Exploration of Compiled Agents in Multi-Agent Taxi

Noah Carver NCARVER1@UMBC.EDU

1. Introduction

Multi agent problems usually call for complicated agent-world structures that suffer from clunky overhead and poor explainability. The intent behind this project is to explore the feasibility of an alternative to distributed multi-agent reinforcement learning.

1.1. Agent Based Reinforcement Learning

Agent based Reinforcement learning is based off of the interaction between two concepts:

- **World** The world receives actions from the agent and returns a representation of the modified state of the world as well as the reward the agent receives and whether or not the agent has completed its task.
- **Agent** The agent takes in the state of the world and chooses an action to send to the world. It learns this choice from previous State-action-reward triplets that it has received.

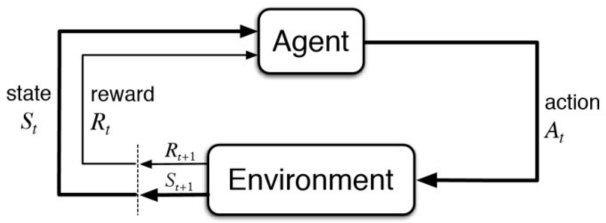

Figure 1. World-Agent Interaction Visualization

Some vocabulary pertaining to Agent Based Reinforcement Learning:

- **Policy**: The learned set of rules that the Agent uses to make its decisions.
- **Epoch**: The period of time in which an action is taken.
- **Episode**: Period of time in training that is bracketed by resetting the domain. It ends either when the problem is solved or at a pre-defined maximum number of epochs

1.2. Markov Decision Process

The Markov decision process (MDP) is simply a way to codify this interaction. It is written as the 5-tuple:

$$\{S, \mathcal{A}, T(s,a,s`), R(s,a,s`), \mathcal{E}\}$$

where S denotes the State space, or the set of all possible states that the world can be in, $\mathcal{A}$ denotes the set of actions that an agent can take, $T(s,a,s`)$ is the probability of reaching state $s` \in S$ by taking action $a \in \mathcal{A}$ at state $s \in S$, $R(s,a,s`)$ is the reward received in that case and $\mathcal{E}$ is the set of states in which the agent has completed its task.

MDPs have a few interesting properties:

- **Markov Property** The Markov Property asserts that the future is independent of the past given the present. This means that future states and rewards are only based on the current state and whatever actions are taken in the future.
- **Fully Observable** This property is somewhat derived in the Markov Property. For a domain to be fully observable worlds have states that do not hide information from the agent.
- **Discrete time** The Discrete time property asserts that all actions take the same time to complete (all epochs are of equal length).

There are MDP Variants that do not have these properties (Continuous Time MDPs, Partially Observable MDPs), but those are not relevant here.

1.3. Multiple Agents

The most noticeable difference when transitioning from single to multi-agent reinforcement learning is that, in the viewpoint of each agent, the result of it's action at a state does not entirely determine the resulting state. The resulting state is also dependant on the other agent's action. This directly contradicts the Markov Property, meaning that no true multi agent structure can be a Markov Decision Process.
This co-dependence extends to each agent's optimal policy, meaning that the optimal policy of one agent changes if the policy of the other agent changes.

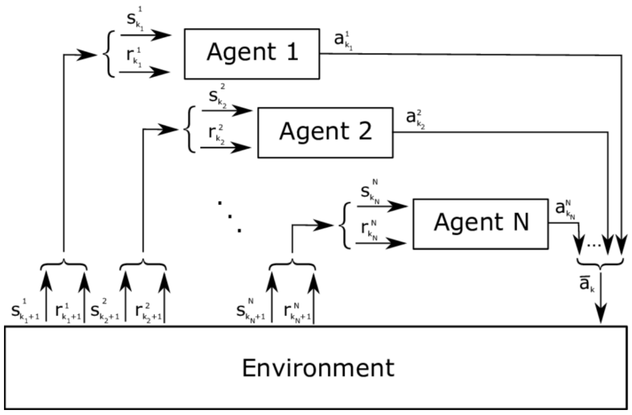

Figure 2. Multi Agent Structure Visualization

1.4. Q-Learning

Q-Learning is a simple and common reinforcement learning algorithm that functions by assigning a value to each state-action pair describing the discounted expected future reward of taking that action at that state. This Q value can be learned iteratively by using the following rule:

$$Q_{s,a} := (1-\alpha)*Q_{s,a}+\alpha(r_{s,a}+\gamma*max_a(Q(s',a))) \quad (1)$$

Policies can be generated using:

$$\pi(s) = argmax_a(Q(s,a)) \quad (2)$$

Over time, Q will converge to an optimal Q* and π will converge to an optimal $\pi*$

2. Problem Definition

2.1. Classic Taxi

The Taxi environment describes a 'taxi' agent that moves in cardinal directions on a grid. The grid also contains a number of 'locations' and 'passengers'. The passengers exist only at locations and in the taxi, and each has a specific location that is their destination. The simulation ends when all passengers are delivered to their desired locations.
In this case, the state space is composed of the location of each passenger and the coordinates of the taxi. Additionally, the action space is composed of moving the agent in each of the cardinal directions and picking up and dropping off a passenger.

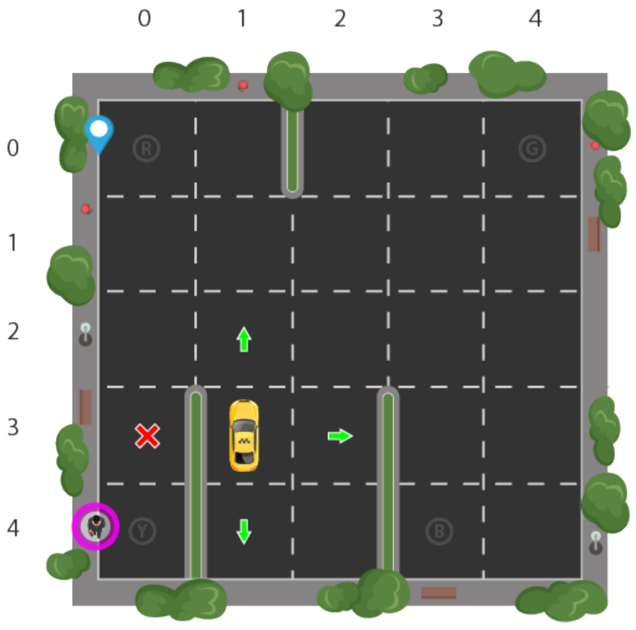

Figure 3. Taxi Domain Visualization

In this example, R, Y, B, and G are locations, the purple circle is a passenger and the blue tag is that passengers destination. The green arrows indicate the directions the taxi can move and the red X indicates that the taxi cannot move through walls.
Note: While this image only has one passenger, the domain used in testing had two.

2.2. Multi-Taxi

The Multi-Taxi Variant of the Taxi environment simply adds a second taxi. However, this results in a number of further complications. Aside from having multiple taxis that must now coordinate which passengers each are intending to pick up, there is the additional issue of the taxis attempting to occupy the same space at the same time.
The state space is largely the same, except that it contains the coordinates of the second taxi. This environment has two action spaces, as there are 2 agents. Each of these action spaces are identical to the Single agent action space.
This means that each agent needs its own policy and commonly a separate learner for that policy. This leads to complicated, if nicely parallel, structures in which multiple, separate learner-agents affect the same world at the same time.

3. Proposed method

The method explored in this project is to compile both agents into one single-agent markov decision process that can be solved with a simple reinforcement learning algorithm like Q-Learning. To do this, the action space becomes the set product of the action spaces of each agent.

$$\alpha = \alpha_1 \times \alpha_2 = \{(a_1, a_2) : a_1 \in \alpha_1, a_2 \in \alpha_2\}$$

Where α refers to an action space and a is an action in said action space. This makes the action space exponentially larger than the equivalent single agent action space. This is similar to the concept of a Joint-action Learner in that actions are learned jointly, as opposed to each agent learning a separate policy based off of a model of the other agent or learned probabilities.
Additionally, only one policy needs to be learned, as each action can be factored into the actions for both agents. Since there is only one macro-action, the Markov Property is restored, meaning that this pseudo-multi-agent structure returns to being a Markov Decision Process. There is no co-dependence between policies since only one policy is needed.

3.1. Intuition

Theoretically, a simple Q-Learner should be able to converge on this domain given an adequate amount of time. The state space and action space are very large, making it very time consuming to explore and to populate Q-values, but it is possible.

4. Experiments

Before running these experiments, tuning was required in order to reach the results shown here. The Hyper-parameters reached are as follows:

$$\begin{aligned} &\alpha = 0.05 \\ &\gamma = 0.7 \\ &\epsilon = 0.001 \\ &\text{number of episodes} = 20000 \\ &\text{maximum number of epochs} = 5000 \end{aligned} \tag{3}$$

These parameters were used for all tests.

4.1. Epochs per episode

The first experiment is to test the number of epochs per episode as this will show the agents solving the domain faster as it learns.

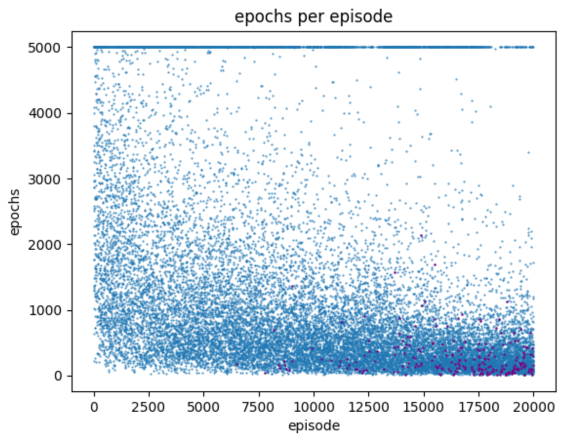

Figure 4.

4.2. Reward per episode

The second Experiment is to view the reward received by the agents. This shows the number of mistakes (ie: collisions) that are made as the algorithm learns.

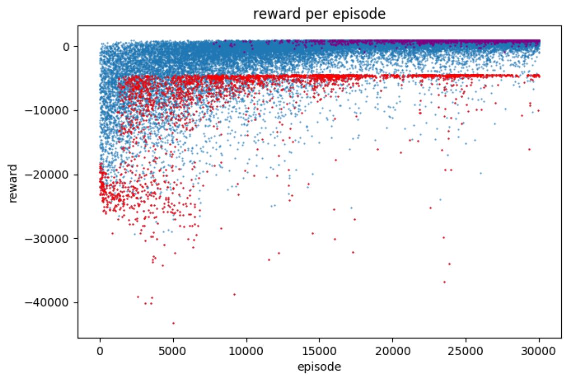

Figure 5.

4.2.1. Episodes with No 'Mistakes'

There are also data points in figures 4 and 5 that are purple. These datapoints represent episodes where no negative reward was received other than -1 each epoch as a temporal motivator. This means that the agents never collide and they never attempt to take the pick up or put down actions when they cannot. Notice how, as the algorithm learns, the frequency of episodes with no mistakes increases.

4.2.2. Cut-off Episodes

There are a number of data points in figure 5 that are colored red. These represent episodes in which the agent never solves the domain. Notice that these points converge to a line at around -5000. This is due to the 5000 epoch cut-off. However, the fact that these converge to around -5000 means that even when the agents fail to solve the domain, they almost qualify as having no mistakes. Additionally the amount of episodes that do not finish decrease as the algorithm learns.

4.3. Comparison to Single Taxi

It is important to be able to compare this with the basic single taxi domain. This was done using the exact same Hyper parameters (α,γ, number of episodes, max epochs per episode)

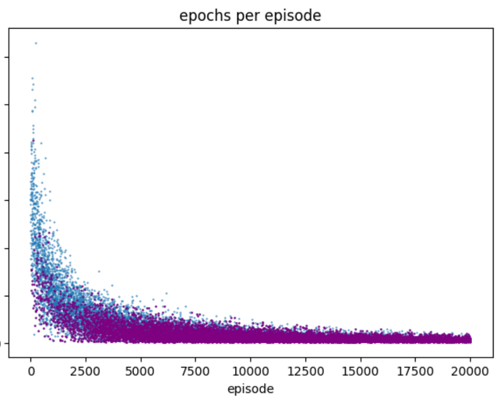

Figure 6. Epochs per Episode (One Taxi)

Compare figure 6 to figure 4. Note that the Algorithm converges faster and reaches a much more concentrated result. Additionally note the lack of the line of data points at the top, this indicates that the single taxi has no difficulty solving the domain. In fact the single agent never failed to solve the domain. In it's first iteration it solved the domain in just over 3000 epochs.

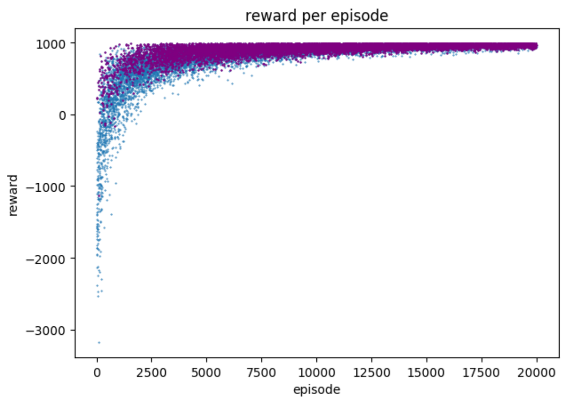

Figure 7. Reward per Episode (One Taxi)

Much of the same observations can be made comparing figure 7 and figure 5. Notice the absence of the red line of cut-off episodes and the larger quantity of purple data points with no 'mistakes'.

4.4. No Epoch Limit

In this experiment, the upper limit to the number of actions the agents can take in an episode was removed. This means that all episodes will run until the agents deliver the passengers, regardless of how long that takes.

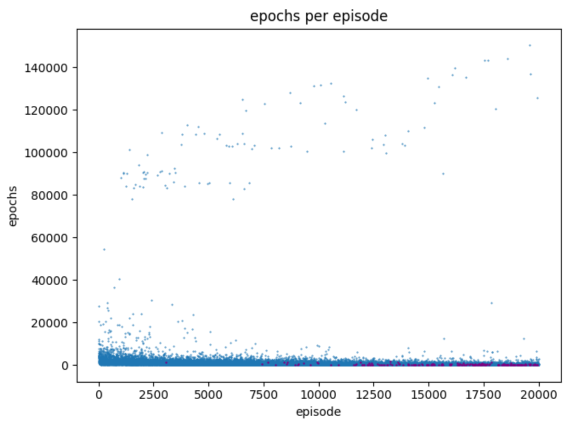

Figure 8. Epochs per Episode (Two Taxis, no cut-off)

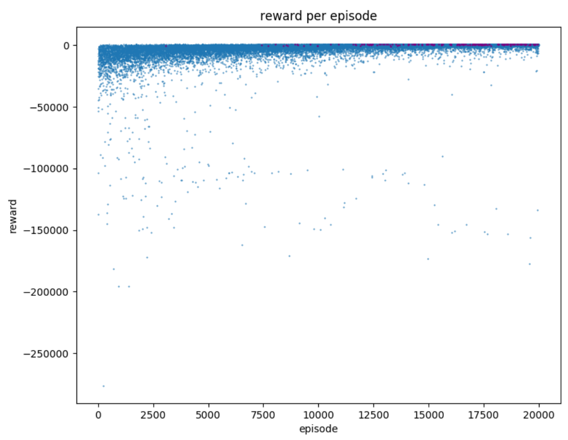

Figure 9. Reward per Episode (Two Taxis, no cut-off)

Initially, these graphs look quite similar to those of the Single Taxi. However, note that the scale of the y-axis is much larger than that of any of the previous figures. Adjusting this scale reveals that removing the upper bound for epochs did little to improve the performance of the algorithm. The only relevant difference is the absence of the lines denoting unfinished episodes. These data points are still present in figures 8 and 9 but were cut off when the y-axis was adjusted.

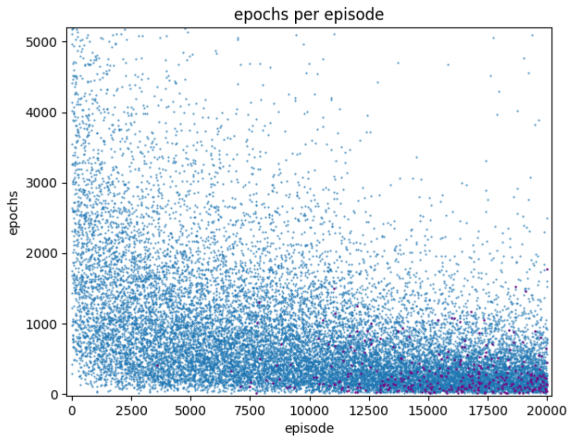

Figure 10. Epochs per Episode (Two Taxis, no cut-off, Cropped)

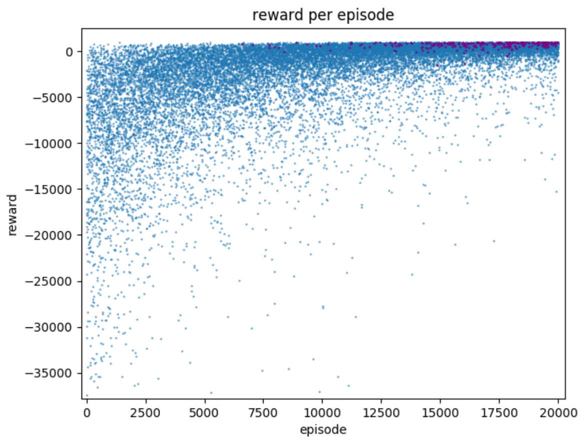

Figure 11. Reward per Episode (Two Taxis, no cut-off, Cropped)

5. Conclusions

Foremost of the conclusions to be found here is that, at least for this particular domain, Multi-Agent Compiling does not present a benefit over a single agent doing twice the work. Even though the performance ceiling is higher in the case of multi-agent compiling – each agent can pick up a passenger and deliver it, halving the work needed from each agent – it does not reach that ceiling, much less reach it consistently, especially when compared to the consistency of the single agent.

References

Claus, Caroline and Boutilier, Craig. The dynamics of reinforcement learning in cooperative multiagent systems. *AAAI/IAAI*, 1998:746–752, 1998.

Ghavamzadeh, Mohammad, Mahadevan, Sridhar, and Makar, Rajbala. Hierarchical multi-agent reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 13(2):197–229, 2006.

Littman, Michael L, Dean, Thomas L, and Kaelbling, Leslie Pack. On the complexity of solving markov decision problems. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pp. 394–402. Morgan Kaufmann Publishers Inc., 1995.

Neto, Gonçalo. From single-agent to multi-agent reinforcement learning: Foundational concepts and methods. 2005.