Obedience-based Multi-Agent Cooperation for Sequential Social Dilemmas

by

Gaurav Gupta

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2020

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

We propose a mechanism for achieving cooperation and communication in Multi-Agent Reinforcement Learning (MARL) settings by intrinsically rewarding agents for obeying the commands of other agents. At every timestep, agents exchange commands through a cheap-talk channel. During the following timestep, agents are rewarded both for taking actions that conform to commands received as well as for giving successful commands. We refer to this approach as obedience-based learning.

We demonstrate the potential for obedience-based approaches to enhance coordination and communication in challenging sequential social dilemmas, where traditional MARL approaches often collapse without centralized training or specialized architectures. We also demonstrate the flexibility of this approach with regards to population heterogeneity and vocabulary size.

Obedience-based learning stands out as an intuitive form of cooperation with minimal complexity and overhead that can be applied to heterogeneous populations. In contrast, previous works with sequential social dilemmas are often restricted to homogeneous populations and require complete knowledge of every player's reward structure. Obedience-based learning is a promising direction for exploration in the field of cooperative MARL.

Acknowledgements

I would like to thank Dr. Jesse Hoey for all of his guidance and support, along with my readers Dr. Kate Larson and Dr. Peter Van Beek.

I would also like to thank all the members of the CHIL lab, in particular Joshua Jung and Alexander Sachs for their insightful discussions.

I'm so deeply grateful to my mother, without whom I would not be where I am today.

Finally, I'd like to thank Damla, my Askim, for all of her patient support at every step of this thesis.

Dedication

This is dedicated to my Nani and Nanu, who always prioritized education and kindness in every aspect of their lives.

Table of Contents

List of Figures

Chapter 1

Introduction

Reinforcement learning is an increasingly-attractive option for adaptive computational agents due to its theoretical generalizability to complex problem spaces [43, 64]. However, most of the successes in reinforcement learning have been limited to single-agent domains [39]. While there has been a growing body of work pertaining to multi-agent reinforcement learning, there remain some key challenges that must be overcome for multi-agent learning to exhibit the same level of success as its predecessor.

Perhaps the most notable of these challenges is that of cooperation and collaboration [6], which is almost impossible to guarantee [32]. Unless the environment is purely competitive, agents must learn to work together in some fashion, if only to avoid wasteful conflict. Somewhat counter-intuitively, even purely cooperative multi-agent environments come with their own unique challenges. Simply training reinforcement learning agents on the joint reward of all the agents in the environment leads to the 'lazy agent' problem, where weaker agents will take a backseat to let higher-performing agents take over [66]. In contrast, evaluating agents based on their individual reward eliminates this problem, while creating a tension between cooperation and competition. Such agents would behave like standard single-agent reinforcement learning algorithms and selfishly compete over the sources of reward in the environment, essentially abandoning all notion of cooperation. We can draw parallels here to human multi-agent problems, where collective reward leads to social loafing and individual reward leads to egocentrism [50, 29].

The most challenging environments come with elements of cooperation and competition baked in to the problem description. The most prominent of these mixed cooperative-competitive environments is the Prisoner's Dilemma, which has guided computational and psychological research for decades [34, 4]. In Prisoner's Dilemma, players are paired up

and can choose to cooperate or defect over a number of rounds. Mutual cooperation is the preferred long-term strategy, but the short-term gains of defection can cause cooperation to break down. Prisoner's Dilemma belongs to a greater class of matrix games known as social dilemmas [41] (mathematical formulation reproduced in Section 2.4), which all capture this tension between short-term defection and long-term cooperation.

While all sorts of complex strategies, including reinforcement learning techniques [71], have been leveraged to play social dilemmas, the best algorithms are often simple and intuitive. An example of this is Rapoport's Tit-for-Tat [56], which has been victorious in multiple tournaments [2, 3] by simply starting with cooperation and subsequently mimicking the opponent's last move. In real-world social dilemmas, cooperation and defection is not a simple atomic action. Rather, a sequence of actions can be construed as cooperative on a graded scale. In order to better capture realistic social interactions, [34] introduced temporally-extended versions of matrix game social dilemmas known as sequential social dilemmas, which is the class of problems we work with in this thesis.

In sequential social dilemmas, the tension between cooperation and defection is maintained but extended temporally. For example, in the Harvest game [26] agents need to collect apples from 'apple patches' but apples re-spawn proportionately to the number of apples in the rest of the patch. In this context, defection is defined as repeatedly harvesting from a patch that is running low while cooperation comprises of staying away from patches that are running low so they have time to replenish. The greater complexity arising from this sequential structure necessitates the use of sophisticated multiagent reinforcement learning methods [34].

Research on sequential social dilemmas has mostly centered around assigning intrinsic rewards to agents based on psychologically-inspired motivations in order to guide cooperation [37]. As noted previously, naively assigning team reward or individual reward is not sufficient for cooperation in mixed cooperative-competitive environments. Hughes *et al.* [26] tackled the problem by introducing intrinsic rewards based on 'guilt' and 'envy' while [20] took an alternative approach by assigning intrinsic rewards based on reciprocity. Recently, [28] used causal influence to achieve state-of-the-art scores, where agents are intrinsically rewarded for disrupting the action distributions of neighbouring agents. Jaques *et al.* [28] conducted an ancillary experiment in which causal influence is combined with a cheap-talk channel where sending and receiving communication has no cost; agents are rewarded for sending symbols that maximise disruption. We build upon this idea within the same problem domain to propose obedience-based learning.

1.1 Problem Description

The question we ask in this work is: **If we define a simple communication protocol that allows agents to exchange commands, does providing intrinsic rewards for obedience and leadership enhance cooperation in partially-observable sequential social dilemmas?**

We believe that since selfish agents are already naturally predisposed to selfishly commanding other agents to cooperate, adding an intrinsic reward for obedience should push the population towards a greater degree of cooperation. Additionally, we believe that making this exchange provide intrinsic rewards to both parties, the commander and follower, should further foster cooperation and speed up learning. While possibly not as powerful as other handcrafted approaches, the strength of this approach lies in its intuitive simplicity and generalizability.

In this work we provide empirical evidence for an affirmative answer to the research question by simulating two sequential social dilemmas with an added cheap-talk channel used to exchange commands. We evaluate the flexibility of our approach by varying vocabulary size as well as using a heterogeneous population. Also, we distinguish ourselves from the growing field of emergent communication in reinforcement learning, as we explicitly define a communication protocol for the agents to use, guided by intrinsic reward.

1.2 Thesis Contributions

The following summarizes the key contributions of this thesis:

- We propose a novel form of psychologically-motivated intrinsic reward denoted as obedience-based learning.
- We demonstrate the potential for obedience-based learning to improve multiagent cooperation by evaluating performance on sequential social dilemmas, a popular and challenging benchmark for cooperation.
- We observe the effects of varying the size of the communication vocabulary as well as the homogeneity of the population.
- We provide an up-to-date open-source implementation of the Harvest and Cleanup sequential social dilemmas with an optional cheap-talk channel built-in.

1.3 Thesis Organization

The rest of the thesis is organized as follows:

- In Section 2 we provide an overview of the necessary background required for the work in the thesis. This includes a review of reinforcement learning and social dilemmas. The latter includes the rules of Harvest and Cleanup, the games we use in this work.
- Sections 3 and 4 encapsulates the entirety of our original work; we start with a detailed explanation of obedience-based learning and our experimental setup before continuing onward to the results of our experiments.
- Section 5 lays out our closing thoughts as well as multiple recommendations on follow-up work.

Chapter 2

Background and Related Works

In this section we provide an overview of the relevant literature across the number of domains touched upon in this thesis. We review Markov Decision Processes before discussing recent advances in reinforcement learning with an emphasis on multiagent reinforcement learning. We will then provide an overview of works pertaining to agent cooperation in multiagent settings before discussing social dilemmas.

Reinforcement learning is an area of machine learning inspired by behavioural psychology, and encompasses a suite of techniques that help software agents take actions in an environment so as to maximize some notion of reward. Here, agents can be defined as "something that acts" or as "something that perceives and acts in an environment" [57].

2.1 Markov Decision Processes

An agent sequentially interacts with the environment over a sequence of timesteps $t = 1, 2, 3, \ldots$ by executing actions. After every action the environment provides feedback in the form of observations and rewards. Observations are used in determining the agent's next action, while maximizing cumulative reward is the agent's goal.

This formulation is typically modelled as a Markov Decision Process (MDP). An MDP can be represented as a 6-element tuple (S, A, P, R, γ, h) where S is a finite set of possible states, A is a finite set of possible actions, $P(s'|s, a)$ is the transition probability of moving from state s to state s' through action a, $R(s, a, s')$ is similarly the reward obtained by using action a to transition from state s to state s', γ is the reward discount factor, and h is the horizon i.e the number of time steps under consideration. The goal of the agent is

to maximise expected cumulative discounted reward, given by $\sum_t^{h-1} \gamma^t R(s_t, a_t, s_{t+1})$. Note here that the discount factor $0 \leq \gamma \leq 1$ penalizes later rewards, which allows us to work with $h = \infty$ while giving priority to early learning.

Agents make decisions based on policies, which are functions that capture the choice of action at each time step. A policy can be defined as $\pi(y_t, s) = a$, which maps the current state and history at timestep t to what action should be taken. In fully-observable environments, the history is the sequence of past actions, states, and rewards that brought us to this point, $y_t = a_1 s_1 r_1 a_{t-1} s_{t-1} r_{t-1}$. In partially-observable environments, the observations o_t provided by the environment do not encapsulate the entirety of the state, so the history becomes $y_t = a_1 o_1 r_1 a_{t-1} o_{t-1} r_{t-1}$. In this case the system is known as a partially observed Markov Decision Process (POMDP). If the choice of action does not depend on how the state was reached, we obtain what is known as a stationary policy which is defined simply as $\pi(s) = a$. For simplicity, we will assume stationary policies going forward until multiple agents are involved.

Clearly, the space of possible policies is immense. Agents can evaluate policies by computing their expected cumulative reward, also known as the value function of a policy:

$$V^\pi(s_0) = \sum_{t=0}^{h} \gamma^t \sum_{s_t} P(s_t|s_0, \pi) R(s_t, \pi(s_t), s_{t+1}) \tag{2.1}$$

We can also define an extended version of value functions known as Q-functions which compute the total value of a state action pair. Intuitively, this is the expected reward of the state action pair followed by the expected value of all subsequent actions. Here we denote the successor state of s by s' instead of using t subscripts:

$$Q^\pi(s, a) = E[R(s, a)] + \gamma \sum_{s'} P(s'|s, a) V^\pi(s') \tag{2.2}$$

The agent's goal is to pick an optimal policy that maximises value, denoted π^*. In fully observed environments, stationary optimal policies are guaranteed to exist [66].

In order to obtain an optimal policy, there are three primary classes of exact solution methodologies for MDPs: value iteration, policy iteration, and linear programming. We will provide a brief overview of value iteration and policy iteration, as they are key components of the RL algorithms we use.

2.1.1 Value Iteration

Value iteration is a dynamic programming method that optimizes decisions in reverse order. By unfolding the top-down recursion of value functions and generalizing we can apply Bellman's equation [7]:

$$V(s_t) = \max_{a_t} R(s_t, a_t) + \gamma \sum_{s_{t+1}} P(s_{t+1}|s_t, a_t)V(s_{t+1}) \tag{2.3}$$

The algorithm works as follows:

Algorithm 1: Value Iteration

$V_0^* \leftarrow \max_a R(s, a)\ \forall s$
for $t = 1$ *to* h **do**
  $V_t^*(s) \leftarrow \max_a R(s, a) + \gamma \sum_{s'} P(s'|s, a)V_{t-1}^*(s')\ \forall s$
end
Return V^*

A policy can be induced by performing one more iteration and instead noting the actions that maximize the value. With an infinite time horizon, value iteration is guaranteed to converge to the optimal value [54]. In practice, value iteration is often terminated once the value function begins changing infinitesimally.

2.1.2 Policy Iteration

Instead of inducing a policy from an optimized value function, this technique directly optimizes the policy. Intuitively, instead of optimizing a value function and then inducing a policy, we can alternate these actions in order to generate a continuously-improving policy.

The algorithm works as follows:

Algorithm 2: Policy Iteration

$\pi \leftarrow$ any random policy
prev-policy $= \pi$
while *True* **do**
  $V^{\pi}(s) \leftarrow R(s, \pi(s)) + \gamma \sum_{s'} P(s'|s, \pi(s)) V^{\pi}(s') \ \forall s$
  $\pi(s) \leftarrow \arg\max_a R(s, a) + \gamma \sum_{s'} P(s'|s, a) V^{\pi}(s') \ \forall s$
  if $\pi ==$ prev-policy **then**
    break
  end
  prev-policy $\leftarrow \pi$
end
Return π

Policy iteration is guaranteed to converge to the optimal value function and policy in a finite number of iterations assuming S and A are finite [59].

2.2 Reinforcement Learning Overview

Markov Decision Processes assume full knowledge of the environment's transition probabilities (P) and reward structure (R). In reality, this level of information about an environment is rarely known beforehand, and must be learned/approximated from experience. This is exactly what reinforcement learning techniques aim to do, by building upon the framework of MDPs. In fact, model-based RL learn a model during execution so the previously-discussed techniques can be applied. In contrast, model-free RL techniques learn the optimal policy directly without learning the model. The high-level structure of a reinforcement learning process is depicted in Figure 2.1.

Here we present an introduction to the fundamentals behind three prominent model-free techniques: Q-learning, policy gradients, and actor-critic. We should note that though we discuss the basic intuition behind each technique, there are a number of variants and optimizations that are situationally applied. Most prominent among these tweaks is the use of neural networks to create estimates of policies as well as value functions. This combination of neural networks and RL techniques is known as deep reinforcement learning (deep RL).

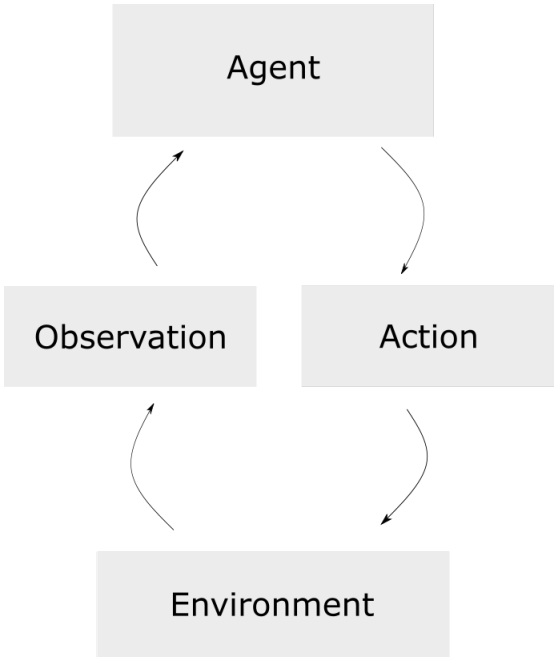


Figure 2.1: A simple single-agent reinforcement learning system.

2.2.1 Q-learning

Q-learning is a well-known example of a model-free RL algorithm. By rewriting (2.2) with one-sample estimations, we can obtain an approximation of the Q function based on reward r as follows:

$$Q^*(s,a) \approx r + \gamma \max_{a'} Q^*(s',a') \tag{2.4}$$

We can then incrementally update an estimate of the Q-function by applying temporal difference error [67]:

$$Q^*_n(s,a) = Q^*_{n-1}(s,a) + \alpha \left(r + \gamma \max_{a'} Q^*_{n-1}(s',a') - Q_{n-1} * (s,a) \right) \tag{2.5}$$

By repeating (2.5) with new actions until $Q*$ converges, we obtain the algorithm for Q-learning. A converged Q-function allows the agent to know the best action to take from

each state, which encodes a policy in and of itself.

Algorithm 3: Q-Learning

while *True* **do**
  Select and execute a
  Observe s' and r from environment
  Update counts: $n(s,a) \leftarrow n(s,a)+1$
  Learning rate: $\alpha \leftarrow 1/n(s,a)$
  $Q^*(s,a) = Q^*(s,a) + \alpha(r + \gamma \max_{a'} Q^*(s',a') - Q*(s,a))$
  if Q^* *converge* **then**
    break
  end
  $s \leftarrow s'$
end
Return $Q*$

While Q-learning is traditionally carried out by computing tables/vectors of Q-values to store mappings from state, action pairs to Q-values, this becomes unsustainable as the size of the state and action space rises. This is tackled by approximating the Q-function rather than storing exact values, usually through the use of neural networks as they are universal function approximators [40]. Q-learning with neural networks is known as Deep-Q Networks (DQN), and while some additional techniques such as experience replay (remembering and reusing past experiences [61]) and target networks (using a separate network for a batch of updates before synchronizing) are necessitated to improve stability the intuition of the algorithm remains the same.

2.2.2 Proximal Policy Optimization

While Q-learning creates an estimate of the value function, policy gradient methods work directly to optimize the policy. This is analogous to the behaviour of value iteration and policy iteration respectively for Markov Decision Processes.

Consider a stochastic policy $\pi_\theta(a|s) = Pr(a|s;\theta)$ characterized by θ. Policy gradient methods work by computing an estimate of the policy gradient and then plugging it into a stochastic gradient ascent algorithm. The most-commonly used gradient estimator has the form:

$$\hat{g} = \hat{E}_t[\nabla_\theta \log \pi_\theta(a_t|s_t) A_t] \tag{2.6}$$

Here, the expectation $\hat{E}_t$ indicates the empirical average over a finite batch of samples, in an algorithm that alternates between sampling and optimization [63]. We also see the advantage function used here which is defined as $A(s,a) = Q(s,a) - V^{\pi}(s)$, capturing the impact of a single action a on the value function.

Recently, a new family of policy gradient methods has emerged known as Proximal Policy Optimization (PPO). These methods follow the same fundamentals, but use a tweaked 'surrogate' objective function to enable multiple epochs of minibatch updates rather than being confined to one gradient update per data sample. The algorithm outline is reproduced from [63] below:

Algorithm 4: PPO, Actor-Critic Style

for *iteration = 1,2, ...* **do**
  for *actor = 1,2, ...* **do**
    Run policy $\pi_{\theta_{old}}$ in environment for T timesteps
    Compute advantage (Q-function) estimates $\hat{A}_1, ..., \hat{A}_T$
  end
  Optimize surrogate L wrt θ, with K epochs and minibatch size $M \leq NT$
  $\theta_{old} \leftarrow \theta$
end
Return π_θ

Every iteration, each of N (parallel) actors collect T timesteps of data. We then construct the surrogate loss on these NT timesteps of data, and optimize it with (minibatch) stochastic gradient descent or Adam [30], for K epochs [63]. Note that multiple actors is not the same as multiple agents; PPO is a parallel algorithm where each actor oversees a separate instance of the environment. It is common to parametrize PPO with neural networks [38], similar to Q-learning.

2.2.3 Actor-Critic

Actor-Critic is an RL technique that uses value functions estimations as seen before to compute an advantage function, which is used to evaluate the current policy and iteratively improve it by moving in the direction of the gradient. The algorithm for Advantage Actor-Critic (A2C) that takes in an initial state s, initial stochastic policy π_θ, and horizon h is

as follows:

Algorithm 5: A2C(s, π_θ, h)

Initialize π_θ to anything
for *each episode* **do**
  Initialize s_0 and set $n \leftarrow 0$
  while *s is not terminal or $n \neq h$* **do**
    Select a_n
    Execute a_n, observe s_{n+1}, r_n
    $\delta \leftarrow r_n + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}) - Q_w(s_n, a_n)$
    $A(s_n, a_n) \leftarrow r_n + \gamma \max_{a_{n+1}} Q(s_{n+1}, a_{n+1}) - \sum_a \pi_\theta(s|s_n) Q(s_n, a)$
    Update $Q : w \leftarrow w + \alpha_w \gamma_n \delta \nabla_w Q_w(s_n, a_n)$
    Update $\pi : \theta \leftarrow \theta + \alpha_\theta \gamma^n A(s_n, a_n) \nabla \log_{\pi_\theta}(a_n|s_n)$
    $n \leftarrow n + 1$
  end
end

Note that this algorithm combines a variety of concepts introduced in previous sections, including temporal difference error from (2.5). Also note that this is the first algorithm presented to explicitly represent the value function and policy simultaneously. The 'actor' (policy) learns by using feedback from the 'critic' (value function). In doing so, these methods trade off variance reduction of policy gradients with bias introduction from value function methods [5, 14].

Recently an asynchronous method of actor-critic has been proposed, known as asynchronous advantage actor-critic (A3C) [44], which asynchronously executes multiple agents in parallel on multiple instances of the environment. This has a similar stabilizing effect as experience replay while speeding up processing. The Q-function is often represented using a Deep Q-Network [45].

2.3 Multi-Agent Reinforcement Learning

The previous section considered a single learning agent in a stationary environment. In multi-agent RL (MARL) multiple reinforcement learning agents are at play simultaneously within the environment. From the perspective of a single agent, the other agents are usually considered to be simply part of the environment, but their continually-changing behaviour renders the environment non-stationary [11].

Formally, consider problems where observations and actions are distributed across n agents with an action space of A and observation space of O. We can denote each agent's reward function as $[R_1, R_2, ..., R_n]$. The state transitions are now n-dimensional vectors; observations $[o_1, o_2, ..., o_n], o \in O$ lead to actions $[a_1, a_2, ..., a_n], a \in A$ with rewards $[R_1(o_1, a_1), R_2(o_2, a_2), ..., R_n(o_n, a_n)]$. If $R_1 = R_2 = R_n$, then the environment is purely cooperative as all agents have the same goal (to maximise the same expected return). Alternatively, if $n = 2$ and $R_1 = -R_2$, then the two agents have opposite goals and the environment is purely competitive. Mixed cooperative-competitive environments exist somewhere between the spectrum of these two extremes. Generally, the environment is modelled as a Markov game where actions are chosen and executed simultaneously, and new observations are perceived simultaneously as a result of a transition to a new state [66]. This is depicted in Figure 2.2.

There are ancillary benefits to MARL beyond the simple speedup offered by parallel computation. For instance, agents can exhibit specialization [22] and experience sharing [68]. Specialized examples of the latter include skilled agents acting as teachers [13] or role models [53] for weaker agents. However, the exponential growth of the state-action space means that deep RL techniques are almost compulsory. Each agent is also faced with a moving-target problem; the best policy for an individual agent changes as the policies of other agents change [11]. In fact, due to the complexities of the environment and the combinatorial nature of the problem, most MARL problems are categorized as NP-hard problems [9].

2.3.1 Cooperation

Except for the extremely rare case where the environment is purely competitive, agents must learn to work together in some fashion, if only to avoid wasteful conflict. While some cooperation will naturally emerge as agents learn to maximize their expected discounted reward, the choice of reward function and algorithm can have a large effect on the degree and form of cooperation induced.

One naive approach to training cooperative MARL problems is to perform the training with a centralized controller, essentially converting the domain to a single-agent RL problem where the controller outputs the joint action of all agents at each timestep. This would ensure cooperation among the agents. However, this approach causes the number of actions to exponentially increase and makes the problem intractable [48]. Also, in partially-observable environments such as the one we use in this thesis, each agent only has a small snapshot of the entire state, which further complicates the implementation of such

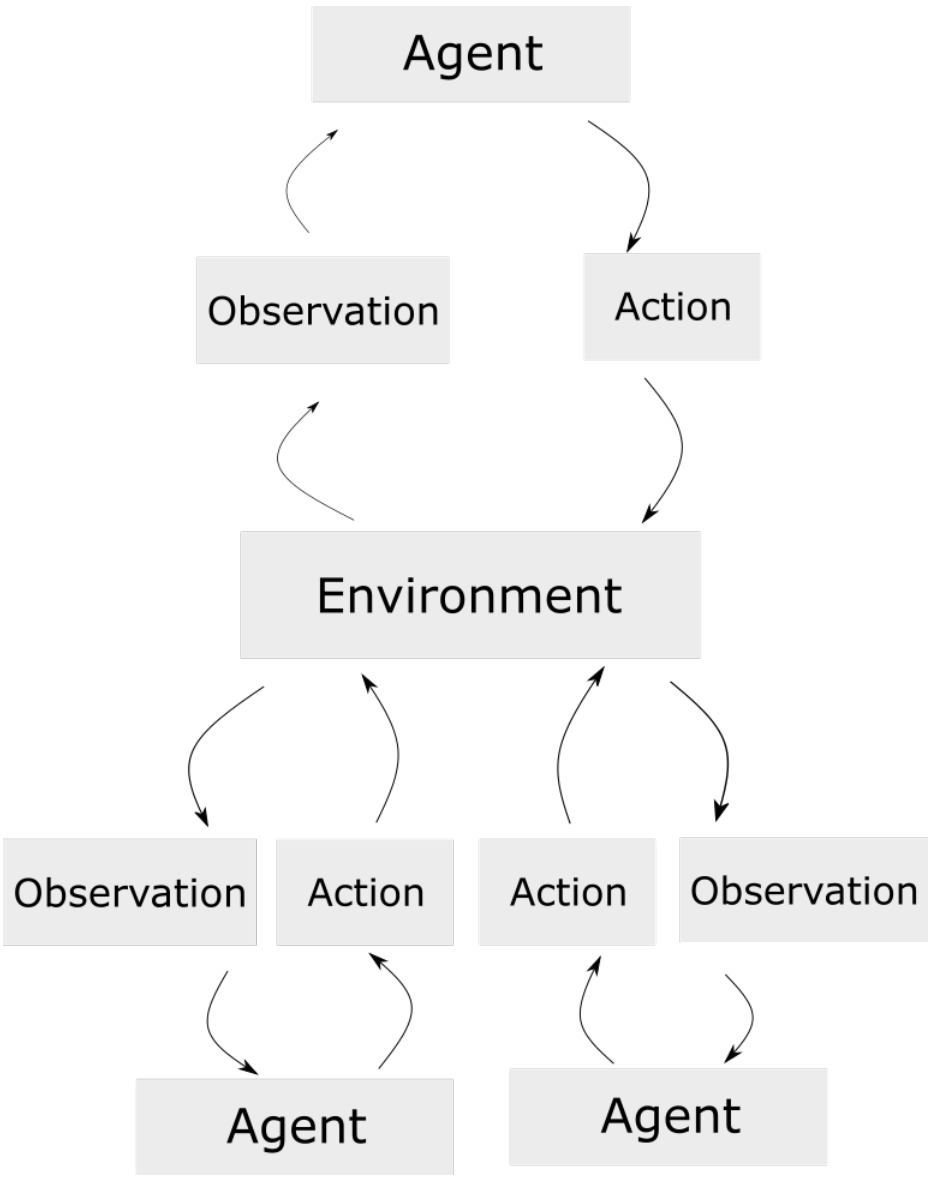


Figure 2.2: An example of a multi-agent reinforcement learning system with 3 agents. Actions are taken simultaneously, which triggers simultaneous experience of observations.

a central controller. However, there has been some success with using centralized critics as seen in [60].

Another intuitive way to go about encouraging cooperation is to give all agents the same reward function, thus making the environment purely cooperative. In other words, agents are trained on the joint reward of all agents. Unfortunately, this leads to the 'lazy agent' problem, where weaker agents will take a backseat to let higher-performing agents take over [66]. For example, imagine training a 2-player soccer team using RL with the number of goals serving as the team reward signal. Suppose one player agent has become a better scorer than the other. When the worse player takes a shot the outcome is on average much worse, and the weaker player learns to avoid taking shots [24], instead passing to the stronger player. By doing this, the weaker player stymies his own learning and prevents the players from ever learning how to properly cooperate to score together.

Clearly, the reward function must be more sophisticated than simple team reward. This was formalized by [65] as the optimal rewards problem (ORP) which was later extended to multiagent settings. Liu *et al.* [37] proposed multiagent ORP as the optimal set of individual reward functions that guide the team of agents to joint-behaviour and in expectation maximize joint objective utility. While there has been recent work pertaining to generating [16] or learning [37] optimal individual reward functions, we focus instead on approaches that allow prior encoding of *a priori* domain knowledge into reward functions.

One approach for modifying reward functions is reward-shaping, where the reward function is subtly modified so as to guide agents on correct paths without affecting the optimal policy [47, 31]. Guaranteeing such a property can be difficult and requires a significant amount of domain knowledge [18]. Regardless, there has been a growing body of work using reward-shaping to improve cooperation [15, 17, 42].

More recently, intrinsic rewards based on psychological motivations such as curiosity and exploration that are added to the given reward as bonuses have been shown to improve the performance of computational agents [37]. Intrinsic rewards are defined in psychology as being moved to do something because it is inherently enjoyable rather than a specific rewarding outcome [12]. Compared to reward-shaping, intrinsic rewards have not been validated on large-scale real-world applications, but are more generalizable as they can lead to the emergence of cooperation without making strong assumptions about 'correct paths' [18]. This generalizability was demonstrated by [18], who proposed task-independent reward-functions for their classifications of maintenance, approach, avoidance, and achievement problems. Iqbal *et al.* [27] were directly able to show a link between improved cooperation and different forms of intrinsic rewards while making a case for the strength of heterogeneous multi-agent architectures, where agents are outfitted with different intrinsic

reward functions.

2.3.2 Communication

One way to improve performance and coordination, especially in partially-observable environments, is to let the agents communicate with each other. Allowing the agents to share messages and learn to communicate can significantly improve their adaptability to a given situation [58]. To allow information to circulate between agents, a variety of communication channels and protocols have been explored. Communication channels can be continuous to allow easy optimization or discrete to mimic human language. The communication protocol can either be hand-crafted in advance or be learned by the agents through reward functions.

Communication is a broad term, and can take a variety of forms in multi-agent systems. While this thesis focuses specifically on communication in the form of discrete hand-coded messages or signals, multi-agent communication often gets obfuscated with semi-shared learning. For example, in [8] agents 'communicate' a shared policy that is learned together. We also distinguish between cheap-talk communication and cost-associated communication. The experiments in this thesis use cheap-talk channels but works such as [23] have experimented with predefined costs for exchanging information in MARL settings.

While communication protocols often bolster coordination and cooperation between agents, it is important to note that these protocols significantly increase the search space of the problem, both by increasing the size of the observation available to the agent (it now knows information coming in from other agents) and by increasing the agent's available choices (as the agent now has to choose environmental actions as well as communication actions) [51]. As noted in [19], this increase in search space can hamper learning optimal behaviours by more than communication itself may help. This is closely tied to the idea of communication vocabulary, as the size of the vocabulary has a huge influence on the performance of communicating agents [58]. As shown in [33], the required size of the vocabulary is correlated with the complexity of the environment.

2.4 Social Dilemmas

Social dilemmas expose tensions between collective and individual rationality [55]. Participants can choose to engage in cooperation, defection, or any combination of the two. Mutual cooperation leads to better outcomes for all than any could obtain individually.

However, the lure of free riding and other such parasitic strategies implies a tragedy of the commons that threatens the stability of any cooperative strategy [1].

Repeated general-sum matrix games provide a mathematical framework for understanding social dilemmas [34] in the context of two-player games. The four possible outcomes of each stage are R (mutual cooperation reward), P (mutual defection punishment), S (sucker outcome obtained by cooperating against a defector), T (temptation outcome obtained by defecting against a cooperator). A two-player repeated general-sum matrix game is a social dilemma when these four payoffs satisfy the following social dilemma inequalities (formulation reproduced from [41]):

1. $R > P$ Mutual cooperation is preferred to mutual defection.
2. $R > S$ Mutual cooperation is preferred to being exploited by a defector.
3. $2R > T + S$ This ensures that mutual cooperator is preferred to an equal probability of unilateral cooperation and defection.
4. either $T > R$ where exploiting a cooperator is preferred over mutual cooperation (greed) or $P > S$ where mutual defection is preferred to being exploited (fear).

While matrix game social dilemmas (MGSD) have successfully served as models for a wide variety of phenomena in theoretical science and biology [34], they often devolve to very simple computational strategies. For example, in the prisoner's dilemma MGSD, players are paired up and can choose to cooperate or defect over a number of rounds. All of the inequalities from 2.4 apply, and agents are motivated to defect out of both greed and fear simultaneously. The best computational solution is Rapoport's Tit-for-Tat [56], which has been victorious in multiple tournaments [2, 3] by simply starting with cooperation and subsequently mimicking the opponent's last move.

2.4.1 Sequential Social Dilemmas

Leibo *et al.* [34] propose several shortcomings of the MGSD model that fail to capture aspects of real-world social dilemmas. They note that realistic cooperation and defection are not simple atomic actions. Rather, a sequence of actions or a policy can be construed as cooperative on a graded scale. Also, these decisions must usually be made with only partial information about the state of the world and the activities of other players. In order to address these shortcomings and create more computationally-interesting simulations of

realistic social interactions, [34] introduced temporally-extended versions of MGSDs known as sequential social dilemmas (SSDs), which is the class of problems we work with in this thesis. The above inequalities still hold, but the payoffs now apply to policies rather than individual moves.

Two examples of SSDs are described below. Note that owing to the greater complexity arising from their sequential structure, it is more computationally demanding to find equilibria of SSD models, necessitating the use of sophisticated multiagent deep reinforcement learning methods [34]. In subsequent works SSDs have become a popular benchmark for multiagent cooperation. All agents would benefit from learning to cooperate in these games, because even agents that are being exploited get higher reward than in the regime where more agents defect [28]. However, traditional RL agents struggle to learn to coordinate or cooperate to solve these tasks effectively [26]. Agents must learn what cooperation and defection entails in a given environment while also learning to abstain from the latter.

Despite being proposed fairly recently, a handful of approaches have been proposed for solving SSDs with MARL, mostly focusing on various psychologically-motivated intrinsic rewards that promote cooperation. Hughes *et al.* [26] proposed a penalty for inequity aversion, where overperforming and underperforming agents are given a negative intrinsic reward, framed as 'guilt' and 'envy' respectively. This essentially punishes overly-cooperative and overly-defective behaviour and helps bring the population to an equilibrium. Note that this approach requires agents to have some knowledge of the rewards of other agents, which might not always be feasible.

Another approach proposed by [20] assigns intrinsic rewards based on reciprocity similar to tit-for-tat. Their setup consists of 'innovator' agents and 'imitators;' they show that the presence of reciprocating imitators push the innovator towards cooperation as the leaders quickly learn that defection will be punished by reciprocal defection. This approach requires the imitators to have complete access to the states and actions of the innovator.

Lastly, [28] demonstrate the first truly-decentralized solution to SSDs by assigning intrinsic rewards based on causal influence; agents are rewarded for disrupting the action distributions of neighbouring agents. By combining this with a model-of-other-agents (MOA) that allows agents to make counterfactual predictions about each other, they achieve state-of-the-art scores on the Harvest and Cleanup SSDs. Jaques *et al.* [28] also conduct an ancillary experiment where they explore combining causal influence intrinsic rewards with a cheap-talk communication channel, where agents are rewarded for sending symbols that maximise disruption, though they found this was not as powerful as the more-complex MOA model.

Harvest

Harvest is a partially-observable public pool resource SSD game introduced in [52]. The goal of the game is to collect apples, which provide a reward of +1. However, harvested apples regrow at a rate proportional to the number of remaining nearby apples. If all apples in a local area, which we refer to as an 'apple patch', are harvested none of them will ever regrow. The episode ends after 1000 timesteps, after which the map resets to its initial state.

This is a tragedy of the commons dilemma in which an individual is tempted by a personal benefit to deplete a resource that is shared by all. Selfish individuals want to harvest as rapidly as possible while the group benefits when individuals abstain from this behaviour, especially around nearly-depleted patches. Agents can also tag each other with a 'penalty beam.' Any agent caught in the path of the beam receives a hefty -50 reward penalty, but the beam costs -1 reward and a wasted action to fire. Earlier works remove victim agents from the game for a number of rounds instead of penalizing their reward, but we adopt the penalty approach used in more recent publications. Figure 2.3 depicts the relative payoff for a defector given a fixed number of other cooperators, known as a Schelling diagram [62, 52]. From it we can see that increased cooperation is a preferred outcome even from the perspective of a defector.

Cleanup

Cleanup is a partially-observable public goods SSD game where the goal is to collect apples which provide a reward of +1. Apples spawn at a rate determined by the state of a geographically separated river. Over time, this river fills with waste, linearly lowering the spawn rate of apples. At a high-enough level of waste, no more apples can spawn. The episode starts with the level of waste slightly above this threshold. The agents can take actions to clean the waste by firing a 'cleaning beam' when near the river, which provides no reward but is necessary to generate any apples. The episode ends after 1000 steps, after which the map is reset to its initial state.

This is a public goods dilemma, where an individual must pay a personal cost in order to provide a resource that is shared by all. If some agents are contributing to the public good by clearing waste from the river, there is an incentive to stay in the apple-spawning region and reap the benefits of their hard work. However, if all players adopt this strategy no further apples will spawn. Similar to Harvest, agents can take a -1 reward to fire a 'penalty beam' that gives a hefty -50 reward fine to any agent caught in its path. The

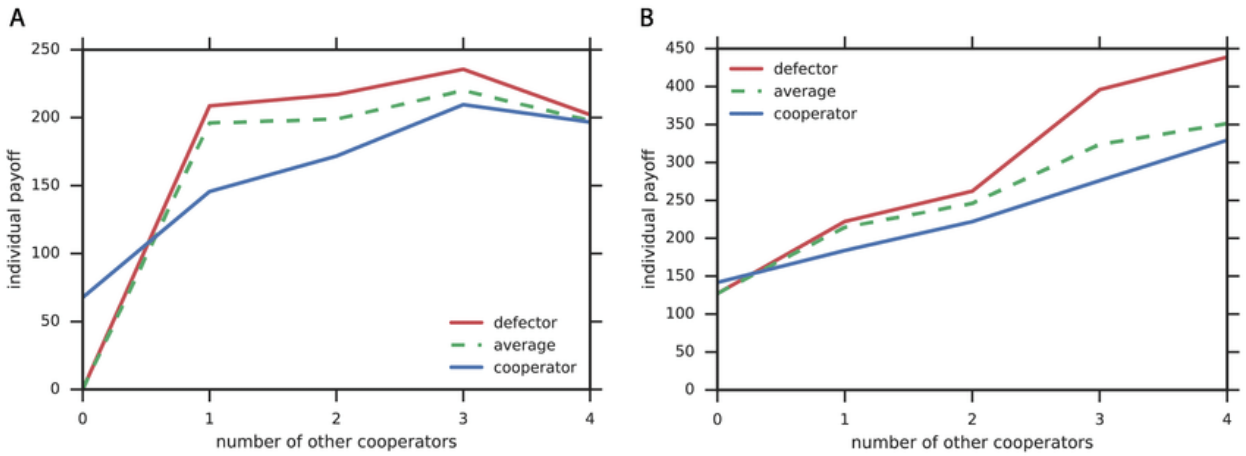


Figure 2.3: Reproduced from [26]. (A) shows the Schelling diagram for Cleanup, (B) shows the Schelling diagram for Harvest. The dotted line shows the overall average return if the individual were to choose defection.

Schelling diagram in Figure 2.3 validates that in this SSD all agents benefit from learning to cooperate, even when an individual agent is defecting.

Chapter 3

Methodology

We iterate upon the ideas of [28] while moving away from their causal influence model to focus on what we denote as obedience-based learning. We augment open-source implementations [69] of the Harvest and Cleanup games detailed in Section 2.4.1 by embedding an all-to-all communication channel into the game environments similar to the approach used by [21].

The neural network model of each agent (based upon the architecture of [28]) is augmented with an additional output head responsible for learning a communication policy π_c separate from the standard environmental policy π_e as seen in Figure 3.1.

If we define c_{ij}^t as the communication symbol sent from agent i to agent j at timestep t, the result of these augmentations to both the environment and agent models means that, at every timestep, agent i is responsible not only for choosing its next game-related action a_i^t, but also a vector $[c_{i1}^t, c_{i2}^t, ..., c_{ii}^t, ..., c_{in}^t]$ of communication symbols where n is the total number of agents in the game. In other words, the action space of agent i changes from simply $[a_i^t]$ to $[a_i^t, [c_{i1}^t, ..., c_{in}^t]]$. Note that this is an all-to-all communication channel where an agent sends a message to every other agent including itself. In a similar way, while the observation space is normally the small portion of the game board visible to the agent (represented as an image), the agent's observations now also include a vector of symbols that were sent to it in the last timestep, i.e $[c_{1i}^{t-1}, ..., c_{ni}^{t-1}]$. This is represented visually in Figure 3.2. In summary, at every timestep agents exchange communication symbols with one another, which can be used to inform their decisions in the next timestep. While this is a large increase in the action and observation space, we hypothesize that the extra information and the intrinsic reward scheme discussed below will prove beneficial to agent cooperation overall.

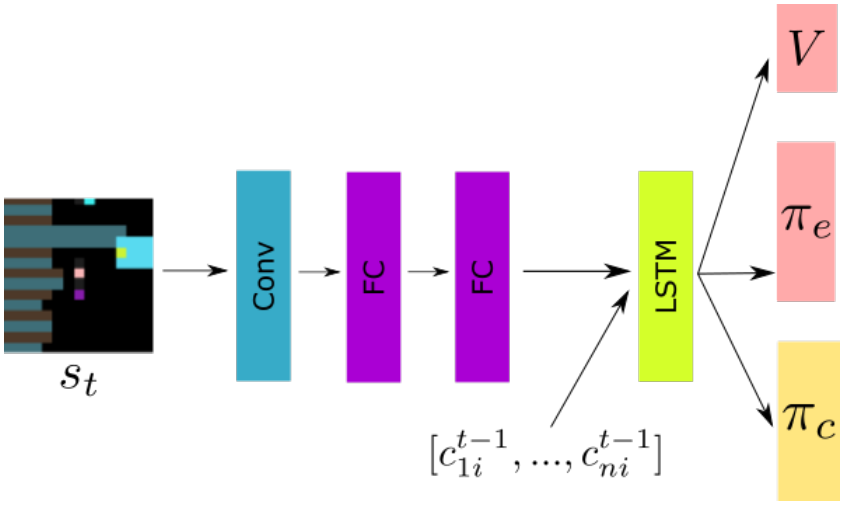


Figure 3.1: An agent's view of the environment (represented as an image) is processed through a single-layer convolutional neural network (Conv) [49]. The output is flattened and connected to two fully-connected layers (FC) before being concatenated with communication symbols to be fed into a long-short-term memory network (LSTM) [25]. Additional details can be found in Appendix A.

There are a handful of actions agents playing the Harvest and Cleanup games can undertake, including moving in the four cardinal directions, rotating their orientation, and firing beams. Apples are automatically collected when an agent occupies the same space.In both games agents can fire penalty beams that 'fine' other agents, whereas the Cleanup game also allows for the firing of a cleaning beam that cleans waste. In our communication vocabulary, we use a unique numeric symbol for each possible action. For example, a symbol of 5 might correspond to a move-left command while a 6 symbol corresponds to move-right. The symbol 0 is reserved as a special value in our vocabulary which we define as 'no-communication.' We set self-communication and communication sent to agents out-of-view to this special value, artificially preventing agents from communicating with themselves and out-of-view agents. Agents can also voluntarily send the 'no-communication' symbol through the channel to abstain from communicating.

Obedience-based learning takes the form of intrinsic rewards given to agents based on the symbols they are sending and receiving. First, we review the sources of environmental rewards in these two SSDs:

1. Agents gain an environmental reward of +1 upon collecting an apple.

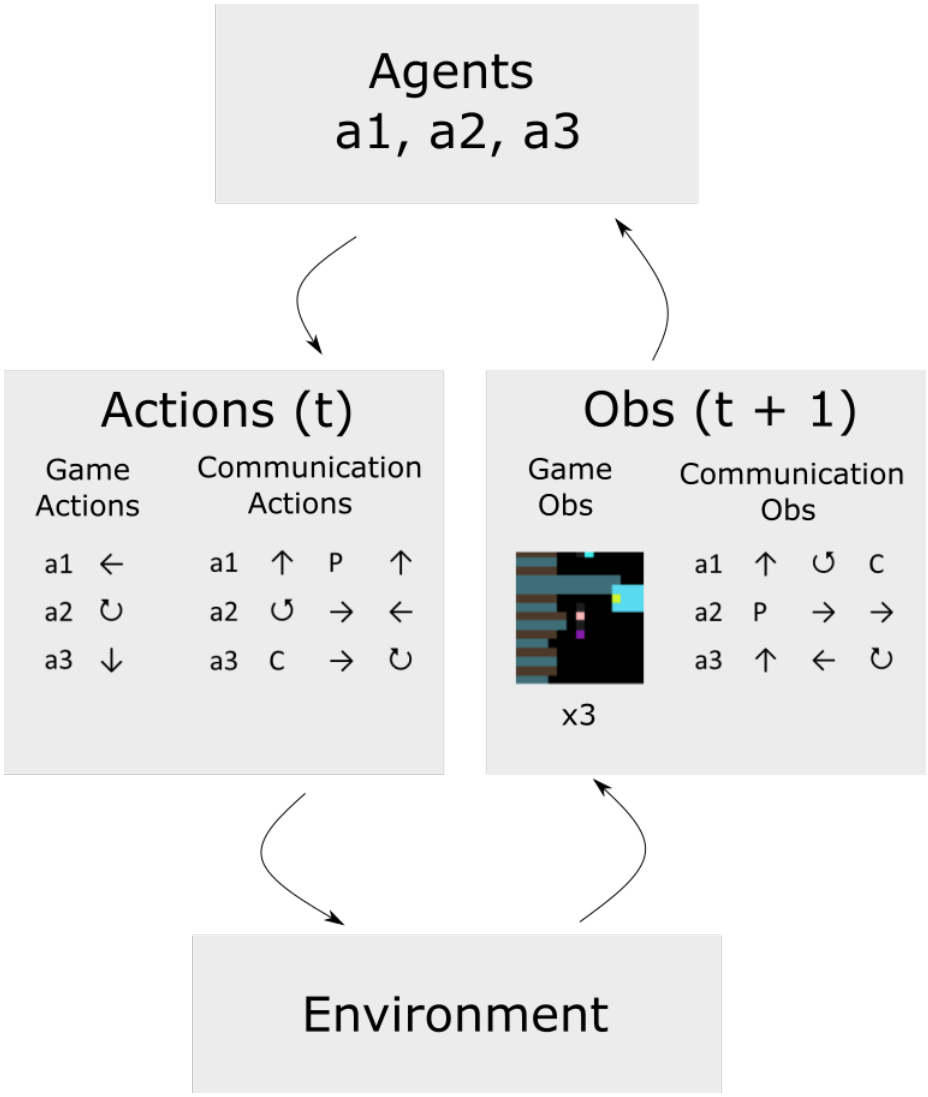


Figure 3.2: A single round of our Cleanup SSD game, with an agent taking an action at time t and getting a $t+1$ observation. Agents (and senders) are rewarded for taking actions that align with observed communication. Note that the communication observation at $t+1$ is the transpose of the communication actions matrix at t. P is a penalty beam command, C a cleaning beam command

2. Agents gain an environmental reward of -1 for firing a penalty beam.

3. Agents receive an environmental reward of -50 for being hit by a penalty beam.

We experiment with adding two sources of intrinsic reward. The first, an obedience reward, is earned when an agent takes an action corresponding to the symbols/commands it received from other agents. As agents receive a distinct command from every other agent, the magnitude of the intrinsic reward scales with the number of commands that were obeyed. Second, we define a leadership reward, which rewards agents for providing commands in the last timestep that were successfully obeyed in the current timestep. In this work we use a linear scaling between intrinsic reward and the number of commands obeyed/given, but we note the potential of alternative reward functions, such as a logarithmic scaling. The total reward then becomes:

$$r = e + \alpha(\text{commands obeyed}) + \beta(\text{successful commands}) \tag{3.1}$$

Where e is the environmental reward, and α and β are the obedience weight and leadership weight respectively. We grid search over the weights to ensure capturing the full spectrum of possible intrinsic rewards.

More formally, expanding our previous notation, at the start of timestep t agent i has sent symbols $[c_{i1}^{t-1}, ..., c_{in}^{t-1}]$ in the last timestep and received symbols $[c_{1i}^{t-1}, ..., c_{ni}^{t-1}]$ as part of the observation of this timestep. Agent i decides to take action a_i^t while the other agents in the system take actions $[a_1^t, ..., a_n^t]$. Let $O(x, s)$ be the number of occurrences of x in the sequence s and let $L(s_1, s_2)$ be the number of element-wise equivalences in sequences s_1 and s_2. The total reward for agent i can then be defined as:

$$r = e + \alpha(O(a_i^t, [c_{1i}^{t-1}, ..., c_{ni}^{t-1}])) + \beta(L([a_1^t, ..., a_n^t], [c_{i1}^{t-1}, ..., c_{in}^{t-1}])) \tag{3.2}$$

We conduct our reinforcement learning experiments using the latest (0.8.2 as of publication) version of Ray [46], a recent framework for building and running distributed applications which comes bundled with the scalable hyperparameter tuning library Tune [36] and scalable reinforcement learning library RLlib [35]. We note that the only existing openly-available implementation of SSD environments [69] was based on an outdated version of Ray, and we provide an updated open-source implementation [1] of these environments with our communication framework available as an optional augmentation.

[1] https://github.com/gauravg11/sequential_social_dilemma_games

Chapter 4

Experiments

In this chapter we describe our investigations into obedience-based learning applied to sequential social dilemmas, starting with the results of our individual experiments before concluding with a summary.

We conducted a variety of experiments to validate our hypothesis regarding obedience-based learning. All experiments follow the game description in Section 2.4.1 and obedience modifications in Chapter 3 unless noted. Additionally, we conduct our simulations with number of agents $n = 5$ and discount factor $\gamma = .99$ as in [28]. Further implementation details can be found in Appendix A.

4.1 Hyperparameter Search

After some initial investigation we chose to use A3C for our analysis, as we found it to be the best-performing algorithm both in terms of reward and performance (experiments detailed in Appendix B).

Next, we conducted randomly-sampled grid search over a number of hyperparameters, most notably obedience weight α and leadership weight β from Equation 3.1. Grid search was used to identify the appropriate order of magnitude for each hyperparameter, before random sampling from a uniform distribution within the chosen order of magnitude to fine-tune the value (details in Appendix B).

We found strong initial evidence for a beneficial effect from obedience-based intrinsic reward, with the best-performing trials using $\alpha = .001, \beta = .001$ for Cleanup and

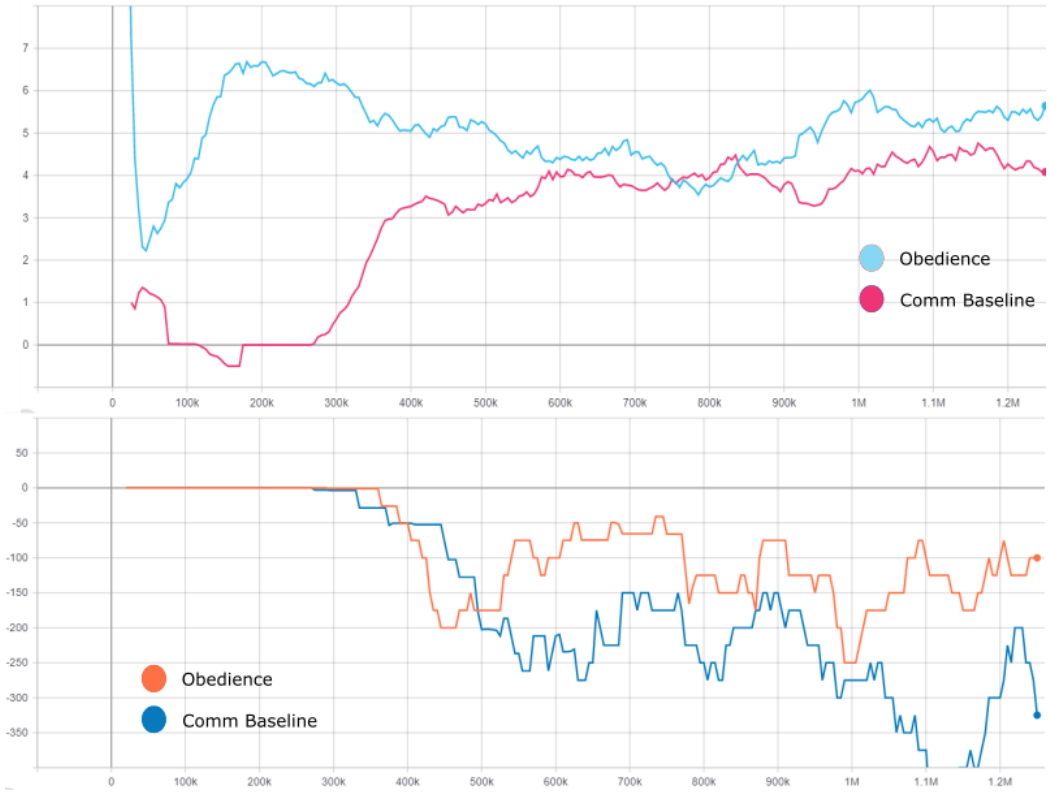


Figure 4.1: Mean **Environmental** Reward per episode of agents playing Harvest (above) and Cleanup (below). Obedience agents are using best-identified intrinsic reward, while baseline agents can communicate but gain no intrinsic reward.

$\alpha = .01, \beta = .001$ for Harvest. This can be seen in Figure 4.1, where we compare the best-performing trials against a baseline of $\alpha = 0, \beta = 0$ that has the communication architecture intact. Clearly, the addition of obedience-based learning is leading to faster emergence of coordination between agents, though the Cleanup agents are still struggling to gain positive reward. Note that Figure 4.1 uses environmental reward alone on its y-axis, intrinsic reward is not considered. We continue to use the hyperparamaters identified here for our subsequent experiments.

4.2 Obedience-Based Learning

We repeat our previous simulation with the identified hyperparameters over a much longer training period; 2500 training iterations of ≈ 5 episodes per iteration. We also evaluate an additional baseline where the agents are prevented from communicating in order to isolate the benefits of communication, but note that by simplifying the problem in this way individual timesteps are no longer directly comparable as a unit of time.

The results of this simulation for Harvest can be seen in Figure 4.2. As expected, we can see from comparing baselines that simply allowing the agents to communicate leads to more effective learning. Obedience-based learning initially struggles but there is a window where it outperforms the communication baseline. Long-term, the baseline agents converge to purely-defective behaviour where the agents greedily grab all the apples in the environment from the start, though this is less prevalent when communication is added. In this instance the addition of obedience seems to be hampering learning, likely due to the greatly increased complexity of the problem space brought about by the added intrinsic reward signal. Interestingly, we can see that the obedient agents converge to a substantially lower value than the communication baseline; agents are either not collecting apples or collecting them too rapidly. This corresponds with an uptick in intrinsic reward, implying that agents have begin to shift their focus towards intrinsic reward over extrinsic reward.

The same setup applied to Cleanup produced similar results, which can be seen in Figure 4.3. Communication baseline agents quickly converge to complete indifference; as they are not able to draw a link between cleaning the river and spawning apples, agents stop exploring moves entirely. Baseline agents instead continuously explore the problem space. The behaviour of obedience-based agents falls between these two extremes, with smaller explorations of the problem space.

Overall, we see some promising preliminary results for obedience-based learning in that it favors the system towards exploration vs exploitation, but also note that the strength of the effect is not sufficient to overcome the challenging temptation of defection for egocentric agents.

4.3 Reduced Vocabulary

Given our previous results, we decided to simplify the complexity of the problem and see if that would encourage further exploration, ideally leading agents to discover the benefits of cooperation.

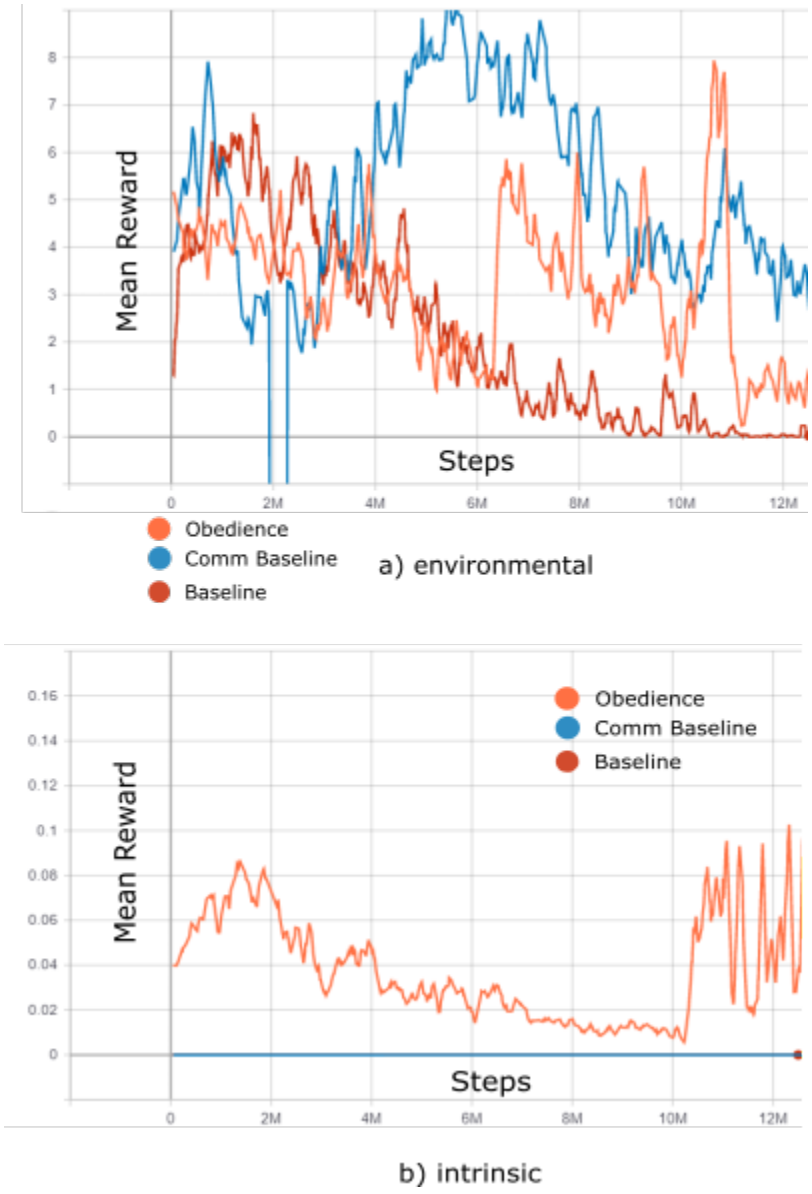
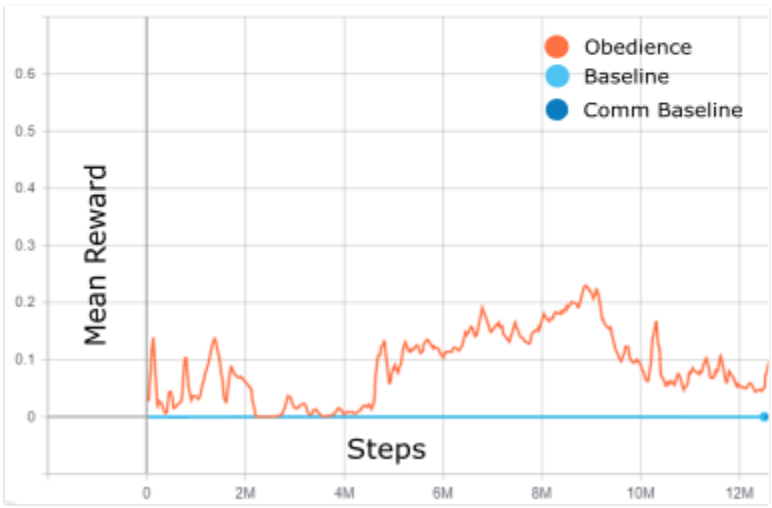


Figure 4.2: Mean Reward per episode of agents playing Harvest. Obedient agents use $\alpha = .01, \beta = .001$ while baseline agents use $\alpha = 0, \beta = 0$. Communication baselines are allowed to communicate. The two baselines overlap in the intrinsic reward portion of this and subsequent figures.

a) environmental



b) intrinsic

Figure 4.3: Mean Reward per episode of agents playing Cleanup. Obedient agents use $\alpha = .001, \beta = .001$ while baseline agents use $\alpha = 0, \beta = 0$. Communication baselines are allowed to communicate.

We carried this out by reducing the size of our communication vocabulary. Intuitively, having a unique communication symbol for each individual action is unnecessary. We modify the communications channel to collapse the vocabulary of symbols to just *move*, *stay*, *rotate*, *penalize*, and *clean*. For example, when an agent now receives a *move* symbol, it receives obedience reward for the move-left, move-right, move-down, and move-up actions. We hypothesized that this modification would drastically reduce complexity without sacrificing the benefits of obedience-based learning. The results of applying this modification can be seen in Figures 4.4 and 4.5.

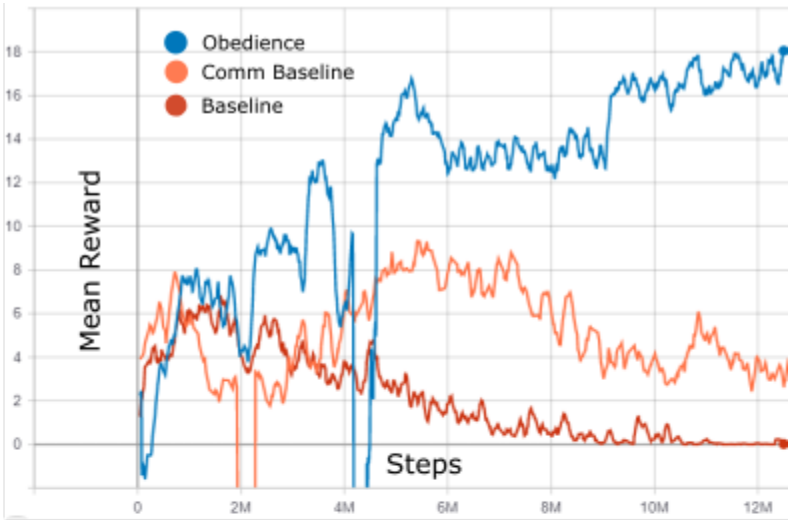
While Cleanup still struggles to achieve positive reward, in Harvest we can now start seeing substantial benefits from obedience-based intrinsic reward. In particular, Harvest obedience with reduced vocabulary greatly outperforms the communication baseline. We also see much higher overall intrinsic reward, indicating that reducing the vocabulary while maintaining semantics has allowed agents to use obedience-based learning more effectively. Clearly, obedience-based learning in Section 4.2 was struggling due to the extremely large problem space. This demonstrates that obedience-based learning is most effective when the communication vocabulary is carefully adjusted relative to the problem domain. This could be done by either encoding *a-priori* domain knowledge as done here or by treating the communication vocabulary as a hyperparameter and searching for optimal values.

4.4 Heterogeneous Agent Population

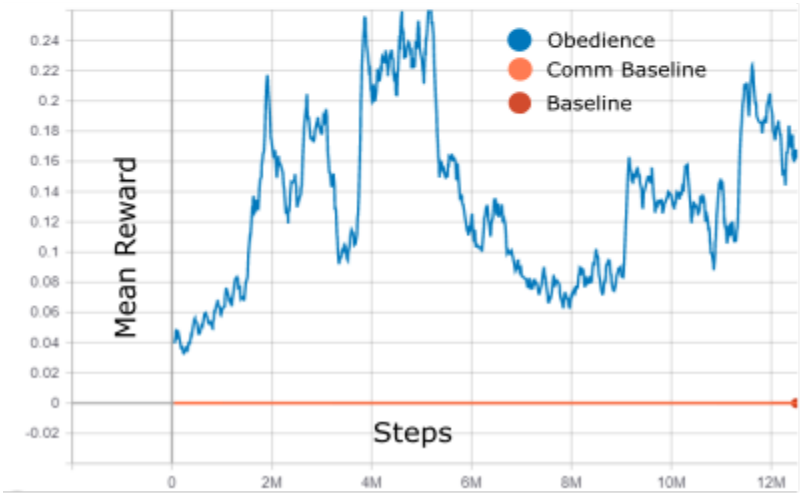
Previous works with SSDs [26, 28, 34] use homogeneous populations of agents in their MARL simulations; all agents are completely identical to each other. Eccles *et al.* [20] use two classes of distinct agents where one class simply mimics the moves of the other. By giving a unique α and β to every agent, we create a truly-heterogeneous agent population in order to gauge the flexibility of obedience-based learning.

We give every agent an index i between $0, 1, ..., n-1$ and then set an individual obedience $\alpha_i = \alpha * i$ and leadership $\beta_i = \beta * i$ weight for each agent in order to model an environment with widely-varying levels of obedience and leadership. In our setup with $n = 5$, this means we have an agent with no obedience ($i = 0$), an agent with standard obedience ($i = 1$), and agents with double ($i = 2$), triple ($i = 3$), and quadruple ($i = 4$) levels of obedience. The results for this setup can be seen in Figures 4.6 and 4.7.

Note that these experiments are not using the reduced vocabulary methods from Section 4.3. While heterogeneous agents perform poorly in Cleanup, with results similar to the no-communication baseline, in Harvest we see obedience agents outperform the communication

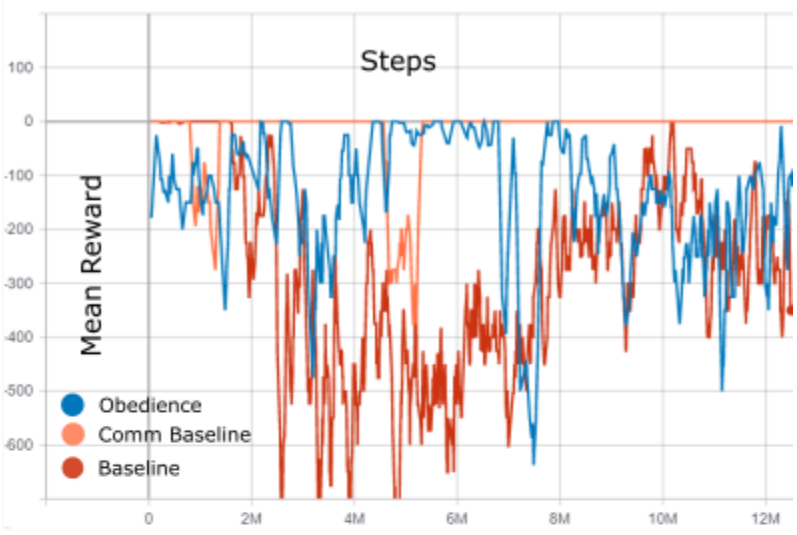


a) environmental

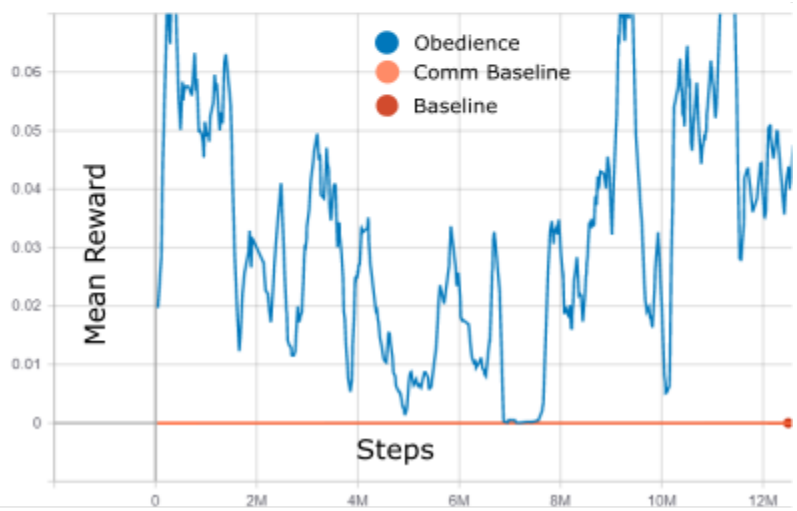


b) intrinsic

Figure 4.4: Mean Reward per episode of agents playing Harvest with a reduced vocabulary.



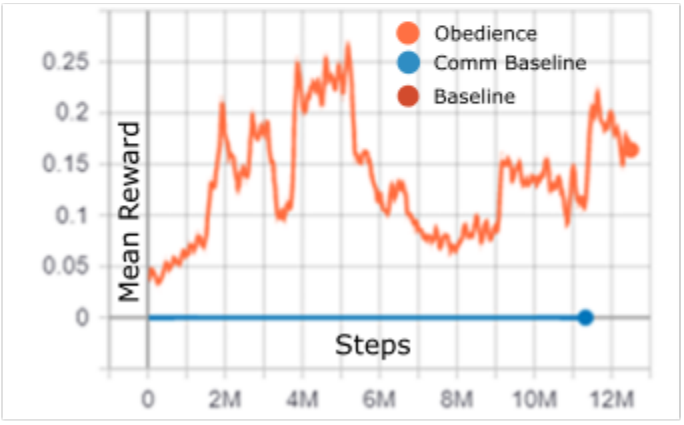
a) environmental



b) intrinsic

Figure 4.5: Mean Reward per episode of agents playing Cleanup with a reduced vocabulary.

a) environmental



b) intrinsic

Figure 4.6: Mean Reward per episode of heterogeneous agents playing Harvest.

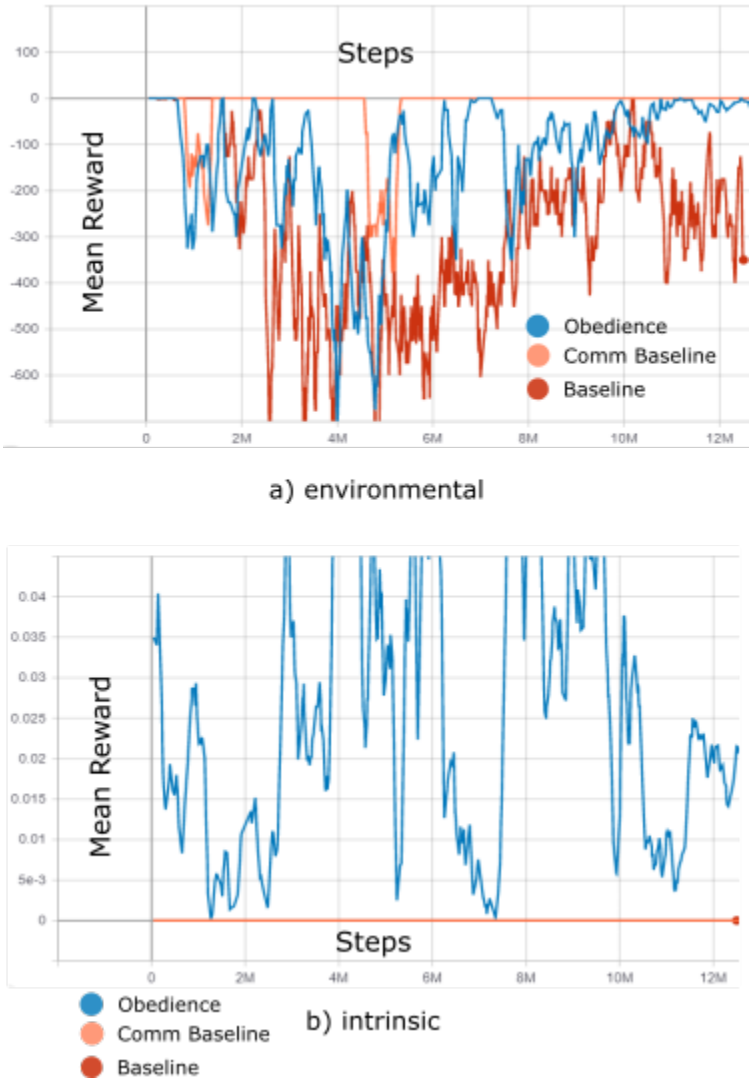


Figure 4.7: Mean Reward per episode of heterogeneous agents playing Cleanup.

baseline. We also see a high level of intrinsic reward, similar to what is seen in Section 4.3. Clearly, when agents learn to properly utilize intrinsic obedience reward the environmental reward also increases, demonstrating the efficacy of this approach. We have also shown that obedience-based learning is not only robust to, but rather can even benefit from, mixed populations.

4.5 Heterogeneous Agents with Reduced Vocabulary

In Sections 4.3 and 4.4, we showed reward gains when applying a reduced vocabulary and heterogeneous agent populations to Harvest. We were curious to see if the benefits from these techniques would be additive when applied together. The application of a heterogeneous agent population to Harvest while using a reduced vocabulary can be seen in Figure 4.8. As these techniques showed no benefits on Cleanup, we did not analyze it in this experiment.

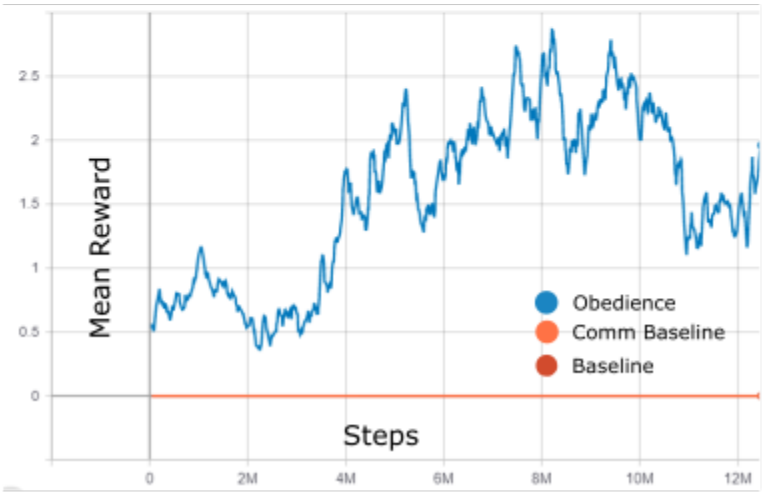
Surprisingly, this caused a complete collapse in environmental reward with a huge surge in intrinsic reward. Using agents with up to $4 \times \alpha$ and $4 \times \beta$ along with a simplified vocabulary caused intrinsic reward to be too easily accessed at too great a magnitude, overpowering the environmental reward signal. This demonstrates one of the pitfalls of obedience-based learning shared by all intrinsic reward systems; the reward scheme needs to be finely-tuned so as not to distract from the problem at hand. We feel this approach has potential given our previous results, but would require substantial tuning to obtain a suitable α and β, and hope to address this in future work.

4.6 Leader-Follower Agents

Lastly, we explored defining explicit 'leaders' and 'followers' in our agent population. This was inspired by Eccles *et al.* [20] where the authors used an analogous 'innovator' and 'imitator' system to apply reciprocity to sequential social dilemmas. However, [20] required a hand-coded 'niceness network' to make the imitators mimic the innovator, and the former required complete knowledge of the reward structure of the latter. We hypothesize that we can create a similar effect by simply tweaking agent's individual α and β values.

We first defined leader agents with $\alpha = 0, \beta = .01$ and follower with agents $\alpha = 0.1, \beta = 0$ so that leaders are rewarded only for successful leadership and followers rewarded only for successful obedience. This led to results that were slightly worse than the communication

a) environmental



b) intrinsic

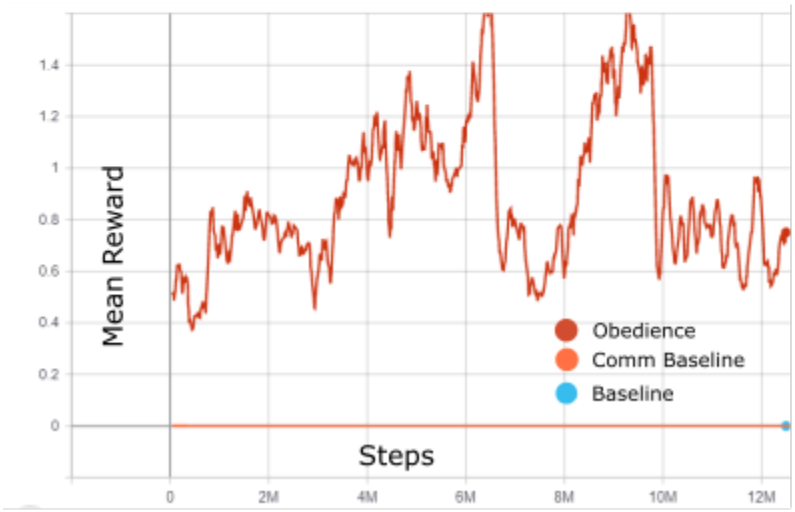
Figure 4.8: Mean Reward per episode of heterogeneous agents with a reduced vocabulary playing Harvest.

baseline. Further investigation showed that both leadership and obedience were necessary within a single agent for successful obedience-based learning. While one force can be stronger than the other, eliminating either cripples learning. We settled upon $\alpha = .001, \beta = .01$ for leaders and $\alpha = .01, \beta = .001$ for followers, the results of which can be seen in Figure 4.9 for Harvest and Figure 4.10 for Cleanup.

Defining explicit leaders and followers leads to significantly more cooperation in Harvest, with some of the highest mean environmental reward values relative to other experiments in this thesis. We also see a significant level of intrinsic reward, though not as high as those seen in Section 4.5, indicating that we've found a healthy trade-off between extrinsic and intrinsic reward. Cleanup continues to struggle to cooperate, though we do note that convergence occurs faster here than in other experiments.

We were interested in how the behaviour of leaders and followers differ while playing Harvest, so we visualized the game during training, an example of which can be seen in Figure 4.11. Over time, the leaders learned to push the followers into a corner and keep them there by continuously sending 'stay' commands. This allowed leaders to collect apples with less competition, causing apple patches to be depleted at a much slower rate. Meanwhile, the followers were able to continuously earn intrinsic reward for obediently staying in a corner. In other words, one group of agents learned to optimize intrinsic reward while the other optimized environmental reward. As agents could only send commands to one another when 'in view,' the leaders also learned not to spend too much time away from the follower's range of view so the latter would not be tempted by environmental reward. These results demonstrate that improved performance solving SSDs is not a guarantee of true collaborative behaviour despite its use as a collaboration benchmark. On the other hand, it could be argued that this is an instance where members of a community are neglecting personal profit for the benefit of others, a form of cooperation otherwise known as altruism. For follow-up work we recommend using additional metrics beyond joint reward to move towards a more formal definition of cooperation.

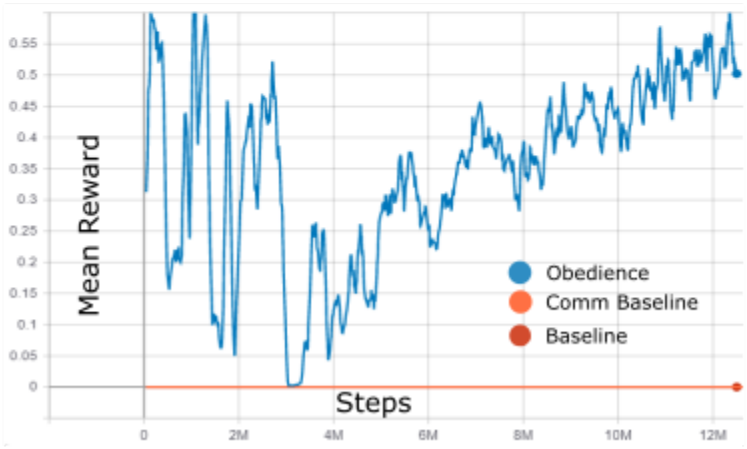
a) environmental



b) intrinsic

Figure 4.9: Mean Reward per episode of explicit leaders and follower agents playing Harvest.

a) environmental



b) intrinsic

Figure 4.10: Mean Reward per episode of explicit leaders and follower agents playing Cleanup.

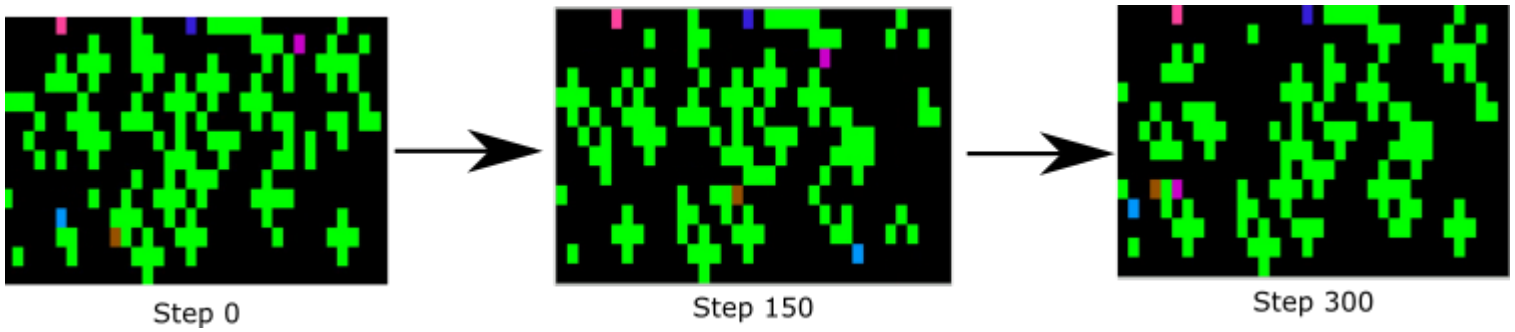


Figure 4.11: Visualization of the Harvest game being played with explicit leaders and followers. The pink and blue agent at the top-left are followers who are being commanded to stay in place, while the remaining leaders consume apples (green) in close proximity to the followers.

Chapter 5

Conclusion

In this work we presented a novel form of psychologically-motivated intrinsic reward denoted obedience-based learning, where agents use a cheap-talk communication channel to exchange commands. We evaluated the potential of the technique using sequential social dilemmas (SSDs) and find promising preliminary evidence of its ability to encourage exploration of the problem space.

While our initial approach struggled to affect collective reward, we found that various heterogeneous populations as well as a reduced communication vocabulary led to overall increases in performance on the Harvest game. Cleanup remains challenging, as it is simply too tempting for defecting agents to take advantage of the hard work of cooperating agents, regardless of the commands being sent their way. We hypothesize that obedience-based learning is more suited for tragedy of the common scenarios rather than public good situations. Validating this would require extending obedience-based learning to further MARL environments as discussed in Section 5.1.

While obedience-based reward is not as powerful as hand-crafted approaches to solve SSDs using MARL (we obtained $\approx$ 600 collective Harvest reward compared to Jaques' 900+), the minimal complexity and decentralization of the approach stands out. Agents do not need to model each other as in [28] or be aware of each other's reward structures as in [26]. Obedience-based learning is a powerful general technique for environments that favor exploration, as the intrinsic reward for obeying commands from a variety of distinct leaders helps encourage agents to continuously 'go out of their comfort zone.'

Though a simple computational problem, solving social dilemmas with multiple egocentric agents remains a challenging problem across the computational and social sciences. We hope that our work here, including up-to-date implementations of Harvest and Cleanup,

will lead to further exploration of psychologically-motivated intrinsic rewards to promote cooperation.

5.1 Further Work

The natural next step would be to evaluate obedience-based learning on other MARL environments. Rather than manually building out a custom modified environment as in this thesis, ideally a general-purpose communication channel could be built out that integrates with OpenAI's Gym [10] environments to allow easy testing of obedience-based learning across a variety of domains. We envision this taking the form of a wrapper[1] module that wraps around MARL environments to modify their reward structure while keeping track of communication.

Based on our results with population heterogeneity, another avenue we would like to explore is applying obedience-based learning to asymmetric games, especially asymmetric social dilemmas [70]. We find this direction especially exciting as 'obedience' takes on a different context when the game becomes asymmetric; perhaps some games simplify to a certain class of agents dictating commands to other classes.

We had two more direct follow-up experiments in mind that we were not able to tackle due to time limitations. Firstly, the agents in our work had a 15×15 view of the environment, it would be interesting to vary the vision-box of the agent. As a corollary, we prevent agents from talking to other agents out-of-view, and relaxing this constraint could lead to interesting results. Secondly, we wanted to explore applying a decay term to the obedience intrinsic reward. Intuitively, decaying intrinsic obedience reward can help the agent favor early exploration while still allowing late exploitation, giving us the best of both worlds.

We would also like to simply repeat our experiments with greater computing power. In particular, our hyperparameter searches were upper-bounded by available computational resources, as MARL simulations are very expensive to simulate, especially with our communication scheme adding multiple extra dimensions to the action and observation space. This is especially noticeable in our experiment combining reduced vocabulary and heterogeneous agents which requires extensive hyperparameter searching.

While our neural network architecture was heavily inspired by the work of [28], there is no guarantee that such an architecture leads to optimal obedience-based learning. While

[1] https://github.com/openai/gym/tree/master/gym/wrappers

we found that our results were resilient to smaller architecture tweaks, we were not able to test a set of substantially different neural network configurations. In the same vein, we fixed the number of agents $n = 5$ and this would be another parameter of our experiment that would be interesting to tune.

Lastly, we use a cheap-talk communication channel where exchanging commands is free, but adding a small cost to communication could reveal interesting insights. In particular, having agents choose between moving and communicating instead of being allowed to do both would simplify implementation while creating an opportunity cost for communication. It would also be of interest to establish a concrete upper-bound for Harvest and Cleanup performance. This could be done through theoretical analysis or by hand-crafting perfectly-cooperative agents that achieve maximum game scores.

References

[1] Bradley Alge and Howard J Klein. Organizational behavior and human decision processes. 1991.

[2] Robert Axelrod. Effective choice in the prisoner's dilemma. *Journal of conflict resolution*, 24(1):3–25, 1980.

[3] Robert Axelrod. More effective choice in the prisoner's dilemma. *Journal of Conflict Resolution*, 24(3):379–403, 1980.

[4] Robert Axelrod and William Donald Hamilton. The evolution of cooperation. *science*, 211(4489):1390–1396, 1981.

[5] Leemon C Baird III. Advantage updating. Technical report, WRIGHT LAB WRIGHT-PATTERSON AFB OH, 1993.

[6] Sean L Barton, Nicholas R Waytowich, Erin Zaroukian, and Derrik E Asher. Measuring collaborative emergent behavior in multi-agent reinforcement learning. In *International Conference on Human Systems Engineering and Design: Future Trends and Applications*, pages 422–427. Springer, 2018.

[7] Richard Bellman. E. 1957. dynamic programming. *Princeton UniversityPress. BellmanDynamic programming1957*, page 151, 1957.

[8] Hamid R Berenji and David Vengerov. Advantages of cooperation between reinforcement learning agents in difficult stochastic problems. In *Ninth IEEE International Conference on Fuzzy Systems. FUZZ-IEEE 2000 (Cat. No. 00CH37063)*, volume 2, pages 871–876. IEEE, 2000.

[9] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

[10] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[11] Lucian Bu, Robert Babu, Bart De Schutter, et al. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

[12] Nuttapong Chentanez, Andrew G Barto, and Satinder P Singh. Intrinsically motivated reinforcement learning. In *Advances in neural information processing systems*, pages 1281–1288, 2005.

[13] Jeffery A Clouse. Learning from an automated training agent. In *Adaptation and learning in multiagent systems*. Citeseer, 1996.

[14] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A survey on policy search for robotics. *Foundations and Trends® in Robotics*, 2(1–2):1–142, 2013.

[15] Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 225–232. International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[16] Sam Devlin and Daniel Kudenko. Plan-based reward shaping for multi-agent reinforcement learning. *The Knowledge Engineering Review*, 31(1):44–58, 2016.

[17] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pages 165–172, 2014.

[18] Paresh Dhakan, Kathryn Merrick, Iñaki Rañó, and Nazmul Siddique. Intrinsic rewards for maintenance, approach, avoidance, and achievement goal types. *Frontiers in neurorobotics*, 12:63, 2018.

[19] Edmund H Durfee, Victor R Lesser, and Daniel D Corkill. Coherent cooperation among communicating problem solvers. *IEEE Transactions on Computers*, 100(11):1275–1291, 1987.

[20] Tom Eccles, Edward Hughes, János Kramár, Steven Wheelwright, and Joel Z Leibo. Learning reciprocity in complex sequential social dilemmas. *arXiv preprint arXiv:1903.08082*, 2019.

[21] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems*, pages 2137–2145, 2016.

[22] Marco Jerome Gasparrini, Ricard Solé, and Martí Sánchez-Fibla. Individual specialization in multi-task environments with multiagent reinforcement learners. In *Artificial Intelligence Research and Development: Proceedings of the 22nd International Conference of the Catalan Association for Artificial Intelligence*, volume 319, page 339. IOS Press, 2019.

[23] Mohammad Ghavamzadeh and Sridhar Mahadevan. Learning to communicate and act using hierarchical reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 3*, pages 1114–1121. IEEE Computer Society, 2004.

[24] Matthew John Hausknecht. *Cooperation and communication in multiagent deep reinforcement learning*. PhD thesis, 2016.

[25] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[26] Edward Hughes, Joel Z Leibo, Matthew Phillips, Karl Tuyls, Edgar Dueñez-Guzman, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin McKee, Raphael Koster, et al. Inequity aversion improves cooperation in intertemporal social dilemmas. In *Advances in neural information processing systems*, pages 3326–3336, 2018.

[27] Shariq Iqbal and Fei Sha. Coordinated exploration via intrinsic rewards for multi-agent reinforcement learning. *arXiv preprint arXiv:1905.12127*, 2019.

[28] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. *arXiv preprint arXiv:1810.08647*, 2018.

[29] Steven J Karau and Kipling D Williams. Social loafing: A meta-analytic review and theoretical integration. *Journal of personality and social psychology*, 65(4):681, 1993.

[30] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[31] Adam Laud and Gerald DeJong. Reinforcement learning and shaping: Encouraging intended behaviors. In *ICML*, pages 355–362, 2002.

[32] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*. Citeseer, 2000.

[33] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[34] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv preprint arXiv:1702.03037*, 2017.

[35] Eric Liang, Richard Liaw, Philipp Moritz, Robert Nishihara, Roy Fox, Ken Goldberg, Joseph E Gonzalez, Michael I Jordan, and Ion Stoica. Rllib: Abstractions for distributed reinforcement learning. *arXiv preprint arXiv:1712.09381*, 2017.

[36] Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E Gonzalez, and Ion Stoica. Tune: A research platform for distributed model selection and training. *arXiv preprint arXiv:1807.05118*, 2018.

[37] Bingyao Liu, Satinder Singh, Richard L Lewis, and Shiyin Qin. Optimal rewards for cooperative agents. *IEEE Transactions on Autonomous Mental Development*, 6(4):286–297, 2014.

[38] Boyi Liu, Qi Cai, Zhuoran Yang, and Zhaoran Wang. Neural proximal/trust region policy optimization attains globally optimal policy. *arXiv preprint arXiv:1906.10306*, 2019.

[39] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pages 6379–6390, 2017.

[40] Zhou Lu, Hongming Pu, Feicheng Wang, Zhiqiang Hu, and Liwei Wang. The expressive power of neural networks: A view from the width. In *Advances in neural information processing systems*, pages 6231–6239, 2017.

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

[42] Patrick Mannion, Jim Duggan, and Enda Howley. Generating multi-agent potential functions using counterfactual estimates. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*, 2016.

[43] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. *The Knowledge Engineering Review*, 27(1):1–31, 2012.

[44] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[46] Philipp Moritz, Robert Nishihara, Stephanie Wang, Alexey Tumanov, Richard Liaw, Eric Liang, Melih Elibol, Zongheng Yang, William Paul, Michael I Jordan, et al. Ray: A distributed framework for emerging {AI} applications. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*, pages 561–577, 2018.

[47] AY Ng, D Harada, and SJ Russell. Policy invariance under reward transformations: Theory and application to reward shaping. icml (pp. 278–287), 1999.

[48] Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

[49] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*, 2015.

[50] Elinor Ostrom. *Governing the commons: The evolution of institutions for collective action*. Cambridge university press, 1990.

[51] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

[52] Julien Perolat, Joel Z Leibo, Vinicius Zambaldi, Charles Beattie, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource

appropriation. In *Advances in Neural Information Processing Systems*, pages 3643–3652, 2017.

[53] Bob Price and Craig Boutilier. Accelerating reinforcement learning through implicit imitation. *Journal of Artificial Intelligence Research*, 19:569–629, 2003.

[54] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990.

[55] Anatol Rapoport. *Game theory as a theory of conflict resolution*, volume 2. Springer Science & Business Media, 2012.

[56] Anatol Rapoport, Albert M Chammah, and Carol J Orwant. *Prisoner's dilemma: A study in conflict and cooperation*, volume 165. University of Michigan press, 1965.

[57] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 2002.

[58] Mohamed Salah Zaïem and Etienne Bennequin. Learning to communicate in multi-agent reinforcement learning: A review. *arXiv*, pages arXiv–1911, 2019.

[59] Manuel S Santos and John Rust. Convergence properties of policy iteration. *SIAM Journal on Control and Optimization*, 42(6):2094–2115, 2004.

[60] Guillaume Sartoretti, Yue Wu, William Paivine, TK Satish Kumar, Sven Koenig, and Howie Choset. Distributed reinforcement learning for multi-robot decentralized collective construction. In *Distributed Autonomous Robotic Systems*, pages 35–49. Springer, 2019.

[61] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.

[62] Thomas C Schelling. Hockey helmets, concealed weapons, and daylight saving: A study of binary choices with externalities. *Journal of Conflict resolution*, 17(3):381–428, 1973.

[63] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[64] Sandip Sen, Mahendra Sekaran, John Hale, et al. Learning to coordinate without sharing information. In *AAAI*, volume 94, pages 426–431, 1994.

[65] Satinder Singh, Richard L Lewis, Andrew G Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2(2):70–82, 2010.

[66] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[67] Richard S Sutton, Andrew G Barto, et al. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge, 1998.

[68] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[69] Eugene Vinitsky. Sequential social dilemma games. https://github.com/eugenevinitsky/sequential_social_dilemma_games, 2019.

[70] Kimberly A Wade-Benzoni, Tetsushi Okumura, Jeanne M Brett, Don A Moore, Ann E Tenbrunsel, and Max H Bazerman. Cognitions and behavior in asymmetric social dilemmas: A comparison of two cultures. *Journal of Applied Psychology*, 87(1):87, 2002.

[71] Chao Yu, Minjie Zhang, and Fenghui Ren. Emotional multiagent reinforcement learning in social dilemmas. In *International Conference on Principles and Practice of Multi-Agent Systems*, pages 372–387. Springer, 2013.

APPENDICES

Appendix A

Implementation Details

A.1 NN Architecture

Inspired by [28], all models were trained with a single convolutional layer with a kernel of size 3, stride of size 1, and 6 output channels. The output is flattened and connected to two fully connected layers of size 32 each. Lastly, this is concatenated with the symbols from the communication channel to be fed into an LSTM of 128 cells. The LSTM has three output heads responsible for the value function, environmental policy, and communication policy respectively. This architecture is visualized in Figure 3.1.

A.2 Hyperparameters

In addition to the obedience weight α and leadership weight β, there are a variety of other hyperparameters that can be independently tuned for each model. We used grid searches to identify the best hyperparameters over 250 training iterations as described in Section 4.1. The below table gives the parameters found to be most effective for each environment. Unfortunately due to limitations on computing power we were not able to re-identify optimal hyperparameters for each experimental condition, so parameters are shared between experimental conditions.

Hyperparameter	Harvest A3C Baseline	Harvest A3C Obedience	Cleanup A3C Baseline	Cleanup A3C Obedience
Entropy reg.	.000687	.000687	.00176	.00176
lr_init	.00136	.00215	.00126	.00126
lr_final	.000028	.000028	.000012	.000013
α	0	.001	0	.01
β	0	.001	0	.001

The learning rate was annealled from an initial value lr_init to lr_final. Other RLlib-specific configurations can be found within the codebase [1].

[1] https://github.com/gauravg11/sequential_social_dilemma_games

Appendix B

Supplementary Experiments

B.1 RL Algorithm Selection

We compared a handful of popular MARL techniques on our SSDs with and without obedience-based learning to identify the most suitable algorithm, both in terms of reward and computational performance.

While we intended to compare DQN, A3C, and PPO, our analysis of DQN was held back by library limitations; as of publication reinforcement learning library RLlib does not yet support the use of complex MultiDiscrete action spaces with their DQN implementation. The results of our investigation can be seen in Figure B.1, where both algorithms were allowed to run on Harvest until mean reward converged.

While both algorithms were eventually able to converge to similar reward values, A3C needed less than a third of the time PPO required. Clearly, obedience-based learning is flexible to the choice of algorithm but the asynchronous nature of A3C puts it ahead in terms of performance.

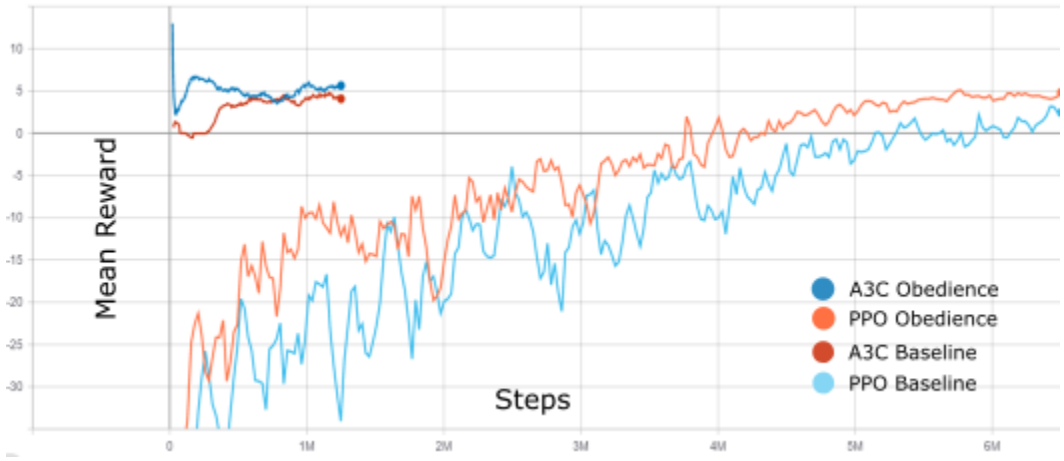


Figure B.1: Mean Reward per episode of agents playing Harvest. Obedient A3C agents use $\alpha = .01, \beta = .001$ while obedience PPO agents use $\alpha = .0001, \beta = .01$. Baseline agents can communicate and use $\alpha = 0, \beta = 0$..