An Exploration of Socially Driven Multi-Agent Reinforcement Learning Models

Yousuf Baker
M.S Electrical Engineering
Columbia University
ykb2105@columbia.edu

Trevor Gordon
M.S Electrical Engineering
Columbia University
tjg2148@columbia.edu

Rakshith Kamath
M.S Electrical Engineering
Columbia University
rk3165@columbia.edu

I. Introduction

This report will undertake a brief exploration and synthesis of two key reinforcement learning papers by *Leibo et. al.* [3] and *Jaques et. al.* [2]. This exploration will begin first by motivating the core concepts of the papers, then developing the theory and mathematical models , and lastly by undertaking an experimental exploration of the papers methods and a discussion of these results. The overarching goal of this paper is to develop a precise theoretical understanding of these papers' concepts, as well as to undertake an exploration of the social behaviors that may arise by adjusting the dynamics of the learning process, a similar approach in spirit to [3]. The order of papers was chosen based on logical progression, but also in terms of what the papers address: the *Leibo et. al.* paper develops an environment which encodes social behaviors, and the *Jaques* paper takes this environment as a given, and seeks to develop agents which have social behavior built into their design and training.

II. Challenges

The main challenges of this paper include:

1) Understanding the theoretical framework in both of these papers. While the papers fall into the landscape of reinforcement learning that we have studied in class, the theory is both novel and orthogonal to concepts we have learned in class. Thus this section in specific will extend well beyond the concepts and mathematics introduced in class in both depth and level of detail.
2) Annotating and understanding the code implementation such that we are able to construct and undertake our own experiments
3) Running code experiments at the sufficient scale as to observe results given our limited hardware

III. Literature Review

A. General Concepts and Review

For both papers it is necessary to introduce the concept of multi-agent partially observable markov games. **Note: the following formulation is adapted directly from [3]**

Let our two player partially observable markov game be represented by $\mathcal{M}$, with the following properties: state space $\mathcal{S}$, observation space defined by the mapping

$$\mathcal{O} : \mathcal{S} \times \{ 1, 2 \} \to \mathbb{R}^d$$

and each agents current observation of the environment defined by $\mathcal{O}_i = \{ o_i \mid s \in \mathcal{S}, o_i = \mathcal{O}(s, i) \}$.

Further, each agent has associated with it a set of actions allowable from any state: $\mathcal{A}_1, \mathcal{A}_2$

From there, our transition function is defined conventionally as a discrete probability distribution over the state space:

$$\mathcal{T} : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \Delta\mathcal{S}$$

Note that though this formulation includes a model (transition function), knowing this transition function explicitly is not necessary to our purposes since we take a model free approach to learning as mentioned later) However, our reward for each player defined by cross between sets of both players actions, and the state space:

$$r_i : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \to \mathbb{R}$$

and our policy for each agent i is constrained by each agents observation of the environment. $\pi_i : \mathcal{O}_i \to \Delta(\mathcal{A}_i)$

given a discount factor $\gamma \in [0, 1]$ we can define a value function for each player i based on the joint policy (tuple of both policies: $\vec{\pi} = (\pi_1, \pi_2)$)

$$V_i^{\vec{\pi}}(s_0) = \mathbb{E}_{\vec{a} \sim \vec{\pi}(O(s_t)), s_{t+1} \sim \mathcal{T}(s_t, \vec{a}_t)} \left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, \vec{a}_t) \right]$$

which diverges from the typical markov process formulation in which each agent's value function depends solely on itself and it's policy. This formulation will be key in developing later environment frameworks.

B. Multi-agent Reinforcement Learning in Sequential Social Dilemmas

1) Key Ideas

This paper develops an environment for multi-agent reinforcement learning tasks, where the environment has built into it a reward scheme inspired by social dilemmas derived from game theoretic principles. Rather than taking a "prescriptive"

stance and trying to figure out an optimal policy for agents based given this social environment, *Leibo et. al.* take the "descriptive" approach of exploring what social behaviors evolve from the learning dynamics and agent-environment interactions [3].

At the highest level, a social dilemma is a situation with competing individual and collective interests and rationality, though this creates the possiblity of self interested behavior because of the allure of individual behavior (a "tragedy of the commons") [3].

*2) **Theoretical Framework and Analysis***

This first paper uses the formulation of social dilemmas as matrix games, known as matrix game social dilemmas (MGSDs) as it's foundation. Though it is important to note that the key difference between matrix games and MGSDs is that the latter have specific social concepts tied to their outcomes, that are modeled by the inequalities listed in [4]: R ("reward for mutual cooperation"), P (punishment from both agents defecting, i.e. "*mutual defection*"), S (outcome for an cooperating agent with defecting partner, "*sucker*" outcome), T ("*Temptation*" outcome achieved by an agent defecting against a cooperator) [4].

However, MGSD formulation fails to capture the following real world properties of social dilemmas [3]:

1) not time discrete event ("temporally extended")
2) Cooperating and defecting are usually defined by a range of actions (i.e. a *policy*), rather than a single action
3) Both cooperating and defecting can be "graded quantities", meaning that there can be degrees of cooperation/defection
4) players take actions with a brief time lag ("quasi-simultaneously") since one players action can inform the others
5) all players only have partial observations of the environment state and actions of other players

and so *Leibo et. al.* address these limitations by combining MGSDs with multi agent partially observable Markov games to develop what is called a sequential social dilemma. Intuitively, the sequential social dilemma can be thought of as an RL environment where the reward scheme and value functions are bound by the game theoretic inequalities that define social dilemmas, and where each successive, individual state is a social dilemma matrix game.

[3] combine the two environment schemes aforementioned (MGSD, social dilemma matrix game) by defining specific value functions, and constraining them as follows:

(note: C is cooperation, D is defection)

$$R(s) := V_1^{\pi^C,\pi^C}(s) = V_2^{\pi^C,\pi^C}(s) \tag{1}$$

$$P(s) := V_1^{\pi^D,\pi^D}(s) = V_2^{\pi^D,\pi^D}(s) \tag{2}$$

$$S(s) := V_1^{\pi^C,\pi^D}(s) = V_2^{\pi^D,\pi^C}(s) \tag{3}$$

$$T(s) := V_1^{\pi^D,\pi^C}(s) = V_2^{\pi^C,\pi^D}(s) \tag{4}$$

these value functions represent our four social dilemma outcomes, adapted from [3], and that must bind to the game theoretic inequalities and formulation of social dilemmas from earlier. Further, each agent has associated with it two policies, π^C and π^D, which are known as cooperation and defection policies. With all these pieces in place, we can fully formalize our sequential social dilemma environment.

From *Leibo et. al*, our sequential social dilemma is defeined by the tuple $(\mathcal{M}, \Pi^C, \Pi^D)$, where Π^C, Π^D define **disjoint** sets of all possible cooperation and defection policies and $\mathcal{M}$ is our Markov game as defined above and we construct an *Empirical Payoff Matrix* from our value functions, induced by our policies: $\pi^C \in \Pi^C, \pi^D \in \Pi^D$. Further, as mentioned above, this tuple must satisfy the constraint that for $s \in \mathcal{S}$ our value functions must satisfy the social dilemma inequalities constraints defined in [4].

Though the section above was heavy in notation, there are key theoretical properties we can observe:

1) inherent to this models construction is a temporal dimension, social dilemmas exist and happen sequentially and each successive social dilemma is based on the previous one
2) the union of the cooperation and defection policy sets does not neccesarily give the set of all legal policies (ie: not necessary that $\Pi^C \bigcup \Pi^D = \Pi$). This has the implication of cooperativeness and defectiveness being graded quantities, ie they do not always combine perfectly into a full set, and can be imbalanced/exist out of mathematically expected bounds. This in turn is more reflective of real world social behavior

Leibo et. al. use the property of point two by "thresholding a social behavior metric" in order to define our cooperation and defection policy sets. They give an example of

$$\alpha : \Pi \to \mathbb{R}$$

(ie maps the policy to a continuous range of values on the real line, describing the "social aggressiveness of the policy, used to threshold cooperation/defection). This has the effect of allowing or disjoint policy sets to capture a gradation in the degree of cooperation or defection. Finally, the "local decision [processes]" [3] are non-markovian which in itself is an important property since social agents in real life generally are not memoryless.

Finally, in terms of training, "each agents learning depends only on the others", which means that agents considers others as part of the environment, where the slowly changing distribution of one agents experience interpreted as non stationary env in eyes of other.

Though all this considered, *Leibo et. al.*'s model still carries with it key limitations:

1) focuses solely on the environment and developing game theoretic environment rewards to encourage social behavior in agents (social models external to agents)
2) this paper does not provide a framework for finding a socially optimal policy
3) all agents training is centralized, as in they have access to each other's reward schemes and internal info

In actuality, the training method and exploration used in this paper is not as important, since this paper is the foundation on which the second paper is built, which provides a novel model of social agents and a decentralized training protocol which we will explore in our experiments.

C. *Social Influence as Intrinsic Motivation for Multi-Agent Deep RL*

1) Key Ideas

Whereas the previous paper was focused on developing an environment that encodes social dilemmas and behavior, the focus of this paper is to develop an agent with internal social behaviors, and specifically a form of internal motivation for agents to cooperate and act in support of the collective.

Another one of the key ideas in this paper is counterfactual reasoning, which is the process of considering alternate situations/states that may transpire would you have taken a different action. By definition counterfactual reasoning/thought tries to explore the causal relationship between your potential actions, and the alternate states. Another key novel contribution of this paper is their model of other agents (MOA) which allows the decentralized training of individual agents based on their cumulative observations of other agents' actions.

Jaques et. al. use counterfactual reasoning to develop the model of "intrinsic reward". This paper defines a "social influence intrinsic motivation" that adds an additional component to an agents reward based on it's "causal influence" on other agents' actions

2) Theoretical Framework and Analysis: Causal Influence

For an arbitrary agent k, the reward it receives is formulated as: $r_t^k = \alpha e_t^k + \beta c_k^t$ where βc_k^t is the reward it receives from "causal influence" [2]. Though how do we calculate this intrinsic influence reward?

Consider two agents, j and k at each time step, agent k asks itself the question "how would j's action change if I had acted differently in this situation?" [2]

Mathematically, this amounts to the following: at each t, a second agent j is able to condition its own internal policy on the action of agent k at that same time step: $p(a_t^j|a_t^k, s_t^j)$, we can them form a range of these distributions/policies for j's next action by sampling a number of alternate ("counterfactual") actions that k could've taken: $\tilde{a}_k^t$, and a conditional policy for j associated with each of these new actions $p(a_t^j|a_t^k, s_t^j)$. Averaging these conditional policies over all the range of sampled counterfactual actions gives us the marginal policy of j:

$$p(a_t^j|s_t^j) = \sum_{\tilde{a}_t^k} p(a_t^j|\tilde{a}_k^t, s_t^j)p(\tilde{a}_t^k|s_t^j)$$

this gives the policy of j in the case where it doesn't consider the counterfactual actions of k [2]

From this the paper defines the causal influence of k on j as the divergence between j's policy conditioned on k's counterfactual actions and j's marginal policy:

$$c_t^k = \sum_{j=0, j\neq k}^{N} \Big[D_{KL}[p(a_t^j|a_t^k, s_t^j) \| p(a_t^j|s_t^j)] \Big]$$

Intuitively, this also amounts to a measure of k's influence, because the divergence measures how much j's actions differ when considering k's action versus not. This formulation of causal reward has a number of interesting implications:

1) The calculation of causal influence is very closely related to the mutual information between the agents' actions (full derivation in appendix of paper, it isn't relevant to recount here)
2) It applies in cases where agents have direct access to information on each others rewards (centralized training as in baselines, communication cases) and when they are learning from observing each others actions (decentralized training, as in model of other agents)
3) experimentally shown to be robust to choice of divergence
4) social influence reward reduces the variance in policy gradient calculations as the number of agents in the MARL increases since it "introduces explicit dependencies across the action's of each agent" [2]

Thus far we have established a social model for the environment, and for the agents' interactions with each other. However, we have yet to consider the training process. To do so we will introduce and discuss the A3C training algorithm.

3) Training Algorithm

The asynchronous advantage actor critic(A3C) algorithm is used to update the value function and the policy at the same time, which was introduced and implemented in Google's DeepMind paper in 2016 [5]. We will begin with an brief explanation of the paper's terms: "asynchronous" here means that agents cab interact with the environment and update themselves at different time steps from each other, they do not need to be in sync. In the case of multi-agent systems, each agent will have its local set of parameters and the environment's copy with which it interacts to get updates. As time steps increases, each agent gains more knowledge which is then used to make asynchronous updates in the global network—thus increasing the total knowledge in the global network.

Next is actor-critic: the actor-critic is a unique technique which updates both the value *V(s)* and policy function $\pi(s)$. The policy function (Actor) is used to update the value function (Critic). This value function (Critic) is used to improve the policy (Actor) until convergence to the optimal policy. This iterative updating of policy and value function gives a simple, easy way of updating the function.

Traditional gradient methods discuss how discounted returns are used to reward/penalize the actions by the agents. To address this, we introduce the concept of advantage to give the agent better intuition understanding, and improve it's learning of the environment. It gives an idea of how much better/worse the rewards were compared to the expected return. The advantage is calculated as follows:

$A = Q(s, a) - V(s)$ [1]

The actor critic policy gradient method function approximates to replace policy for every possible action and state. This approximation is really helpful in situations that are high dimensional and continuous action spaces. Since it uses the simple iterative method which was discussed above it is more computationally efficient and has good convergence as well.

The policy gradient "score" function

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(s, a) Q^{\pi_\theta}(s, a) \right]$$

The advantage of A3C is that it can train many environments in parallel.

4) Theoretical Framework and Analysis: Modeling Agents and Decentralized training

Calculating the influence reward for the given agent requires the information about the other agent's actions, which was traditionally done by providing all agents with direct information of other agents (a "centralized" approach [2]) regardless of whether the agents could observe each other or not. The more practical and human-like approach would be to have each agent build a model of other agents based on the actions that it observes, and calculate its influence reward accordingly. This decentralized approach of independently training other agent's model is called *Model of other agents* (MOA).The MOA model for an agent k consists of two fully-connected layers connected to an LSTM, with each other agent $j \neq k, \forall j$ represented by their own LSTM trained on their actions as observed by k. This helps us in predicting the next state of other agents given their past actions [2].The learning of the MOA can be defined as the probability of predicting agent j's action at time step $t + 1$ given that we have the agent k's state and actions at time step t which can be denoted as-$p(a^j_{t+1}|a^k_t, s^k_t)$. As said previously this can only be updated only if the agent j is in the observable space of agent k, and thus MOA cannot update all agents like centralized methods.[6] One possible downside of this is that the agent might stay in the close proximity of the other just so as to have an option of passively increasing cumulative reward through influence, which might affect it's own actions and collective reward in the end. This can be seen as one of the caveats in the decentralized approach due to the lack of complete information available about other agents. Another possible downside is that one agent might only gain influence reward from observing one other agent, while not observing all others, which inherently limits it's long term reward and learning. While many papers have resorted back to centralized approach which is a well-known and tested for multi-agent reinforcement learning systems, the above said method can be used as a decentralized method with significant success.

IV. Experiment/Exploration Outline

Notes on our Experiments:

The code used for project is directly from [9], our main point for this project was not writing the code, but designing the experiments. The GitHub repository cited allows you to pass parameters for an experiment through the command line, and so we have included our experiment commands in the appendices.

The code used for project is directly from [9]. The authors structured their code in a way that significant experimentation can be executed from the command line as opposed to needing to modify source code. Our research involved reproducing the original authors experiments and analyzing performance across a range of experiment parameters. Our experimentation and analysis was motivated by our understanding of multi agent reinforcement learning concepts. To get started, we dove into the source code to understand how the command line arguments effect the experiment.

The following libraries are used in this project which are OpenAI gym, Petting Zoo and Ray RLlib. OpenAI gym is one of the python modules used to developing reinforcement learning algorithms and also helps in the API build and connection of algorithms to the environments and the agents. Petting Zoo is one of the libraries specifically used to deal with the multi-agent RL problems and it's simulations.Ray RLlib is an open source library used to support and distribute the large workloads which is especially a requirement for multi-agent RL experiments. This platform also provides scalability and its ease of integration with Pytorch and and Tensorflow makes it use a commonplace in these situations.

A. Experiment Setup

All experiments were run in the *Harvest* multi-agent RL environment created in petting zoo, and explained in detail in [3] [2] [9]. To give a brief description, harvest is an environment where at each step an agent can choose to move, shoot a beam at other agents, and harvest apples. Harvesting apples yields a positive reward, getting beamed yields a negative reward, and the other actions yield reward 0.The agents in the environment should make sure that each individual agent does not get greedy and eat all the apples which results in very slow re-growth of the apples/rewards. They need to co-operate amongst themselves ideally so as to leave sufficient apples. Figure 1 shows a labeled screenshot of the rendered harvest environment for our experiments which gives the idea of the setup and the environment with which we are working with.

For our experiments, we ran trials adjusting the number of agents, the weight of influence reward, the influence reward schedule, and the learning rate and have a study on how these parameters affect the agents and in general the social problem itself.

In the first sets of experiments we tried reproducing results. Given we have limited time and resources, we didn't think it was practical to run full length averaged experiments. We found that we saw meaningful changes in results after $5 \cdot 10^6$ time steps, since at a lower timescale our collective reward function was still in a transient state, implying that our agents had not learnt a fixed set of social behaviors yet (whether that is acting passively, or eating apples).

For the sections below, we aimed to reproduce parts of the research on a smaller scale. The first experiments were

computed for a smaller number of time steps due to timing constraints.

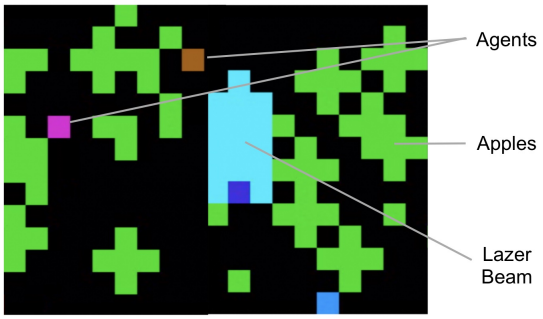

Figure 1. Screen capture from the harvest multi agent environment

V. Results and Discussion

A. Experiment 1: Agents learning to cooperate

One of our first experiments was run to understand whether agents could learn to cooperate. This experiment was ran with 5 agents for a fixed influence reward weight. Figure 2 shows the extrinsic rewards experienced per agent for a given simulation. It can be seen that in the beginning the agents experience negative rewards as they have not learned to cooperate and will beam each other with lasers, which we find to be reflected in the learning dynamics of the collective reward in later experiments. Further, it can be seen that agent_0 takes longer to learn than the other agents. There is a period where this agent's extrinsic rewards remain at zero as it is no longer getting hit by lasers but it hasn't learned to eat apples yet. In regards to it's connection to the mathematical theory, this is because we are optimizing for the collective reward, so it is one possible optimal solution for the harvest environment that the agent learns to stop beaming, and being beamed, but does not learn to collect apples since it is still receiving a large collective reward irrespective of the extrinsic apple reward [2].

B. Experiment 2: Adjusting the influence reward

We performed experiments where we adjusted the influence reward while keeping other parameters constant. We were looking to observe where a large influence reward would start to impede agent's learning. Furthermore, given time and compute resources, we were looking to observe a significant difference in agent behavior early on in the simulation compared to longer time frame learning. As such, we experimented with influence rewards that were significantly higher than the original authors optimal rewards. Our results in Figure 3 show our agent's learning didn't break down while increasing the reward as high as 1000 (a 400x increase from the original authors influence reward). A rendered video of the agents cooperating is included at [7]. It wasn't until we increased the reward to 10000 that we saw a decrease in agent episode reward mean. A rendered video of the agents failing to learn is included at [8]. This result shows two things. First, the author's use of causal influence as a means to to incentivize cooperation in a non direct way is robust to these large changes in influence reward. Second, as we expected, there is a value of influence reward where the agents become too tempted by influencing each other that the cease to learn useful skills for the environment and perform poorly as a result. below it is shown that agents collectively achieve high rewards and learn despite drastically larger influence rewards. These results are consistent with our expectation that agents that are too tempted to influence other agents will not learn.

We dove into the instantaneous extrinsic and social influence rewards to understand why we needed to increase the influence reward so high before the learning broke down. We found that even with a 10x increase in the relative weighting of the agents social influence reward, extrinsic (apple eating) rewards were still several magnitudes larger than the social influence reward. This is a combined effect of the environment and the number of agents. An agent will encounter many more apples than agents in the environment. We needed to drastically increase the social influence reward such that the sparse influential encounters with agents are worth optimizing for.

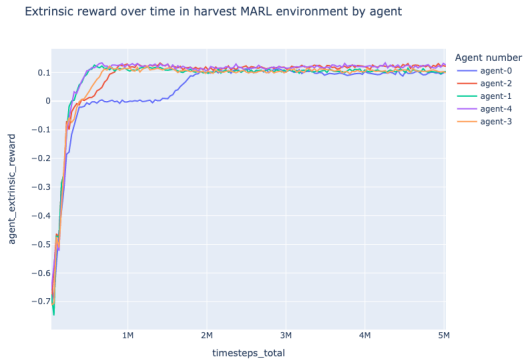

Figure 2.

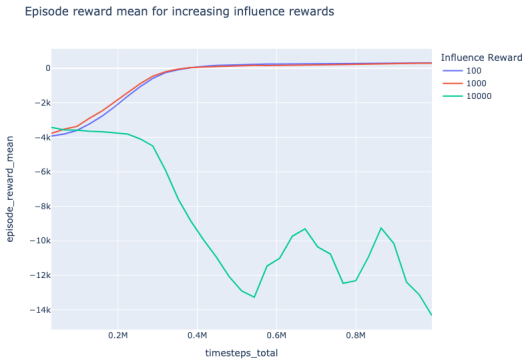

Figure 3.

C. Experiment 3: Adjusting the number of agents

For the next experiment, we fixed all learning parameters and environment variables, and ran our harvest environment for 2,4, and 6 agents. The resultant plot shows collective reward vs time step is seen in figures 3 and 4.

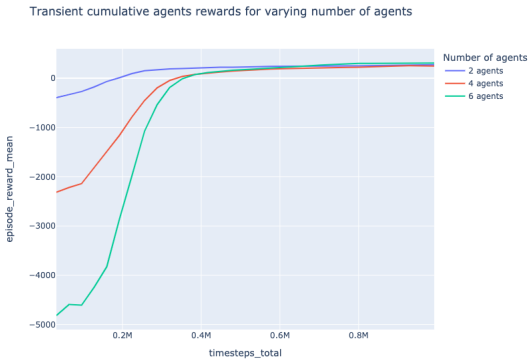

Figure 4.

The transient behavior in the learning process is best seen in figure 3, in which we see the initial collective rewards all begin in the negative, and their magnitude proportional to the number of agent. This implies that the agents begin by acting un-cooperatively and beaming each other, and thus inducing the negative reward. The case with the most negative initial collective reward is the 6 agent case, which relative to our lower agent cases, has the strongest incentive for defection since you have a higher number of agents competing for the same (and relatively scarcer) amount of resources. We can similarly see that the six agent case, while showing the steepest slope in learning from negative to positive, is the slowest to break into the positive collective reward range. This can be for a multitude of reasons, but some possibilities include:

- It is harder to coordinate a collective effort between a higher number of agents
- the agents take a longer time to observe sufficient amounts of other agents' actions to train their internal MOA and thus build their intrinsic motivation
- the initial reward for acting un-cooperatively is so high, because disabling even one other agent for an amount of time makes available to the remaining agents that many more apples

However, interestingly, we see the benefit of this intrinsic motivation and coordination in the steady state behavior shown in figure 4.

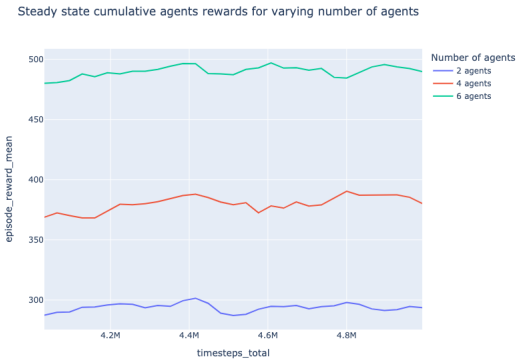

Figure 5.

This steady state behavior implies that the average collective reward received by each agent is significantly higher than the 2 and 4 agent cases, meaning that while the training and learning to coordinate is slower on the update in the 6 agent case, the incentive to coordinate and act collectively is significant in the long run as you increase the number of agents. In terms of social behaviors, this can be analysed in many ways, though one of the most interesting is that acting collectively as you increase the population size allows for a more efficient, equitable distribution of resources since coordination is required to optimally distribute the relatively scarcer number of apples amongst the growing population. Though another possibility is that some agents learn to not beam others, and is also no longer getting beamed, but also do not learn to eat apples, which yields a higher collective reward but does not entail a more equitable distribution of resources. This is

1) Adjusting Influence Reward Weight vs Number of Agents

This figure was compiled from our experiments above, and sought to establish a relationship between the different rewards, the number of agents, and the influence reward weight. Each single data point represent the per-agent steady state mean of either extrinsic rewards (dotted lines) or social influence rewards (solid lines) for varying number of agents and social influence reward weight. From this graph, we can see that for both a small and large influence reward weight, the mean received extrinsic reward decreases with increasing number of agents. It is likely that this result stems from a relative scarcity in resources as the number of agents increase. Next, it can be seen that by drastically increasing the reward weight from 2.5 to 1000, the agent's steady extrinsic rewards decreased at all number-of-agent values. The difference was largest for a smaller number of agents. This suggests that an above optimal influence reward weight has a higher cost when there is an abundance of resources. This makes intuitive sense as in the two agent case there are plenty of apples to be eaten. It wouldn't be in the agent's interest to try and influence other agents when they could be exploiting the environment. Lastly, it can be seen that the highest social influence reward was achieved with the largest influence reward weight and the highest number of agents. This makes sense as the more agents

there are, there is more opportunity to influence them. It is important to note that the instantaneous social influence reward in this graph doesn't necessarily translate into performing well in the environment. As shown in Figure 3, increasing the social influence reward from 100 to 1000 showed negligible effects on steady stead episode reward mean.

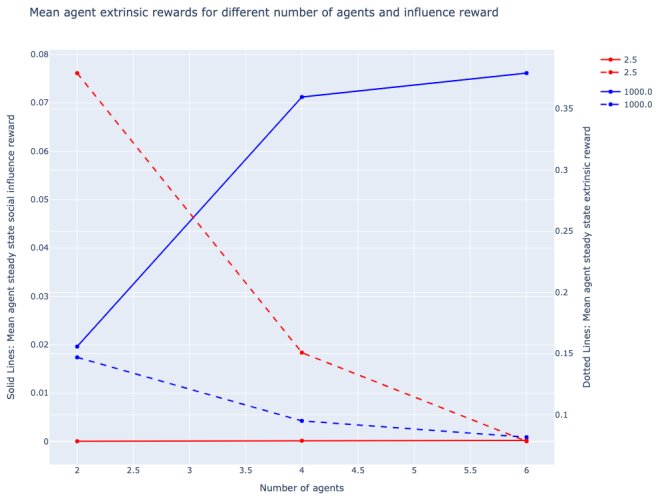

Figure 6. Extrinsic and social influence rewards summarized from above experiments. Each single data point represent the per-agent steady state mean of either extrinsic rewards (dotted lines) or social influence rewards (solid lines) for varying number of agents and social influence reward weight.

Further, we can see that in the case where the influence reward weight is lower, no influence reward is received regardless of the number of agents. In the higher influence reward weight case, the rate of increase in mean received influence reward is much larger from 2 to 4 than it is from 4-6. With this, it is difficult to discuss the implications of this with regards to potential social behaviors without tying it to a more granular examination of the MOA LSTMs at key change points for each of these scenarios and with the corresponding video. In actuality, this plot is made up of 3 points for line, so our analysis is also limited by the resolution of the data which in turn arises from our hardware limitations in running more trials and higher number of agents.

VI. Conclusion and Future Work

To motivate future work, we must first understand the limitations of this current work, not least of which is the need to hand tune the extrinsic and influence reward. From our literature review, and our experiments, we conclude that further limitations include:

1) In real world situations, it isn't clear how much you can influence others relative to non-stationary environment rewards (both these papers rely on the SSD environment, in which each successive state has a fixed social dilemma reward scheme)
2) While agents can optimize according to collective reward, and maximise their influence, this does not always translate to socially productive behavior.
3) Neither of these approaches seek to solve for an optimal set of cooperation and defection policies for agents in these social environments
4) the papers assume that agents act "*quasi-simultaneously*" [2], which is not accurate to real world behavior

The last two points especially limit the applicability to real world social situations, such as a generator selling to a wholesale energy market where we would want to optimize our policy to the competing interests of profit (defection) and acting to regulation (cooperation), and where most sales happen asynchronously.

Jaques et. al.'s paper also mentions that their approach is experimentally shown to be robust to the choice of divergence metric, though the exploration of such is omitted. Exploring the choice of this metric further could be of interest.

Further possible explorations in social behavior driven RL models include exploring methods where agents learn by parroting other, better performing agents' actions by keeping track of what agents do and trying to perform similarly. Another possibility is exploring models of where agents can enlist in social contracts with each other such that they don't need to influence each other overtly (such as by staying out of each others way, acting in shifts, or forming factions)—a potential application for block chain technologies.

There is much work to be done in pushing exploration of these social models outwards by synthesizing past and novel RL methods with those from other fields, as well as pushing exploration of these models inwards by expanding the range of applications they are experimented with.

References

[1] Asynchronous advantage actor critic (a3c) algorithm. https://www.geeksforgeeks.org/asynchronous-advantage-actor-critic-a3c-algorithm/. Accessed: 2021-12-14.

[2] N. Jaques, A. Lazaridou, E. Hughes, Ç. Gülçehre, P. A. Ortega, D. Strouse, J. Z. Leibo, and N. de Freitas. Intrinsic social motivation via causal influence in multi-agent RL. *CoRR*, abs/1810.08647, 2018.

[3] J. Z. Leibo, V. Zambaldi, M. Lanctot, J. Marecki, and T. Graepel. Multi-agent reinforcement learning in sequential social dilemmas, 2017.

[4] M. W. Macy and A. Flache. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7229–7236, 2002.

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

[6] B. K. P. Hernandez-Leal and M. E. Taylor. Agent modeling as auxiliary task for deep reinforcement learning. *AIIDE*, vol. 15:no. 1, pp. 31–37, Oct. 2019.

[7] Y. B. Rakshith kamath and T. Gordon. harvest example of co-operation.

[8] Y. B. Rakshith kamath and T. Gordon. harvest example of non-cooperation.

[9] E. Vinitsky, N. Jaques, J. Leibo, A. Castenada, and E. Hughes. An open source implementation of sequential social dilemma games. https://github.com/eugenevinitsky/sequential_social_dilemma_games/issues/182, 2019. GitHub repository.