*Md Mahin and Christoph F. Eick*

COSC 4368 Group Project Spring 2023

*Using Reinforcement Learning*

*To Discover Paths in a 2-Agent 3D Transportation World*

4 (3) Students per Group)

Fourth Draft



Last updated April 4, 2023

Deadlines: Submit project report and source code by Saturday, April 22 end of the day

In this project we will use reinforcement to learn and adapt “promising paths” in 2-agent setting. Learning objectives of the 2023 COSC 4368 Group Project include:

* Understanding basic reinforcement learning concepts such as utilities, policies, learning rates, discount rates and their interactions.
* Obtain experience in designing agent-based systems that explore and learn in initially unknown environment and which are capable to adapt to changes.
* Learning how to conduct experiments that evaluate the performance of reinforcement learning systems and learning to interpret such results.
* Development of visualization techniques summarizing how the agents move, how the world and the q-table changes, and the system performance.
* Development of path visualization and analysis techniques to interpret and evaluate the behavior of agent-based path-learning systems.
* Develop and learn coordination strategies for collaborating agents
* Creating solutions for 3D visualization problems
* Learning to develop AI software in a team.



Figure 1: Visualization of the 3D PD-World



Figure 2: An Urban Grid World.

In particular in this project you will use *Q-learning/SARSA[[1]](#footnote-1) for the 3- dimensional PD-Word assuming a 2 agent setting (*[*http://www2.cs.uh.edu/~ceick/ai/2023-World.pptx*](http://www2.cs.uh.edu/~ceick/ai/2023-World.pptx) *)*, conducting four experiments using different parameters and policies, and summarize and interpret the experimental results. Moreover, you will develop path analysis and visualization techniques that are capable to shed light on what paths the learning system actually has learnt from obtained Q-Tables—we call such paths *attractive paths* in the remainder of this document. Moreover, you will analyze if the two agents collaborated well by avoiding blockage that occurs if the two agents work on the same path. Finally, you will analyze how close your approach came to an Optimal 2 Agent Policy.

Two agent named ‘M’ (male) and ‘F (female) are solving the block transportation problem jointly. Agent alternate applying operators to the 3D PD-World, with the female agent acting first. In this world the agents can move east, west, north, south, up and down. Moreover, both agents cannot be in the same position at the same time; consequently, there is a blockage problem, limiting agent mobility and ultimately efficiency in case that both agents work on the same path at the same time. Additionally, there are two cells that are very risky and they return 2 times more negative reward than normal cells. There are two approaches to choose from to implement 2-agent reinforcement learning.

1. Each agent uses his/her own reinforcement learning strategy and Q-Table. However, we assume that the position the other agent occupies is visible to each agent, and can therefore can be part of the chosen reinforcement learning state space.
2. A single reinforcement learning strategy and Q-Table is used which moves both agents, selecting an operator for each agent and then executing the selected two operators.

Extra credit is given to groups who devise and implement both 2-agent learning approaches with all the constraints and compare their results for experiments 2 and 3 (see below)

In experiments we assume that q values are initialized with 0 at the beginning of the experiment. The following 3 policies will be used in the experiments:

* **PRANDOM**: If pickup and dropoff is applicable, choose this operator; otherwise, choose an applicable operator randomly.
* **PEXPLOIT**: If pickup and dropoff 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 0.85 and choose a different applicable operator randomly with probability 0.15.
* **PGREEDY**: If pickup and dropoff 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).



Figure 3: Visualization of an Attractive Path for a Search Problem

Objectives of the Experimental Evaluation: Besides analyzing the performance of various variations with respect to the bank account/how quickly the transportation problem was solved, the experimental evaluation should also additionally analyze:

1. Agent coordination: Do the two agents get in their ways blocking each other or do they do a good job in dividing the transportation task intelligently between each other. Agent coordination could, for example, be measured by computing the average Manhattan distance between the two agents during the run of a specific experiment. You can show the progress of average distance for different iterations.
2. Paths learned: Does the particular approach do a good job in learning paths between block sources and block destinations; is the learnt path the shortest path or close to the shortest path between the source and the destination.
3. Cost Efficient Learning: Does the particular approach do a good job learning efficient paths or avoiding risky areas. How much are risky areas traversed by the agents as the training goes on.
4. Closeness to the “Optimal Subdivision of Work” Between the Two Agents: How good a job does the particular approach do in Experiments 1.b, 1.c and 2 with respect an “optimal subdivision of work” where one agent transport blocks from the pickup location (2,2,1) to the drop off locations (1,1,2), and (1,1,3) and the other agent transports blocks from the pickup location (3,3,2) to the drop off locations (3,1,1) and (3,2,3). If this strategy is used no blockage problem between the two agents occurs as they work on disjoint paths.

Please conduct the following four experiments:

1. In Experiment 1 you use α=0.3 and γ=0.5, and run the traditional Q-learning algorithm for 10000 steps; initially you run the policy PRANDOM for 500 steps, then
	1. Continue running PRANDOM for 9500 more steps[[2]](#footnote-2) (only policy will change, agents will keep the behavior from the first training)
	2. Run PGREEDY for the remaining 9500 steps (only policy will change, agents will keep the behavior from the first training)
	3. Run PEXPLOIT for the remaining 9500 steps (only policy will change, agents will keep the behavior from the first training)

Summarize and interpret the different results you obtain by running these three strategies. Also report one of the final Q-Table of experiment 1.c. Also assess the quality of the coordination between the two agents for experiments 1b and 1c and how close they get towards the optimal subdivision of work.

1. Experiment 2 is the same as experiment 1.c except you run the SARSA q-learning variation for 9500 steps. When analyzing Experiment 2 center on comparing the performance of Q-learning and SARSA. Also report one of the final Q-tables of this experiment. Also assess the quality of agent coordination and how well the solution gets to the optimal subdivision of work.
2. In Experiment 3 you rerun either[[3]](#footnote-3) Experiment 1.c or 2 but with learning rates α=0.1 and α=0.5. When interpreting the results focus on analyzing the effects of using the 3 different learning rates on the system performance.
3. Experiment 4 is the somewhat similar to Experiment 1c or 2; you use α=0.3 and γ=0.5 in conjunction with either Q-learning or SARSA[[4]](#footnote-4) as follows: you run PRANDOM for the first 500 steps; next, you run PEXPLOIT; however, after a terminal state is reached the third time, change the two pickup locations to: (2,3,3) and (1,3,1); the drop off locations and the Q-table remain unchanged; finally, you continue running PEXPLOIT with the “new” pickup locations until the agent reaches a terminal state the sixth time. When interpreting the results of this experiment center on analyzing on how well the learning strategy was able to adapt to the change of the pickup locations and to which extend it was able to learn “new” paths and unlearn “old” paths which became obsolete.

For all experiments, if a terminal state is reached, restart the experiment by resetting the PD world to the initial state, but do not reset the Q-table. Run each experiment twice, and report[[5]](#footnote-5) and interpret the results; e.g., utilities computed, rewards obtained in various stages of each experiment.

Assess which experiment obtained the best results[[6]](#footnote-6). Next, analyze the various q-tables you created and try identify attractive paths[[7]](#footnote-7) in the obtained q-tables, if there are any. Moreover, briefly assess if your system gets better after it solved a few PD-world problems—reached the terminal state at least once. Briefly analyze to which extend the results of the two different runs agree and disagree in the 4 experiments. Finally, analyze how well the approach adapted to change in the fourth experiment.

Moreover,

* Make sure that you use different random generator seeds in different runs of the same experiment to obtain different results—having identical results for the 2 runs of the same experiment is unacceptable. It is okay just to report and interpret the Q-tables for the better of the two runs for each experiment, but you should report the performance variables for all eight runs.
* You should use the traditional Q-learning and SARSA algorithm in the project and not any other Q-learning variations or reinforcement learning algorithms!
* Never update the q-values of operators are not applicable in a Q-Tables!
* Groups who develop good methods for visualizing q-tables and good visualizations for the analysis of attractive paths obtain extra credit.
* Groups who develop sophisticated methods to analyze agent coordination receive a small amount of extra credit.
* Allow in your software design that the positions of dropoff and pickup positions might be changing before and during a run; otherwise, you will need to write a lot of additional software for experiment4.
* Evidence of the running of your system has to be provided using screen shots that will be delivered in a separate document.
* Groups that develop a very well designed and visually appealing visualization component will receive extra credit for this part of the 4368 Group Project!

Write a 8-12 pages report that summarizes the findings of the project. Be aware of the fact that at least 15-20% of the points available for this project are allocated to the interpretation of the experimental results. Finally, submit the source code of the software you wrote in addition to your project report and be ready to demo the system you developed.

**Implementation Guidelines**

1. Try to create whole project in a modular manner. There are numerous ways modules can be designed. Below one example is given:
	1. Modules:
		1. Environment: Define the environment in which the agent interacts and collects rewards.
		2. Agent: The agent decides what actions to take based on the state of the environment.
		3. State representation: Convert the environment state into a format that can be used by the agent.
		4. Action space: Define the set of possible actions that the agent can take.
		5. Reward function: Specify the reward signal that the agent receives after taking an action.
		6. Policy: The policy defines the strategy that the agent uses to select actions based on the state.
		7. Value function: The value function estimates the expected future reward of a state or state-action pair.
		8. Model: The model predicts the next state, reward, and terminal state given an action and state.
		9. Algorithm: The algorithm updates the agent's policy and value function using the experiences it collects while interacting with the environment.
		10. Training: Train the agent by repeatedly interacting with the environment and updating its policy and value function until it converges to an optimal solution.
		11. Evaluation: Evaluate the agent's performance by measuring its average reward, success rate, and other relevant metrics.
	2. Interaction Among the Modules:
		1. Environment: The environment generates an initial state and sends it to the state representation module. After that, the agent selects an action based on its policy and the current state, and sends it to the environment. The environment then transitions to a new state, calculates the reward based on the reward function, and sends the new state, reward, and terminal information back to the agent.
		2. Agent: The agent receives the state from the state representation module and calculates the action to take based on its policy. The agent also updates its policy and value function based on the feedback from the environment and the algorithm.
		3. State representation: The state representation module receives the raw state information from the environment and converts it into a format that can be used by the agent.
		4. Action space: The action space defines the set of possible actions that the agent can take. The agent selects an action based on its policy and the current state and sends it to the environment.
		5. Reward function: The reward function receives the current state and action from the agent and calculates the reward signal that the agent will receive after taking the action.
		6. Policy: The policy defines the strategy that the agent uses to select actions based on the state. The agent updates its policy based on the feedback from the environment and the algorithm.
		7. Value function: The value function estimates the expected future reward of a state or state-action pair. The agent updates its value function based on the feedback from the environment and the algorithm.
		8. Model: The model predicts the next state, reward, and terminal state given an action and state. The agent uses the model to plan its actions in advance and to improve its policy and value function.
		9. Algorithm: The algorithm updates the agent's policy and value function based on the experiences it collects while interacting with the environment. The algorithm uses the feedback from the environment, the model, and the value function to update the agent's policy and value function.
		10. Training: The training process is executed by repeatedly interacting with the environment and updating the agent's policy and value function until it converges to an optimal solution.
		11. Evaluation: The evaluation module measures the agent's performance by calculating its average reward, success rate, and other relevant metrics. The evaluation results are used to fine-tune the agent's policy and value function and to improve the overall performance of the system.

**Deliverable Materials**

1. A report
	1. The report must contain design of your system, e.g. modules and interactions among the modules. You can use (but not necessary) software development tools such as flow diagram or any other diagram to show the design
	2. You must explain the state space design, q-table design and implantation of all constraints assigned for each module
	3. It should include all the experiments you have conducted. The format you can write explaining an experiment is as follows:
		1. First give a summary of the experiment you are conducting its purpose
		2. Write down the steps as form of psudo code
		3. Explain the steps
		4. Demonstrate the results and explain them
2. Code
	1. Design the whole system modular fashion
	2. At the beginning of each module comment its purpose and how it interacts with other module
	3. Add comments for each block and function summarizing their purpose
	4. Name each variable and function such a way that their name explains them or add comment for each name
	5. A readme and/or makefile explaining your system dependencies and how to run it
	6. It would be good, if we can see the behavior of your system during training and after the completion of training

**Grading Rubric**

1. How accurately project is designed
	1. How accurate and sophisticated the state-space is?
	2. How accurate the Q-table is?
	3. How accurately agents follow the constraints?
	4. How accurately experiments are implemented (e.g. the way policies are instructed to be applied)?
2. How efficiently project is implemented
	1. How accurately modules are designed (e.g. every module is designed with proper task in mind)?
	2. All modules and variables are properly commented (e.g. easy to find the role of each module, function, block and variable just by seeing the code)?
	3. Is the any instruction about running the project (readme) and how good the readme is?
	4. Is there any make file for the project?
3. How good the report is?
	1. Does the report is well written, e.g.
		1. It has index of sections
		2. A well written brief summary about all the sections
		3. Properly organized sections
	2. Does the report contain a proper system design with instruction how different module works?
	3. Does the report contain the proper design and explanation for state space?
	4. Does the report contain the proper design and explanation for q-table?
	5. Does every experiments have:
		1. Proper instruction pseudo-code with explanation, and how good they are
		2. Outputs with explanation, and how good they are
4. Extra Credit
	1. Additional Visualization
	2. Addition module design
5. Team-work
	1. Based on evaluation of teammates among themselves. We will ask all the members to evaluate performance of other members. A person’s personal score will be evaluated based on the feedbacks he got from the teammates. We will first assign a score to the project, next we will assign individual score based on the project score and individual contribution. Consequently, personal score may be less than the main project score, if the person does not made enough contributions. Feedback need to be submitted by each member individually and separately (No other teammate will see this). A team member evaluation form is given at the end of this draft.

Project Links

<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>

[1911.10635.pdf (arxiv.org)](https://arxiv.org/pdf/1911.10635.pdf)

Team Member Evaluation Guideline

Please individually grade all of your group members with in the scale of 1 to 11 and upload it to the teams individually.

1. Totally Missing and were not involved at all (0% contribution)
2. Were involved in slight discussion only (10% contribution)
3. Were involved in discussion and slight planning about the project, but did not do anything else (20% contribution)
4. Were involved in discussion and generous amount of planning about the project, but did not anything else (30% contribution)
5. Were involved in discussion and total planning for the project, and helped slightly writing report but did not do anything significant (40% contribution)
6. Were involved in discussion and total planning for the project, and helped generously writing report, but contribution is still low (50% contribution)
7. Were involved in discussion, planning and also contributed slightly in implementation (moderate or 60% contribution)
8. Were involved in discussion, planning and also contributed generously in implementation (slightly over than moderate or 70% contribution)
9. Were involved in discussion, planning and also contributed slightly in both implementation and writing report (good contribution or 80% contribution)
10. Were involved in discussion, planning and also contributed generously in both implementation and writing report (very good contribution or 90% contribution)
11. Were very devoted to the project, involved in discussion, planning and also contributed highly in implementation and demonstrated leadership or unique creativity (best contribution or 100% contribution)

Group Name:

Member1 - Name: Grade:

Member2 - Name: Grade:

Member3 - Name: Grade:

Member4 - Name: Grade:

Member5 - Name: Grade:

1. SARSA is a variation of Q-learning that uses the q-value of the actually chosen action and not the q-value of the best action! [↑](#footnote-ref-1)
2. A step is the application of an operator by one of the two agents. [↑](#footnote-ref-2)
3. You have a choice here! [↑](#footnote-ref-3)
4. Again you have a choice here. [↑](#footnote-ref-4)
5. Additionally, report the following Q-tables for Experiments 2 (or Experiment 3 if you prefer that, in this case you will only need to report the final Q-Table of Experiment 2) in your report a) when the first drop-off location is filled (the fifth block has been delivered to it) and b) when a terminal state is reached and c) the final Q-table of each experiment.

The Q-table in the screenshot should be presented as a matrix, with *s* rows (states) and *t* columns (operators).  Thus, the Q-table for State Space 0, in the World 2022 pptx slides, has 25 x 2 rows and 6 columns; however, the q-values for the drop-off and pickup operators do not need to be reported. [↑](#footnote-ref-5)
6. Provide graphs that show, how the algorithm’s performance variables changed over the duration of the experiment in the three experiments. [↑](#footnote-ref-6)
7. A path going from (i,j) to (i’,j’) is attractive if, the q-values of the motion operators are high in comparison to other directions. Be aware of the fact that different paths are attractive for agents who hold a block and for agents how do not hold a block. [↑](#footnote-ref-7)