and applied to our program and each policy has a different constraint when delivering one block from pick up to drop off location. Each policy leads the agent to complete the task differently while the heatmap results also vary to show the different paths.

PRANDOM POLICY

For PRANDOM policy, the two agents check if there is a pick-up or drop-off location in the next state before making the move. If pick up or drop off is applicable, the agent chooses that operator first. However, the agent will choose the next state randomly if the pick-up and drop-off location are not applicable in the next state.

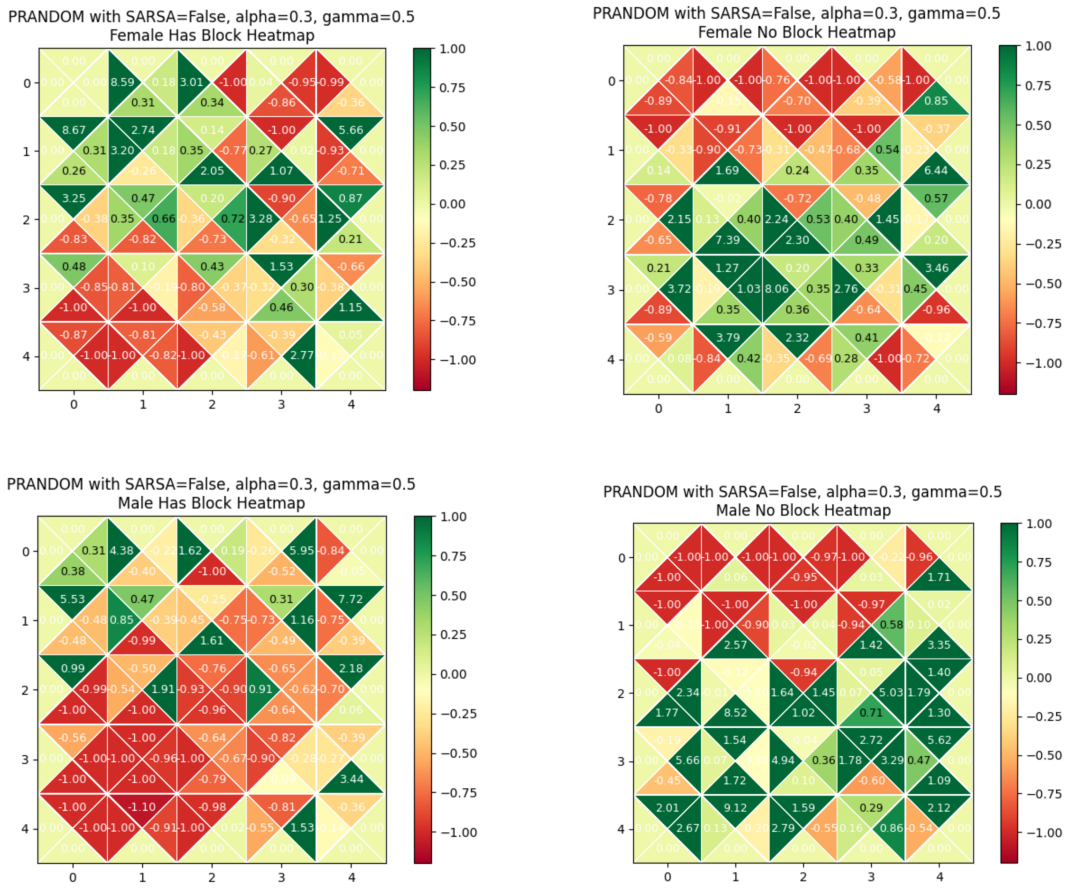


Figure 3: A generation of PRandom heat-maps for when the female and male agent has a block versus when they do not have a block.

We can see from Figure 3 that each heatmap has a different "hot section" where each agent moves around in conjunction with the other agent. Because the heat map is random, we see a huge conglomeration of hot sections all over each heat map, but the triangle sections where the agents could possibly move do not look very uniform for what best action to take next for each agent.

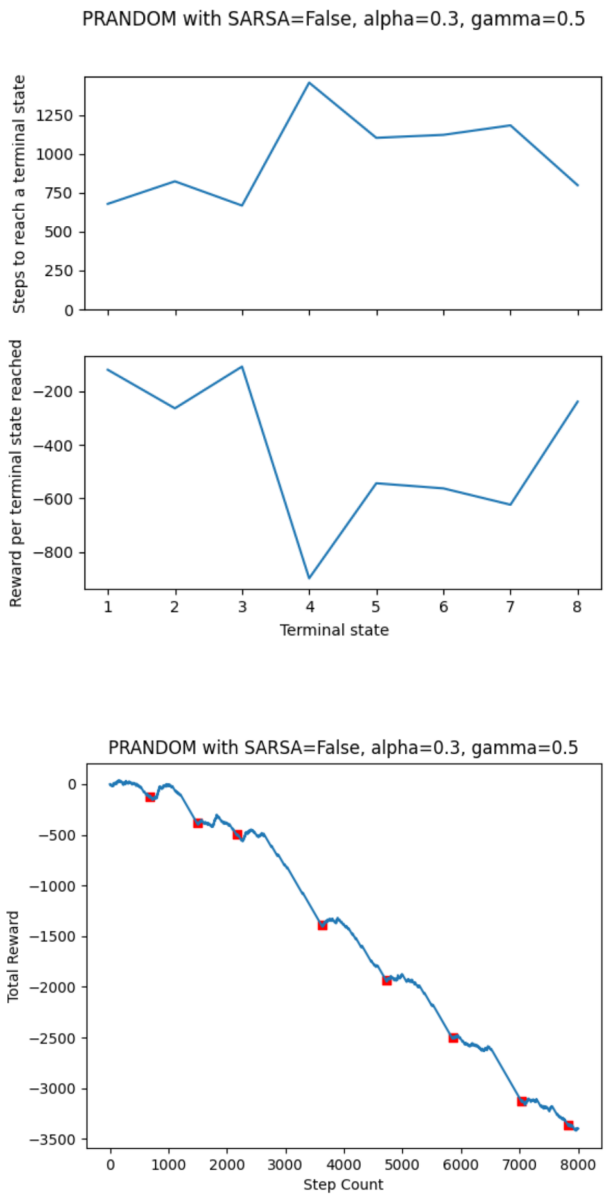


Figure 4: The steps per terminal space and the terminal log reward below for PRandom.

PGREEDY

In PGREEDY policy, the agents choose the pick-up and drop-off location first if which is applicable. If pick-up or drop-off is not applicable, the agents will always choose the highest Q

value from available directions and choose that stage. The agent will randomly choose the next stage if the available directions have the same Q value.

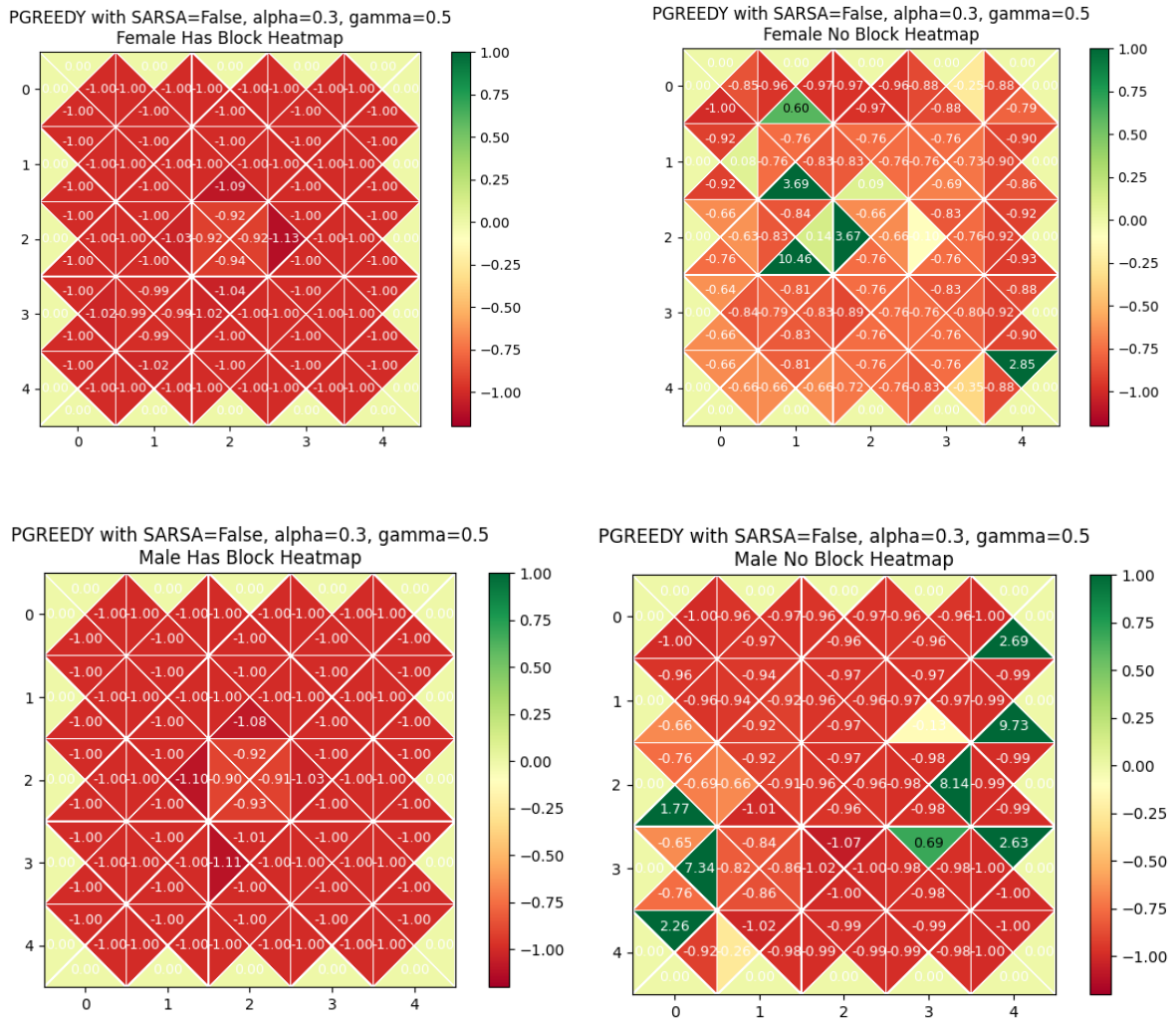


Figure 5: The heatmaps given from the PGreedy policies.

We can see from Figure 5 that each heatmap has a different "hot section" where each agent moves around in conjunction with the other agent. Because the agents always take the maximal path, we can see that the male with no block has the highest Q-values for the pick-up and drop-off locations on the bottom half of the map. This is likely due to the male's proximity to both pick-up locations. Thus, it is not optimal for the male to go to the top half of the map and since the policy is PGREEDY, it rarely goes up there. However, the female with no block has

high Q-values all across the map. This is likely because it has to do a lot of exploration to find pickup locations since they are comparatively distant.

Although the heatmaps for the agents holding no block are very different, their heatmaps when holding a block are very similar. This makes sense because drop-off locations are very close to the pickup locations and these Q-values are being used when the agents arrive at a pick-up location and perform the pick-up action.

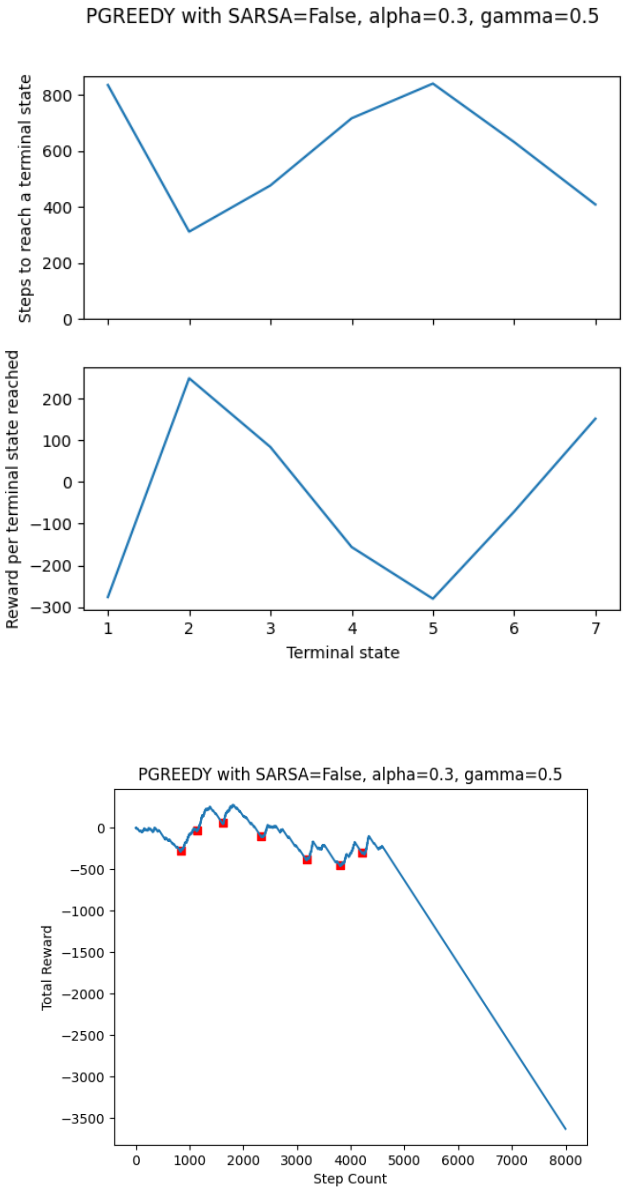


Figure 6: The steps per terminal space and the terminal log reward below for PGreedy.

PEXPLOIT

In PEXPLOIT policy, if pick-up or drop-off is applicable, the agents choose this operator first and move to that stage. Otherwise, the agent will pick the highest Q value 80% of the time from available directions and random stage if the available direction of Q value from available 20%.

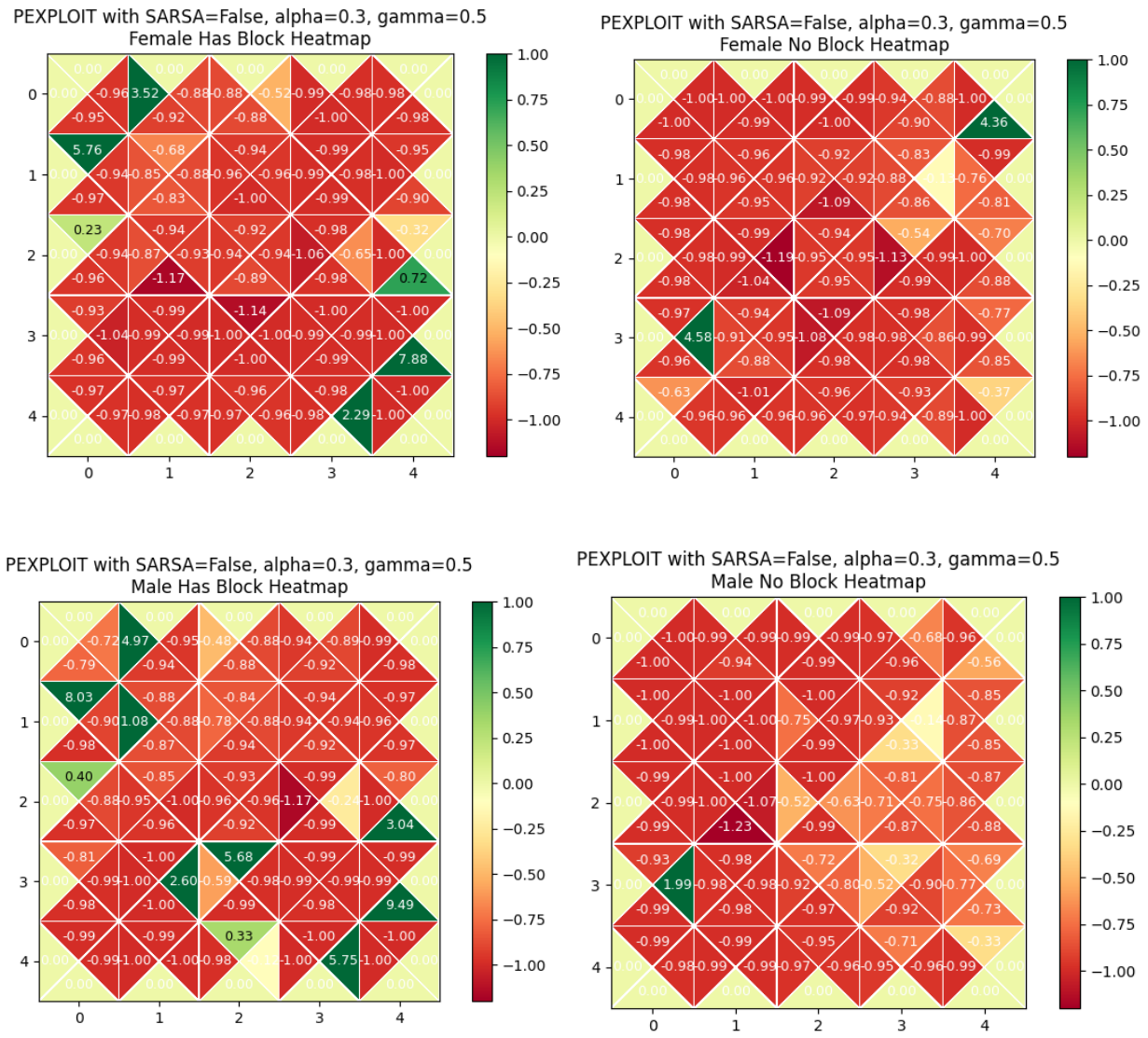
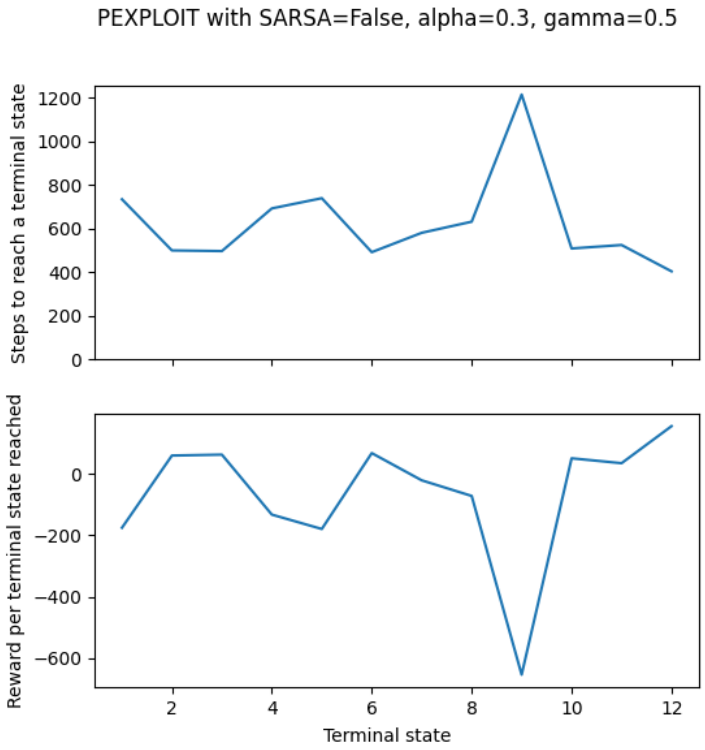


Figure 7: Heatmaps for the policies PExploit.

We can see from figure 7 that the Q-values when the agents do not have a block are higher near the pick-up locations on the right and near the bottom and are overall very similar.

Probably due to the randomness of paths taken and the fact that most of the pick-up locations are on the lower right side of the world. The male has higher Q-values on average though since it is closer to these locations.

The heatmap for when the agents do have a block is similar as well, but oddly enough the male has higher Q-values on average near the drop-off locations. We expected the opposite to be true because the female tends to explore more due to its location, but we'll chalk this up to randomness since these Q-values are mostly dependent on what happens after the agents pick up which all happens in the same general area of the world.



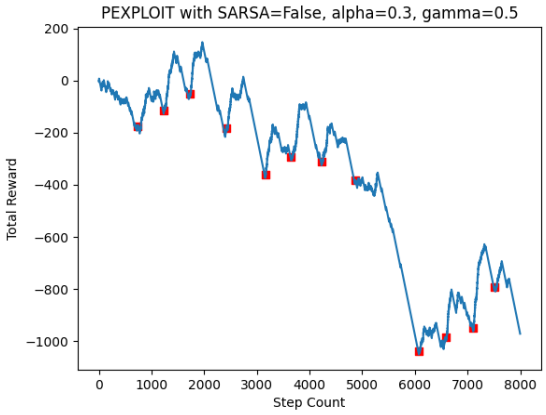
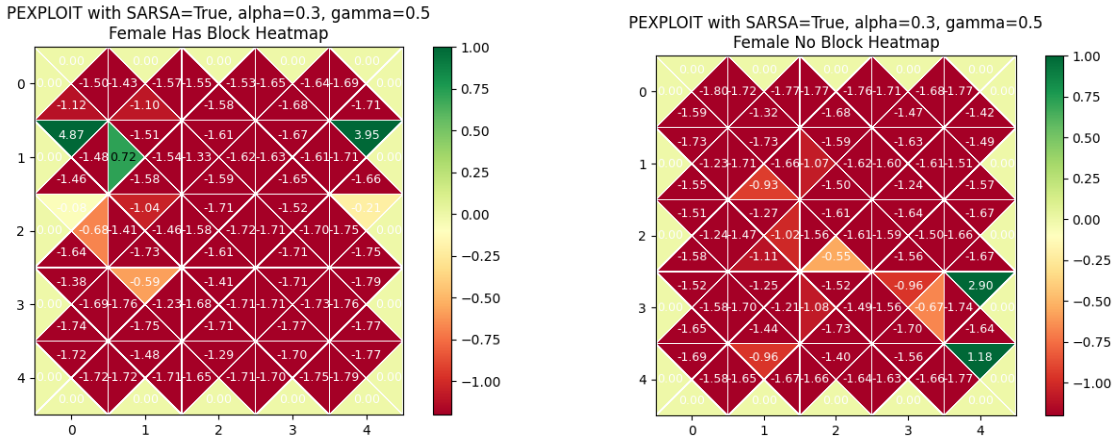
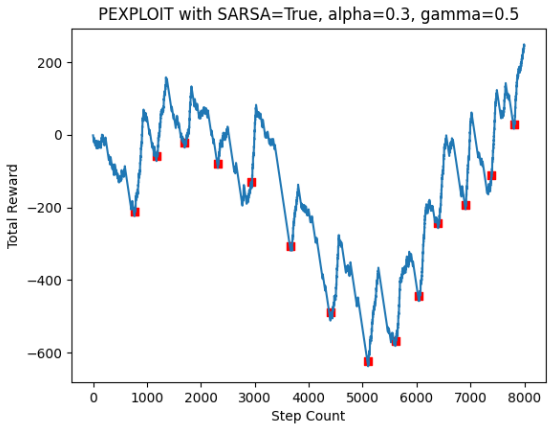


Figure 6: The steps per terminal space and the terminal log reward below for PExploit.

Experiment 2

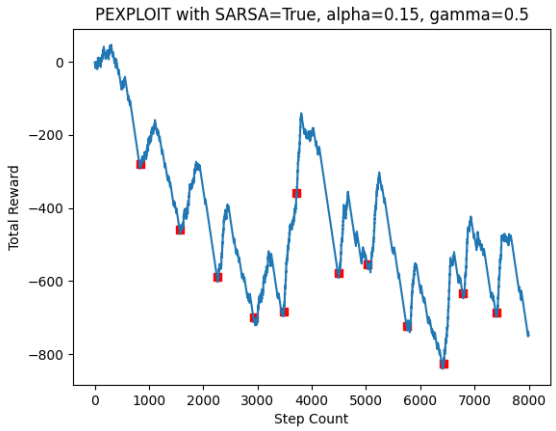
For experiment 2 we were tasked to utilize the PEXPLOIT policy once again, but instead of using the Q-Learning algorithm to update the Q-tables we would use the SARSA algorithm instead. Both algorithms had similar performance.

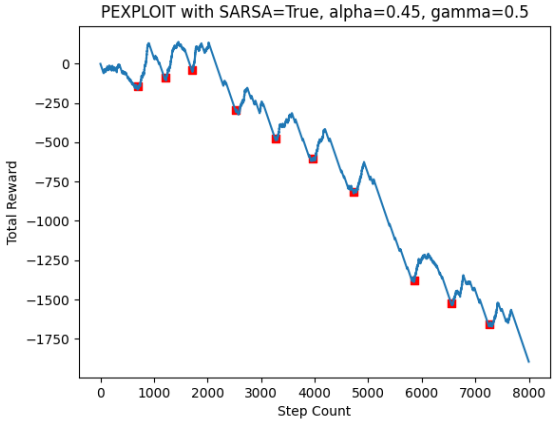




Experiment 3

For experiment 3 we were tasked with testing different learning rates for PEXPLOIT utilizing SARSA. Experiments 1 and 2 used a learning value of 0.3 and we will test 0.15 and 0.45 as well now. A learning rate of 0.15 performed more precisely than 0.3; resulting in a narrower spread of reward values across multiple runs of the program, but did not find more terminal states. A learning rate of 0.45 performed worse.





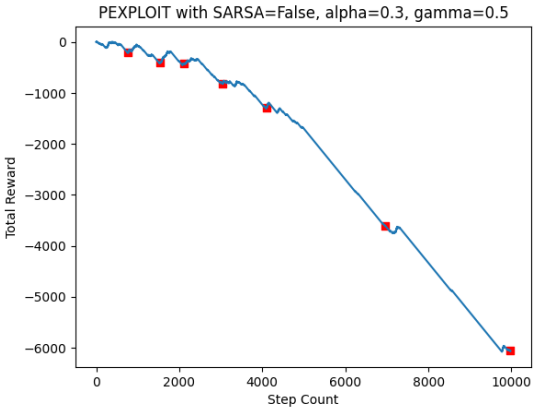
Problems agents faced in Experiments 1, 2, and 3

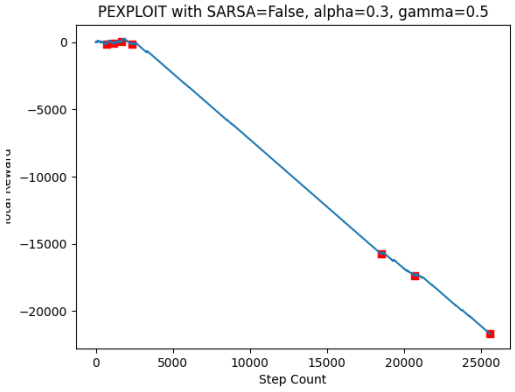
What held our agents back the most was that while they were able to quickly move the majority of the blocks from pickup to drop-off locations, they struggled to place the last blocks every time. This is shown by positive spikes in the Step Count vs Total Reward graphs followed by

long periods of only receiving negative rewards from moving. This occurred because agents would pick up a block and then travel to a drop location that had already reached maximum capacity and would have to spend time unlearning that location was a good drop location. The next improvement to these agents' behavior would be adding a boolean value to the state space for each pickup or dropoff location. The boolean values would be 1 if a pickup location still had blocks available or if the drop-off location had yet to reach capacity.

Experiment 4

The goal of experiment 4 is to evaluate how the model will perform if the pickup locations change mid-execution. We changed the pickup locations after the termination state was reached four times and printed the results of the model after the termination state was reached seven times. The model performed extremely poorly in response to the pickup location being changed meaning this learning strategy is ineffective at unlearning old obsolete paths.





Results and Performance

Policy	Run	Terminal States	Reward
PRANDOM	1	8	-3282
	2	8	-3044
PGREEDY	1	8	-3072
	2	7	-3632
PEXPLOIT	1	14	386
	2	12	-972
SARSA a = 0.3	1	12	-748
	2	14	246
SARSA a = 0.15	1	13	-356
	2	13	-174
SARSA a = 0.45	1	10	-1896
	2	12	-1000
Change pickup location midrun	1	7	-6059
	2	7	-21687

We can see from the results that PExploit and the SARSA on-policy runs had the highest performing agents who did not collide with one another and were able to find the best paths to pick up and drop off blocks. Although all on and off policies were able to find paths, PRandom, PGreedy, change start exploit, and a SARSA alpha value greater than 0.45 all had too many steps and way more negative rewards than the others.

From the heat maps shown too, the agents, when forced to avoid each other, only avoided each other in PExploit and PExploit with SARSA. In PGreedy and PRandom, the agents collided with each other differently in their runs to negative effect. However, in PExploit and PExploit with SARSA, the agents' avoidance of each other led to higher rewards for the agents.

With the rewards per steps graphs, PExploit with SARSA also ended up being the only net positive after the total amounts of steps. PRandom and PExploit with Change pickup location mid-run ended up being nearly flat-line no awards have been given to agents and no bouncing back on the graph has occurred, so this makes it seem as if the agents were aimlessly wandering (and they were according to the definition of randomness). However, changing the pick-up location midrun was very interesting because the agents had to adapt a new reinforcement learning style.

With PExploit and PExploit with SARSA, there was at least some life within the graph which shows that the agent is still optimizing each move that they take based on their past experiences, too.

Conclusion

In conclusion, Q-learning is an off-policy reinforcement learning algorithm that finds the best action to take from a given current state. It's considered off-policy because the q-learning function learns from actions that are outside the current policy, like taking random actions. On the other hand, SARSA is an on-policy reinforcement learning algorithm where the agent chooses the action following the same current policy and updates its Q-value. In spite of getting

quite the limited feedback, the reinforcement learning approach and SARSA approach is indeed a decent strategy for transferring blocks from their pickup spots to the drop-off locations. The hyper-parameters that were the best leading performers were of an alpha learning rate of 0.15 and of a gamma discount rate of 0.3. We found that changing the learning rate from 0.3 to 0.15 resulted in much more results across multiple runs of the program. The reinforcement learning approach had different policies that led to more efficient paths from block source to block destination. The policies that were good are: PEXPLOIT, SARSA (a = 0.3), and SARSA (a = 0.15). Based on the heat maps and coordinating collision detection, we can see that the agents were able to quickly learn how the other agent moves and not cross paths. Although PEXPLOIT policy is the best model in this project, after changing the pickup location in experiment 4, the model performed extremely poorly and this learning strategy is ineffective at unlearning old obsolete paths. Overall, this project gives us a deeper understanding of Q Learning and SARSA (State Action Reward State Action) algorithm and identifies which algorithm will perform better in these situations.

Bibliography

https://cs.stanford.edu/people/karpathy/reinforcejs/index.html

https://medium.com/swlh/introduction-to-reinforcement-learning-coding-sarsa-part-4-2d64d6e37617

http://www2.cs.uh.edu/~ceick/ai/2021-World.pptx

http://www-all.cs.umass.edu/rlr/domains.html

http://courses.cs.washington.edu/courses/cse473/15sp/assignments/project3/project3.html

http://ai.berkeley.edu/project_overview.html

https://github.com/kristofvanmoffaert/Gridworld

http://cs.stanford.edu/people/karpathy/reinforcejs/gridworld_td.html

https://mediatum.ub.tum.de/doc/1238753/1238753.pdf

http://www2.econ.iastate.edu/tesfatsi/RLUsersGuide.ICAC2005.pdf

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.113.7978&rep=rep1&type=pdf

https://medium.com/swlh/introduction-to-reinforcement-learning-coding-sarsa-part-4-2d64d6e37617

https://medium.com/swlh/introduction-to-reinforcement-learning-coding-sarsa-part-4-2d64d6e37617