The challenge of hidden gifts in multi-agent reinforcement learning

Dane Malenfant[*] [†] [§] [☿], **Blake A. Richards** [†] [§] [☿] [¶] [‡]
dane.malenfant@mail.mcgill.ca, blake.richards@mila.quebec

[†] **School of Computer Science, McGill University, Canada**
[☿] **Mila - The Québec AI Institute**
[¶] **CIFAR Learning in Machines and Brains**
[‡] **Dept of Neurology & Neurosurgery, and Montreal Neurological Institute, McGill University**

*corresponding author

Abstract

Sometimes we benefit from actions that others have taken even when we are unaware that they took those actions. For example, if your neighbor chooses not to take a parking spot in front of your house when you are not there, you can benefit, even without being aware that they took this action. These "hidden gifts" represent an interesting challenge for multi-agent reinforcement learning (MARL), since assigning credit when the beneficial actions of others are hidden is non-trivial. Here, we study the impact of hidden gifts with a very simple MARL task. In this task, agents in a grid-world environment have individual doors to unlock in order to obtain individual rewards. As well, if all the agents unlock their door the group receives a larger collective reward. However, there is only one key for all of the doors, such that the collective reward can only be obtained when the agents drop the key for others after they use it. Notably, there is nothing to indicate to an agent that the other agents have dropped the key, thus the act of dropping the key for others is a "hidden gift". We show that several different state-of-the-art RL algorithms, including MARL algorithms, fail to learn how to obtain the collective reward in this simple task. Interestingly, we find that independent model-free policy gradient agents can solve the task when we provide them with information about their own action history, but MARL agents still cannot solve the task with action history. Finally, we derive a correction term for these independent agents, inspired by learning aware approaches, which reduces the variance in learning and helps them to converge to collective success more reliably. These results show that credit assignment in multi-agent settings can be particularly challenging in the presence of "hidden gifts", and demonstrate that learning awareness in independent agents can benefit these settings.

1 Introduction

In a social world we often rely on other people to help us accomplish our goals. Sometimes, people help us even if we aren't aware of it or haven't communicated with them about it. A simple example would be if someone decides not to take the last cookie in the pantry, leaving it for others. Another interesting example is the historical "Manitokan" practice of the plains Indigenous nations of North America. In an expansive environment with limited opportunities for communication, people would cache goods for others to use at effigies (Barkwell, 2015); a temporally delayed form of reciprocal decision making. Notably, in these cases there was no explicit agreement of a trade or articulation of a "tit-for-tat"(Axelrod, 1980). Rather, people simply engaged in altruistic acts that others could then

benefit from, even without knowing who had taken the altruistic act. We refer to these undeclared altruistic acts as “hidden gifts”.

Hidden gifts represent an interesting challenge for credit assignment in multi-agent reinforcement learning (MARL). If one benefits from a hidden gift, assigning credit to the actions of the other is essentially impossible, since the action was never made clear to the beneficiary. As such, standard Bellman-back-ups (Bellman, 1954) would likely be unable to identify the critical steps that led to success in the task. Moreover, unlike a scenario where cooperation and altruistic acts can emerge through explicit agreement or a strategic equilibrium (Nash Jr, 1950), as in general sum games (Axelrod, 1980), with hidden gifts the benefits of taking an altruistic action are harder to identify.

To explore the challenge of hidden gifts for MARL we built a simple grid-world task where hidden gifts are required for optimal behavior (Chevalier-Boisvert et al., 2023). We call it the Manitokan task, in reference to the "take what you need, leave what you don’t need" inspiration from Manitokan of plains indigenous communities. In the Manitokan task, two-or-more agents are placed in an environment where each agent has a “door” that they must open in order to obtain an individual, immediate, small reward. As well, if all of the agents successfully open their door then a larger, collective reward is given to all of them. To open the doors, the agents must use a key, which the agents can both pick up and drop. However, there is only a single key in the environment. As such, if agents are to obtain the larger collective reward then they must drop the key for others to use after they have used it themselves. The agents receive an egocentric, top-down image of the environment as their observation in the task, and they can select actions of moving in the environment, picking up a key, dropping a key, or opening a door. Since the agents do not receive information about other agents’ actions, key drops represent a form of hidden gift – which make the credit assignment problem challenging. In particular: **1.** The task is fully cooperative so there is no negative reward for holding the key, and **2.** dropping the key only leads to the collective reward if the other agents take advantage of the gift.

We tested several state-of-the-art MARL algorithms on the Manitokan task. Specifically we tested Value Decomposition Networks (VDN, QMIX and QTRAN) (Sunehag et al., 2017; Son et al., 2019; Rashid et al., 2020), Multi-Agent and Independent Proximal Policy Optimization (MAPPO and IPPO) (Schulman et al., 2017; Yu et al., 2022), counterfactual multi-agent policy gradients (COMA) (Foerster et al., 2018; She et al., 2022), Multi-Agent Variational Exploration Networks (MAVEN) (Mahajan et al., 2019), an information bottleneck based Stateful Active Facilitator (SAF) (Liu et al., 2023) and standard REINFORCE policy gradients (PG) (Williams, 1992; Sutton et al., 1999; 1998; She et al., 2022). Notably, we found that none were capable of learning to drop the key and obtain the collective reward reliably. In fact, many of the MARL algorithms exhibited a total collapse of key-dropping behavior, leading to less than random performance on the collective reward. These failures held even when we provided the agents with objective relevant information, providing inputs indicating which doors were open and whether the agents were holding the key.

Interestingly, when we also provided the agents with a history of their own actions as one-hot vectors, we observed that policy gradient agents without proximal policy optimization could now solve the collective task, whereas others still failed to do so. However, these successful agents’ showed high variability in their success rate. Based on this, we analyzed the value estimation problem for this task formally, and observed that the value function necessitates an approximation of a non-constant reward. That is, the collective reward is conditioned on the other agent’s policy which is non-stationary. Inspired by learning awareness (Willi et al., 2022; Foerster et al., 2017), we derived a new term in the policy gradient theorem which corresponds to the Hessian of the collective reward objective weighted by the other agent’s policy with respect to the collective reward. Using this correction term, we show that we can reduce the variance in the performance of the independent agents and achieve consistent learning to drop the key for others.

Altogether, our key contributions in this paper are:

- We introduce a novel MARL task, the Manitokan task, involving hidden gifts that is challenging for credit assignment, but tractable for mathematical analysis.

- We provide evidence that several state-of-the art MARL algorithms cannot solve the Manitokan task, despite its apparent simplicity.
- We demonstrate that when action history is provided to the agents with memory, then independant PG agents can solve the task, but other algorithms still cannot.
- We provide a theoretical analysis of the Mantokan credit assignment problem and use it to derive a correction term based on learning-aware approaches (Foerster et al., 2017).
- We show that the derived correction term can reduce variance in the Manitokan task and improve convergence towards policies that involve leaving hidden gifts.

2 The Manitokan task for studying hidden gifts

The Manitokan task is a cooperative MARL task in a grid world. The task has been designed to be more complex than matrix games, such as Iterative Prisoner's Dilemma (Axelrod, 1980; Chammah, 1965), but capable for mathematical analysis of strategic behaviour and different from past cooperative environments (see Related Work 3). At the beginning of an episode each agent is assigned a locked door (Fig.1A) that they can only open if they hold a key. Agents can pick up the key if they move to the grid location where it is located (Fig.1B). Once an agent has opened their door it disappears and that agent receives a small individual reward immediately (Fig.1C). However, there is only one key for all agents to share and the agents can drop the key at any time if they hold it (Fig.1D). Once the key has been dropped the other agents can pick it up (Fig.1E) and use it to open their door as well (Fig.1F). If all doors are opened a larger collective reward is given to all agents, and at that point, the task terminates.

We now define the notation that we will use for describing the Manitokan task and analyzing formally. The environment is a decentralized partially observable Markov decision process (Dec-POMDP) (Goldman & Zilberstein, 2004; Bernstein et al., 2002).

Let $M = (\mathcal{N}, T, \mathcal{T}, \mathcal{O}, \mathcal{A}, \Pi, \mathcal{R}, \gamma)$, where:

- $\mathcal{N} := \{1, 2, \ldots, N\}$ is the set of N agents.
- $T \in \mathbb{N}$ is the maximum timesteps in an episode.
- $\mathcal{O} := \times_{i \in \mathcal{N}} O^i$ is the joint space of observations for the N agents and $o_t^i \in O^i \to \mathbb{N}^9$ is a partial observation for an agent i at timestep t. This is the only input agents take and thus the state $\mathcal{S} = \mathcal{O}$
- $\mathcal{A} := \times_{i \in \mathcal{N}} A^i$ is the joint space of actions and $a_t^i \in A^i$ is the action of agent i at time t.
- $\Pi := \times_{i \in \mathcal{N}} \pi^i$ is the joint space of individual agent policies.
- $\mathcal{R} \to \mathbb{R}$ is the reward function composed of both individual rewards, r_t^i, which agents receive for opening their own door (i.e. an individual objective), and the collective reward, r^c, which is given to all agents when all doors are opened (i.e. a collective objective). (See equation 1 below.)
- $\mathcal{T} : \mathcal{O} \times \mathcal{A} \to \Delta(\mathcal{O})$ is the transition function specifying the probability $\mathcal{T}(o^{i'}, \mathcal{R}^i(o^i, a^i) | o^i, a^i)$ that agent i transitions to $o^{i'}$ from o^i by taking action a^i for a reward $\mathcal{R}^i$.
- $\gamma \in [0, 1)$ is the discount factor.

The observations, o_t^i, that each agent receives are egocentric images of the 9 grid locations surrounding the current position of the agent (see the lighter portions in Fig. 1). The key, the doors, and the other agents are all visible if they are in the field of view, but not otherwise (hence the task is partially observable). The actions the agents can select, a_t^i, consist of '*move forward*', '*turn left*', '*turn right*', '*pick up the key*', '*drop the key*', and '*open the door*'. Episodes last for $T = 150$ timesteps at maximum, and are terminated early if all doors are opened.

The monotonic reward function $\mathcal{R}^i$ is defined as:

$$\mathcal{R}^i(o_t^i, a_t^i) := \begin{cases} r_t^i = r^i \text{ door opened} \\ r^c = \sum_j^N r^j \text{ all doors opened} \\ 0 \text{ otherwise} \end{cases} \tag{1}$$

The Manitokan task is unique from other credit assignment work (see. 3) in MARL due to the number of keys being strictly less than the number of agents. This scarcity requires the coordination of gifting the key between agents as a necessary critical step for success and maximizing the cumulative return. But, notably, unlike most other MARL settings the altruistic act of dropping the key is not actually observable by other agents — when an agent picks up the key they do not know if they were the first agent to do so or if other agents had held the key and dropped it for them. Thus, key drop acts are "hidden gifts" between agents and the task represents a deceptively simple, but actually complex structural credit assignment problem (Tumer et al., 2002; Agogino & Tumer, 2004; Gupta et al., 2021).

Importantly, with this set-up, the collective reward is necessarily delayed relative to any key drop actions. Moreover, key drop actions only lead to reward if the other agents have learned to accomplish their individual tasks. It then follows that the delay between a key drop action and the collective reward being received will be proportional in expectation to the number of agents, rendering a more difficult credit assignment problem for higher values of N. In the data presented here we only consider the easiest version of the task, where $N = 2$.

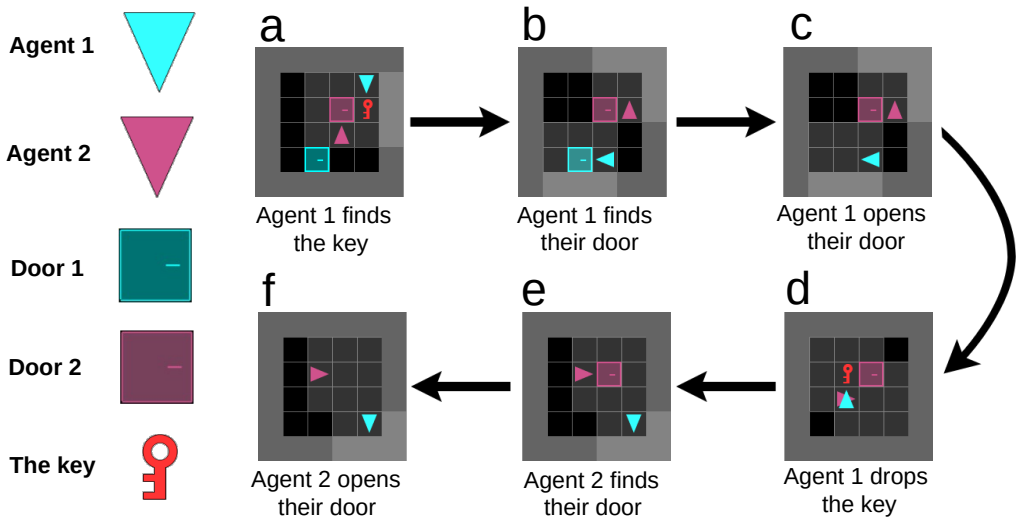


Figure 1: The deceivingly simple steps to success in the Manitokan task. a) Agent 1 finds the key; b) Agent 1 then finds their door; c) Agent 1 opens their door; d) Agent 1 drops the key as a "hidden gift"; e) Agent 3 finds their door; f) Agent 2 opens their door.

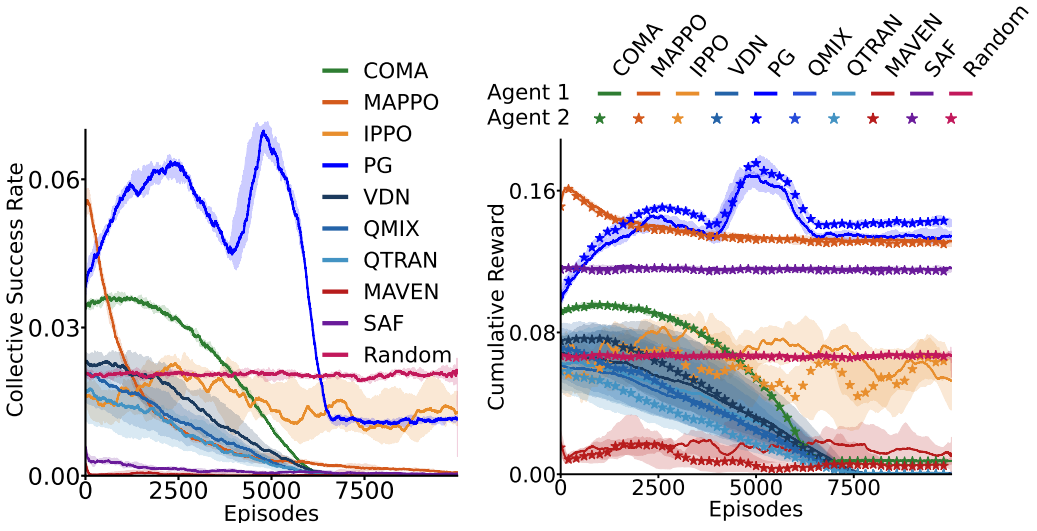
3 Results

We begin by testing the ability of various state-of-the-art model-free RL algorithms to solve this task, both multi-agent models, and independent models. For the multi-agent models, we selected ones that are prominently used as baselines for credit assignment in fully cooperative MARL tasks. These included the counterfactual model COMA, the centralized critic multi-agent PPO (MAPPO), and global value mixer models VDN, QMIX and QTRAN (Foerster et al., 2018; Yu et al., 2022; Sunehag et al., 2017; Rashid et al., 2020; Son et al., 2019). We used REINFORCE policy gradient methods, and gradient decoupled independent PPO agents (IPPO) (Williams, 1992; Sutton et al., 1999; Schulman et al., 2017). In order to alleviate problems with exploration and changing policies

we also tested MAVEN (which provides more robust exploration) and SAF (which is a meta-learning approach with a communication protocol network for learning with multiple policies) (Mahajan et al., 2019; Liu et al., 2023). All algorithms were built with recurrent components in their policy (specifically, Gated Recurrent Units, GRUs (Cho et al., 2014)) in order to provide agents with some information about task history. (See Methods A for more details on design and training.) In our initial tests we provided only the egocentric (i.e agent's "self" is included) observations as input for the agents. Hyperparameters were optimized by tuning from the sets provided in the original papers with a search to avoid overfitting on the immediate reward. As well, we trained 10 simulations with different seeds that initialized 32 parallel environments also with different random seeds. These parallel environments make the reward signals in each batch less sparse. For each simulation we ran 10,000 episodes for each 32 parallel environments, except in Figure 6 where we did 26,000 episodes. Training was done with 2 CPUs for each run and SAF required an additional A100 GPU per run. An emulator was also used to improve environment step speed (Suarez, 2024).

3.1 All algorithms fail in the basic Manitokan task

To our surprise, everything we tested converged to a level of success in obtaining the collective reward that was *below* the level achieved by a fully random policy (Fig. 2(a)) even though reward was being maximized. In fact, with the sole exception of MAPPO, all of the MARL models we tested (COMA, VDN, QMIX, QTRAN) exhibited full collapse in hidden gift behavior: these models all converged to policies that involved *less* than random key dropping frequency. Randomizing the policy can slightly improve success rate but reduced cumulative reward (Appendix 4). Notably, the agents that didn't show full collapse in collective success (MAPPO, IPPO, and SAF) were still successfully opening their individual doors, as seen by the fact that their cumulative reward was higher than the cumulative reward obtained by a random policy (Fig. 2(b)). But, the MARL agents that showed total collapse of collective behavior also showed collapse in the individual rewards. We believe that this was due to the impact of shared value updates. With shared value updates the reward signal could be swamped by noise from the unrewarded agents in the absence of key drops, and be confused by a lack of reward obtained when agents' dropped the key before opening their doors. (See more below in 4)



(a) Collective Success Rate

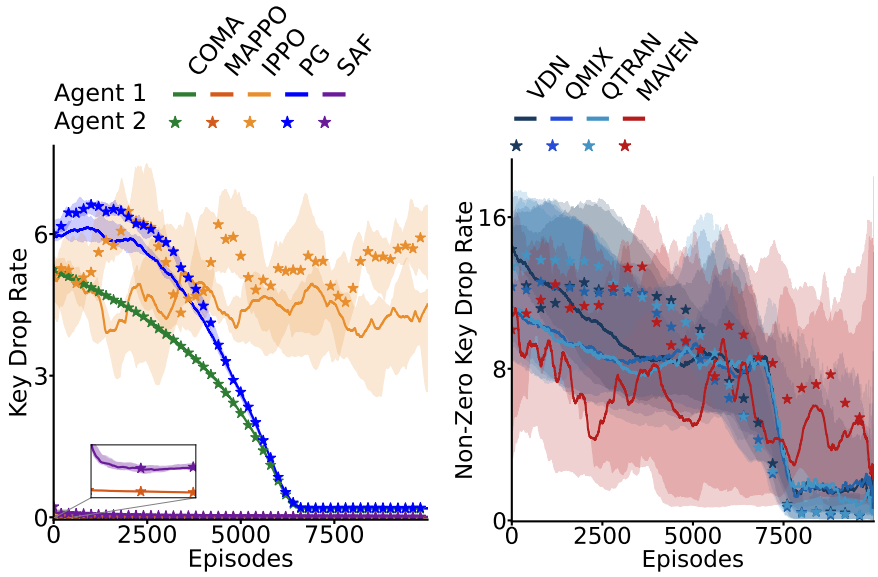
(b) Cumulative Reward

Figure 2: a) Success rate for the collective reward, i.e. percentage of trials where both agents opened their doors. b) Cumulative reward of both agents across 10000 episodes with 32 parallel environments limited to 150 timesteps each.

To maximize the cumulative reward, agents had to learn that dropping the key after opening their door is a necessary action to take (Fig 1d). As a consequence, the number of key drops that should occur in an optimal policy between both agents asymptotically on average is 1 (corresponding to a strategy of one agent always being the first to use the key) or 0.5 per agent (corresponding to both agents sharing the role of first to use the key).

We found that the key drop rates could explain the lack of collective success in this task. For most of the MARL agents (VDN, QMIX, QTRAN, MAVEN) the key drop rate always converged to exactly zero (Fig. 3(a) and E.2), hence the total collapse in collective success in the task. In the case of MAPPO, and SAF, we observed that the agents learned to pick up the key and open their individual doors, but minimized the number of key drops to close to zero (Fig. 3(a)). As a result, the collective success rate was also close to zero. Interestingly, COMA and independent PG showed very low, but non-zero rates of key drop (Fig. 3(a)), however only PG exhibited a non-zero collective success rate (Fig. 2(a)). This was because even though COMA agents learned to occasionally drop the key, the counter-factual baseline caused the loss to become excessively negative (Appendix 1). In contrast, IPPO did not exhibit a collapse in key drops, which explains its slightly better success in obtaining the collective reward (Fig. 2(a)).

One complication with measuring the key drop rate is that if the agents never even pick up the key then the key drop rate is necessarily zero. To better understand what was happening in the MARL agents, therefore, we examined the "non-zero key drop rate", meaning the rate at which keys were dropped if they were picked up. The non-zero key drop rate showed that the mixer MARL agents begin by dropping the key after picking it up some of the time, but eventually converge to a policy of simply holding or avoiding the key (Fig. 3(b)). These results show that the Manitokan task is a challenging credit assignment problem that all of the RL models we tested here failed to solve.



(a) Key Drop Rate

(b) Non Zero Key Drop Rate

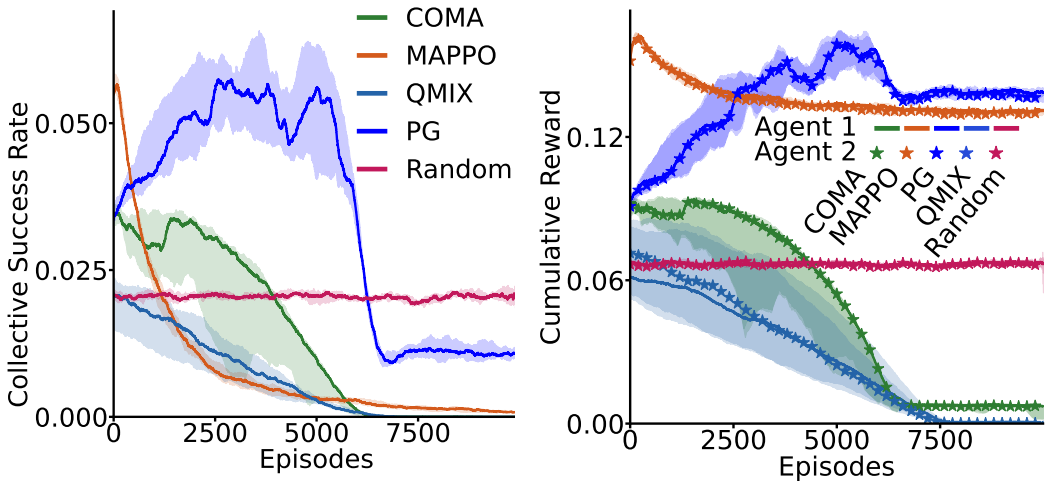
Figure 3: a) Key drop rate (i.e. cumulative key drops summed across a single episode) averaged between parallel episodes and runs. b) Non-zero key drop rate (i.e. cumulative key drops summed across a single episode) averaged between parallel episodes that had key drops and runs.

3.2 Observability of door and key status does not rescue performance in the Manitokan task

To succeed in the collective reward, agents needed to learn to pick up the key, use it, then drop it, in that order. If they did these actions out of order (e.g. dropping the key before using it), then they could not achieve collective success. As such, we reasoned that one potential cause for collapse in

performance in this task could be the fact that agents did not receive an explicit signal that they had opened their door or that they held the key (i.e. the task was partially observable with respect to these variables). Therefore, to make the task easier we provided the agents with an additional pair of observations, one which indicated whether their door was open, the other which indicated whether they held the key. With this information, theoretically, the agents could at least learn to only drop the key after their door was opened.

Surprisingly, even in this easier version of the task, the models we tested all failed to achieve collective success rates above random. In fact, the same behavior occurred, with the MARL agents (MAPPO, QMIX, COMA) showing total collapse, and the independent PG agents showing some collective success, but still below random (Fig. 4(a)). As well, as before, we found that only MAPPO and independent PG showed any learning in the task, with QMIX and COMA showing collapse in the individual success rate as well (Fig. 4(b)). Thus, the lack of information about the status of the door and key was not the cause of failure to solve the initial version of the Manitokan task.



(a) Collective Success Rate

(b) Cumulative Reward

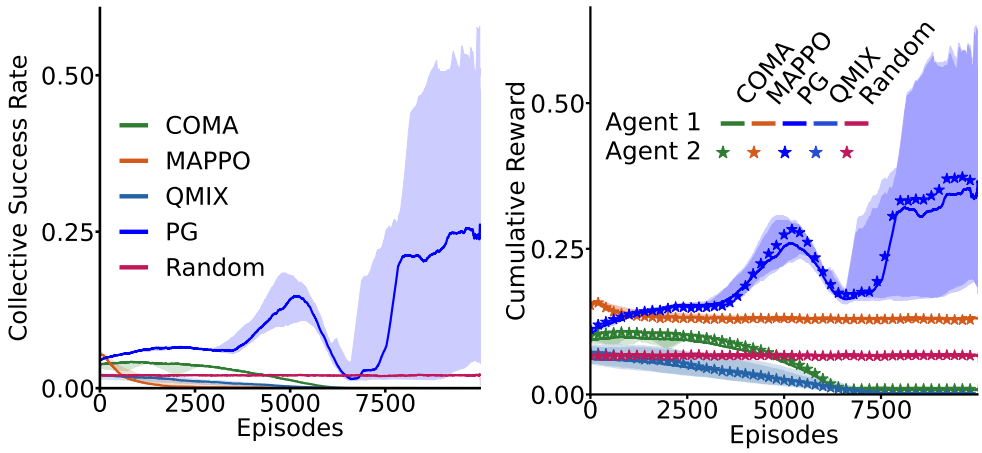
Figure 4: a) Success rate when each agent receives information about whether they have opened their door or not and if they have the key or not. b) Cumulative reward of both agents with information about whether they have opened their door or not and if they have the key or not.

3.3 Adding action history helps independent agents but not MARL agents

Next, we reasoned that another potential cause of failure was the fact that agents did not even know if they themselves had dropped the key in the past. This would make credit assignment to the key drop action very hard. To make the task easier, we provided the agents with an additional observation input, namely the last action that they took. Coupled with the recurrence in the models, this would permit the agents to know that they had dropped the key in the past if/when the collective reward was obtained. This would, in theory, make the credit assignment problem easier.

When we added the past action to the observation, we found that the PG agents now showed signs of being able to learn to obtain the collective reward, much better than random (Fig. 5). This also led to better cumulative reward for the PG agents (Fig. 5). However, interestingly, the other MARL agents still showed no ability to learn this task, exhibiting the same collapse in collective success rate and same low levels of cumulative reward as before (Fig. 5a & 5b). These results indicated to us that there is something about the credit assignment problem in the Manitokan task that can be addressed by independent policy gradients agents, but not fancier trust region mechanisms. Additionally, these PG agents still showed very high variance in their collective success rate (Fig. 5), suggesting that

there is something unique about the credit assignment problem in this task. We therefore turned to a formal analysis of the task to better understand the credit assignment problem therein.



(a) Collective Success Rate

(b) Cumulative Reward

Figure 5: a) Success rate when each agent receives their last action in the observation. b) Cumulative reward of both agents with last action information.

4 Formal analysis and correction term

For ease of analysis we focus on the situation where $N = 2$, i.e. there are only two agents. We begin by considering the objective function for agent i with parameters Θ^i, for an entire episode of the Manitokan task, where we ignore the discount factors (which don't affect the analysis):

$$J(\Theta^i) = \mathbb{E}[\sum_{t=0}^{T} \mathcal{R}^i(o_t^i, a_t^i)] = \mathbb{E}[\sum_{t=0}^{T} r_t^i + r_t^c] = \mathbb{E}[\sum_{t=0}^{T} r_t^i] + \mathbb{E}[\sum_{t=0}^{T} r_t^c] \tag{2}$$

If we consider the sub-objective related solely to the collective reward $J_c(\Theta^i) = J(\Theta^i) - \mathbb{E}[\sum_{t=0}^{T} r_t^i] = \mathbb{E}[\sum_{t=0}^{T} r_t^c]$, we can then also consider the sub-policy of the agent related to the collective reward (π_c^i), and the sub-policy unrelated to the collective reward (π_d^i). If we condition the collective reward objective on the door for agent i being open, then $J_c(\Theta^i)$ is independent of π_d^i. Therefore, when we consider the gradient for agent i of the collective objective, conditioned on their door being open, we get:

$$\nabla_{\Theta_i} J_c(\Theta^i) = \mathbb{E}[\nabla_{\Theta_i} \log \pi_c^i(a^i|o^i) Q_c(o^i, a^i)] = \mathbb{E}[\nabla_{\Theta_i} \log \pi_c^i(a^i|o^i)]\mathbb{E}[Q_c(o^i, a^i)] \tag{3}$$

where $Q_c(o^i, a^i)$ is the value solely related to the collective reward. With this set-up, we can then prove that the gradient of this collective objective is inversely related to the entropy of the other agent's policy.

Theorem 1 *Let $J_c(\Theta^i) = \mathbb{E}[\sum_{t=0}^{T} r_t^c]$ be the collective objective function for agent i, and assume that agent i is the first to open their door. Then the gradient of this objective function is given by:*

$$\nabla_{\Theta^i} J_c(\Theta^i) = \mathbb{E}[\nabla_{\Theta^i} \nabla_{\Theta^j} J_c(\Theta^j) \Psi(\pi_c^j, a^j, o^j)] \tag{4}$$

where $\Psi(\pi_c^j, a^j, o^j) = \mathbb{E}[\frac{1}{\nabla_{\Theta^j} \log \pi_c^j(a^j|o^j)}]$ and $i \neq j$.

See the P. 1 for the full proof. As a proof sketch, we rely on two key assumptions. The first key assumption is that agent i is the first to open their door. As a result, agent j's entire policy is related directly to the collective reward, and hence the sub-policy π_d^j does not exist. The second key assumption is that the other agent's collective reward policy is differentiable. With those assumptions we can then use the objective of agent j as a surrogate for the collective reward in the look-ahead step of the policy gradient derivation (Sutton et al., 1998), similar to the approach taken in mutual learning aware update rules (Willi et al., 2022; Foerster et al., 2017). The complete gradient objective from P. 1 becomes:

$$\nabla_{\Theta^i} J(\Theta^i) = \mathbb{E}[\nabla_{\Theta^i} \log \pi^i(a^i|o^i) Q(o^i, a^i) + \nabla_{\Theta^i} \nabla_{\Theta^j} J_c(\Theta^j) \Psi(\pi_c^j, a^j, o^j)] \tag{5}$$

4.1 Use of a correction term in the value function

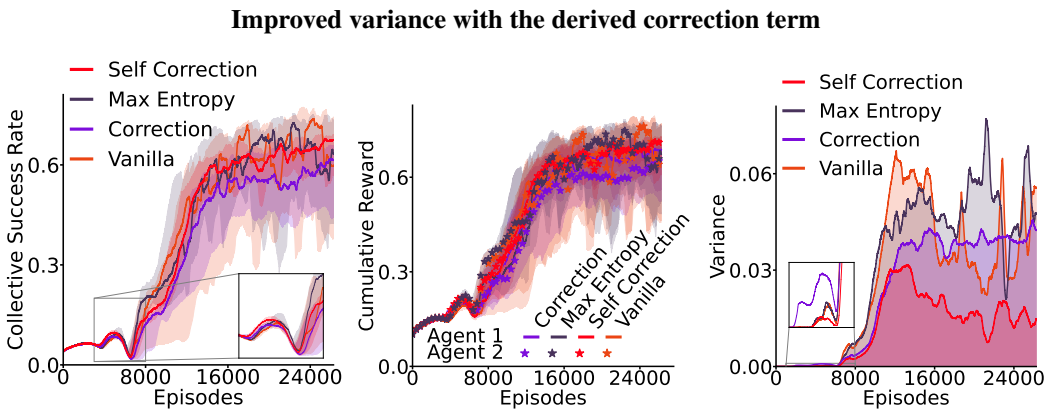
Using theorem 1 and eq. (5) we use the policy gradient theorem with the correction term. This correction should reduce the variance in the agents' abilities to obtain the collective reward by stabilizing their policies with respect to each other. As well since the reward is shared, agents only need to correct with their own parameters in expectation (see proof in P. 2). This leads to a correction term of $\nabla_{\Theta^i} \nabla_{\Theta^i} J_c(\Theta^i) \Psi(\pi_c^i, a^i, o^i)$, which we term "Self Correction". Hence, we next tested whether these correction terms applied to PG agents would indeed reduce the variance in their performance.

We tested PG agents, with action history inputs, over many episodes to ensure that we could see convergence. We examined the original PG agents and compared them to both PG agents with the correction term above, and the self-correction term. Additionally, we examined PG agents with a max-entropy term, which theoretically should also reduce the variance in the learned policies (Ahmed et al., 2019; Haarnoja et al., 2018; Eysenbach & Levine, 2022). We found that all of the agents converged to a fairly high success rate over time (Fig. 6a) and high cumulative reward (Fig. 6b). But, the variance was markedly different. The variance of the standard PG agents was quite high, and the variance of the max-entropy agents were not any lower throughout most of the episodes, with the exception of the early episodes (Fig. 6c). In contrast, the variance of the agents with the correction term was a bit lower. But, interestingly, the agents with the self-correction term showed the lowest variance. We believe that this may be due to added noise from considering multiple policies in the update. Altogether, these results confirm that the correction term we derived from our formal analysis can reduce variance in performance on the Manitokan task, but the greatest reduction in variance is achieved by using a self-correction term. This is interesting, in part, because it shows that it may be possible to resolve the complexities of hidden gift credit assignment using self-awareness, rather than full collective agent awareness.

5 Discussion

In this work we developed a MARL task to explore the complexities of learning in the presence of "hidden gifts", i.e. cooperative acts that are not actually revealed to the recipient. The Manitokan task we developed, inspired by the concept in Indigenous plains communities across North America, requires agents to open doors using a single shared key in the environment. Agents must drop the key for other agents after they have used it if they are to obtain a larger collective reward. But, these key drop acts are not apparent to the other agents, making this a task with hidden gifts.

We observed that in the basic version of the Manitokan task none of the algorithms we tested were able to solve it. This included both policy gradient agents (PG, PPO), meta-learning agents (SAF), enhanced exploration agents (MAVEN), counterfactual agents (COMA), and agents with collective value functions (VDN, QMIX, QTRAN, and MAPPO). When we added additional information to the observations the more sophisticated models tested were still not able to solve this task. However, when we provided previous action information, then the REINFORCE, PG agents could solve the task, though with high variance. Formal analysis of the value function for the Manitokan task showed that it contains a second-order term related to the collective reward that can introduce insta-



(a) Collective Success Rate (b) Cumulative Reward (c) Variance

Figure 6: Collective success rate of PG agents comparing the normal, vanilla model against one with a maximum entropy term, with the correction term, and a the self-correction term. b) Cumulative reward with and without correction terms. c) Variance in collective success rate across episodes.

bility in learning. We used this to derive a correction for the PG agents that successfully reduced the variance in their performance. Altogether, our results demonstrate that hidden gifts introduce challenging credit assignment problems that many state-of-the-art MARL algorithms cannot overcome.

5.1 Limitations

We intentionally used a grid world task to make formal analysis more tractable. But, there remains a question of whether the simplicity of the task actually made the credit assignment problem harder—it is possible that in an environment with more salient information and actions available to the agents the models could have solved the task. For example, if some form of explicit communication between agents was permitted, then perhaps it would be possible for agents to first learn to communicate their gifts to each other, only to have them become implicit and unspoken over time. This may have been how similar practices developed in the plains of North America.

Another limitation is the limited memory provided by the GRU architecture. It is possible that with a more explicit form of memory (e.g. a long context-window transformer (Ni et al., 2023; Chen et al., 2021; Cross et al., 2025) or retrieval augmented model (Hung et al., 2019) agents could more easily assign credit to their gifting behavior.

5.2 Rethinking reciprocity

A broader implication from our work is that the emergence of reciprocity in a multi-agent setting can be complicated when acts of reciprocity themselves are partially or fully unobservable and therefore temporally indirect (Nowak & Sigmund, 2005; Santos et al., 2021). One potential interesting way of dealing with these situations would be to develop agents that are good at either predicting the actions of other agents or influencing other agents with implicit information (Jaques et al., 2019; Xie et al., 2021), which would ease the inference that other agents have taken altruistic actions. The reciprocity in MARL settings with any form of "hidden gift" may generally be aided by the ability of RL agents to successfully predict the actions of others when information is asymmetric. Given that the correction term that we derived from our formal analysis was motivated by the gradient steering effect in various learning aware approaches (Willi et al., 2022; Foerster et al., 2017; Meulemans et al., 2025; Aghajohari et al., 2024), it seems reasonable to speculate that abstracting properties from learning awareness have an untapped potential exterior to the domains in which they were designed.

6 Acknowledgments

DM would like to thank Vedant Shah for the experimental ideas for further validating the results, Michael Noukhovitch for the insightful perspectives on state-information experiments and profound MARL knowledge and Arna Ghosh for verifying and questioning the correction term derivation as well has Hessian approximations.

This research was supported by the Abundant Intelligences Partnership, funded by the Government of Canada's New Frontiers in Research Fund (NFRFT-2022-00241) and the Social Sciences and Humanities Research Council (SSHRC PG 895-2023-1022). BAR also received support from NSERC (Discovery Grant: RGPIN-2020-05105; Discovery Accelerator Supplement: RGPAS-2020-00031) and CIFAR (Canada AI Chair; Learning in Machine and Brains Fellowship). DM received support from an NSERC CGSM and a Rathlyn Fellowship from the Indigenous Studies Department of McGill. This research was enabled in part by support provided by (Calcul Québec) (https://www.calculquebec.ca/en/) 9and the Digital Research Alliance of Canada (https://alliancecan.ca/en). The authors acknowledge the material support of NVIDIA in the form of computational resources.

References

Milad Aghajohari, Juan Agustin Duque, Tim Cooijmans, and Aaron Courville. LOQA: Learning with opponent q-learning awareness. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=FDQF6A1s6M.

Adrian K Agogino and Kagan Tumer. Unifying temporal and structural credit assignment problems. In *Autonomous agents and multi-agent systems conference*, 2004.

Zafarali Ahmed, Nicolas Le Roux, Mohammad Norouzi, and Dale Schuurmans. Understanding the impact of entropy on policy optimization. In *International conference on machine learning*, pp. 151–160. PMLR, 2019.

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

Lawrence J Barkwell. Manitokanac. *Gabriel Dumont Institute of Native Studies and Applied Research*, 2015. URL https://www.metismuseum.ca/resource.php/148154.

Richard Bellman. The theory of dynamic programming. *Bulletin of the American Mathematical Society*, 60(6):503–515, 1954.

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.

Micah Carroll, Rohin Shah, Mark K Ho, Tom Griffiths, Sanjit Seshia, Pieter Abbeel, and Anca Dragan. On the utility of learning about humans for human-ai coordination. *Advances in neural information processing systems*, 32, 2019.

Albert M Chammah. *Prisoner's dilemma; a study in conflict and cooperation*. Ann Arbor, U. of Michigan P, 1965.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. *Advances in Neural Information Processing Systems*, 36:73383–73394, 2023.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics. DOI: 10.3115/v1/D14-1179. URL https://aclanthology.org/D14-1179/.

Filippos Christianos, Lukas Schäfer, and Stefano Albrecht. Shared experience actor-critic for multi-agent reinforcement learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 10707–10717. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/7967cc8e3ab559e68cc944c44b1cf3e8-Paper.pdf.

Logan Cross, Violet Xiang, Agam Bhatia, Daniel LK Yamins, and Nick Haber. Hypothetical minds: Scaffolding theory of mind for multi-agent tasks with large language models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=otW0TJOUYF.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

Benjamin Eysenbach and Sergey Levine. Maximum entropy RL (provably) solves some robust RL problems. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=PtSAD3caaA2.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jakob N Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *arXiv preprint arXiv:1709.04326*, 2017.

Tobias Gessler, Tin Dizdarevic, Ani Calinescu, Benjamin Ellis, Andrei Lupu, and Jakob Nicolaus Foerster. Overcookedv2: Rethinking overcooked for zero-shot coordination. *arXiv preprint arXiv:2503.17821*, 2025.

Claudia V Goldman and Shlomo Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of artificial intelligence research*, 22:143–174, 2004.

Dhawal Gupta, Gabor Mihucz, Matthew Schlegel, James Kostas, Philip S Thomas, and Martha White. Structural credit assignment in neural networks using reinforcement learning. *Advances in Neural Information Processing Systems*, 34:30257–30270, 2021.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Chia-Chun Hung, Timothy Lillicrap, Josh Abramson, Yan Wu, Mehdi Mirza, Federico Carnevale, Arun Ahuja, and Greg Wayne. Optimizing agent behavior over long time scales by transporting value. *Nature communications*, 10(1):5223, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro Ortega, DJ Strouse, Joel Z Leibo, and Nando De Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *International conference on machine learning*, pp. 3040–3049. PMLR, 2019.

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.

Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*, 2017.

Dianbo Liu, Vedant Shah, Oussama Boussif, Cristian Meo, Anirudh Goyal, Tianmin Shu, Michael Curtis Mozer, Nicolas Heess, and Yoshua Bengio. Stateful active facilitator: Coordination and environmental heterogeneity in cooperative multi-agent reinforcement learning. In *ICLR*, 2023.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Andrei Lupu and Doina Precup. Gifting in multi-agent reinforcement learning. In *Proceedings of the 19th International Conference on autonomous agents and multiagent systems*, pp. 789–797, 2020.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in neural information processing systems*, 32, 2019.

Alexander Meulemans, Seijin Kobayashi, Johannes von Oswald, Nino Scherrer, Eric Elmoznino, Blake Richards, Guillaume Lajoie, Blaise Agüera y Arcas, and João Sacramento. Multi-agent cooperation through learning-aware policy gradients, 2025. URL https://arxiv.org/abs/2410.18636.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

John F Nash Jr. Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49, 1950.

Tianwei Ni, Michel Ma, Benjamin Eysenbach, and Pierre-Luc Bacon. When do transformers shine in rl? decoupling memory from credit assignment. *Advances in Neural Information Processing Systems*, 36:50429–50452, 2023.

Martin A Nowak and Karl Sigmund. Evolution of indirect reciprocity. *Nature*, 437(7063):1291–1298, 2005.

Barak A. Pearlmutter. Fast exact multiplication by the hessian. *Neural Computation*, 6(1):147–160, 1994. DOI: 10.1162/neco.1994.6.1.147.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

Fernando P Santos, Jorge M Pacheco, and Francisco C Santos. The complexity of human cooperation under indirect reciprocity. *Philosophical Transactions of the Royal Society B*, 376(1838): 20200291, 2021.

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

Jennifer She, Jayesh K Gupta, and Mykel J Kochenderfer. Agent-time attention for sparse rewards multi-agent reinforcement learning. *arXiv preprint arXiv:2210.17540*, 2022.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pp. 5887–5896. PMLR, 2019.

Joseph Suarez. Pufferlib: Making reinforcement learning libraries and environments play nice. *arXiv preprint arXiv:2406.12905*, 2024.

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. *The International Foundation for Autonomous Agents and Multiagent Systems*, 2017.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

Kagan Tumer, Adrian K Agogino, and David H Wolpert. Learning sequences of actions in collectives of autonomous agents. In *Proceedings of the first international joint conference on autonomous agents and multiagent systems: Part 1*, pp. 378–385, 2002.

Li Wang, Yupeng Zhang, Yujing Hu, Weixun Wang, Chongjie Zhang, Yang Gao, Jianye Hao, Tangjie Lv, and Changjie Fan. Individual reward assisted multi-agent reinforcement learning. In *International conference on machine learning*, pp. 23417–23432. PMLR, 2022.

Timon Willi, Alistair Hp Letcher, Johannes Treutlein, and Jakob Foerster. Cola: consistent learning with opponent-learning awareness. In *International Conference on Machine Learning*, pp. 23804–23831. PMLR, 2022.

Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

Annie Xie, Dylan Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. In *Conference on robot learning*, pp. 575–588. PMLR, 2021.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Supplementary Materials

The following content was not necessarily subject to peer review.

A Appendix

M Methods

Methods This section contains the hyperparameters for the results, hardware details for training and minor details on the task setup.

M.1 Hyperparameters

Table 1: Model architecture and hyperparameters used for MAPPO.

Component	**Specification**
Policy Network Architecture (Joint)	1-layer CNN (outchannels = 32, kernal = 3, ReLU), 1-layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 64, output=64, ReLU), 1 layer GRU (input = 64, output = 64, with LayerNorm), 1 layer Categorical (input=64, output=6)
Value Network Architecture (Joint)	1-layer CNN (outchannels = 32, kernal = 3, ReLU), 1-layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 64, output=64, ReLU), 1 layer GRU (input = 64, output = 64, with LayerNorm), 1 layer MLP(input = 64, output = 1, ReLU)
Optimizer	Adam, learning rate: 1×10^{-5}
Discount Factor γ	0.99
GAE Parameter λ	0.95
PPO Clip Ratio ϵ	0.2
Entropy Coefficient	0.0001
Data chunk length	10
Parallel Environments	32
Batch Size	Parallel Environments × Data chunk length × number of agents
Mini-batch Size	1
Epochs per Update	15
Gradient Clipping	10
Value Function Coef.	1
Gain	0.01
Loss	Huber Loss with delta 10.00

Table 2: Model architecture and hyperparameters used for IPPO.

Component	**Specification**
Policy Network Architecture (Disjoint)	1-layer CNN (outchannels = 32, kernal = 3, ReLU), 1-layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 64, output=64, ReLU), 1 layer GRU (input = 64, output = 64, with LayerNorm), 1 layer Categorical (input=64, output=6)
Value Network Architecture (Disjoint)	1-layer CNN (outchannels = 32, kernal = 3, ReLU), 1-layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 32, output=64, ReLU), 1 layer MLP (input = 64, output=64, ReLU), 1 layer GRU (input = 64, output = 64, with LayerNorm), 1 layer MLP(input = 64, output = 1, ReLU)
Optimizer	Adam
Learning rate	1×10^{-5}
Discount Factor γ	0.99
GAE Parameter λ	Not used
PPO Clip Ratio ϵ	0.2
Entropy Coefficient	0.0001
Data chunk length	10
Parallel Environments	32
Batch Size	Parallel Environments × Data chunk length × number of agents
Mini-batch Size	1
Epochs per Update	15
Gradient Clipping	10
Value Function Coef.	1
Gain	0.01
Loss	Huber Loss with delta 10.00

Table 3: Model architecture and hyperparameters used for PG.

Component	**Specification**
Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Critic Network Architecture (Disjoint)	1-layer MLP (input = 27, output=64, ReLU), 1-layer MLP (input = 64, output=64, ReLU), 1-layer MLP (input=64, output=1)
Target Critic Network Architecture (Disjoint)	1-layer MLP (input = 27, output=64, ReLU), 1-layer MLP (input = 64, output=64, ReLU), 1-layer MLP (input=64, output=1)
Actor optimizer	RMSprop, alpha 0.99, epsilon 1×10^{-5}
Critic optimizer	RMSprop, alpha 0.99, epsilon 1×10^{-5}
Discount factor γ	0.99
Target network update interval	1 episode
Learning rate	5×10^{-5}
TD Lambda	1.0
Replay buffer size	32
Parallel environment	32
Parallel episodes per buffer episode	1
Training batch size	32

Table 4: Model architecture and hyperparameters used for COMA.

Component	**Specification**
Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Critic Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1-layer MLP (input = 64, output=64, ReLU), 1-layer MLP (input=64, output=6)
Target Critic Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1-layer MLP (input = 64, output=64, ReLU), 1-layer MLP (input=64, output=6)
Actor optimizer	RMSprop, alpha 0.99, epsilon 1×10^{-5}
Critic optimizer	RMSprop, alpha 0.99, epsilon 1×10^{-5}
Discount factor γ	0.99
Target network update interval	1 episode
Learning rate	5×10^{-5}
TD Lambda	1.0
Replay buffer size	320
Parallel environment	32
Parallel episodes per buffer episode	1
Training batch size	32

Table 5: Model architecture and hyperparameters used for SAF.

Component	**Specification**
Policy Network Architecture (Disjoint)	2-layer MLP (input = 64, output=128, Tanh),
Value Network Architecture (Joint)	2-layer MLP (input = 80, output=128, Tanh),
Shared Convolutional Encoder (Joint)	1-Layer CNN (outchannels = 64, kernal = 2)
Knowledge Source Architecture (Joint)	
Query Projector	1-layer MLP (input = 128, output=64, Tanh)
State Projector	1-layer MLP (input = 128, output=64, Tanh)
Perceiver Encoder	(latents = 4, latent input = 64, cross attention channels = 64, cross attention heads = 1, self attention heads = 1, self attention blocks = 2 with 2 layers each)
Cross Attention	(heads = 1, query input = 64, key-value input = 64, query-key input = 64, value channels = 64, dropout = 0.0)
Combined State Projector	1-layer MLP (input = 128, output=64, Tanh)
Latent Encoder	1-layer MLP (input = 128, output=64, Tanh), 1-layer MLP (input = 64, output=64, Tanh),1-layer MLP (input = 64, output=16, Tanh)
Latent Encoder Prior	1-layer MLP (input = 64, output=64, Tanh), 1-layer MLP (input = 64, output=64, Tanh),1-layer MLP (input = 64, output=16, Tanh)
Policy Projector	1-layer MLP (input = 128, output=164, Tanh)
Optimizer	Adam, epsilon 1×10^{-5}
learning rate	3×10^{-4}
Discount Factor γ	0.99
GAE Parameter λ	GAE not used
PPO Clip Ratio ϵ	0.2
Entropy Coefficient	0.01
Data chunk length	10
Parallel Environments	32
Batch Size	Parallel Environments × Data chunk length × number of agents
Mini-batch Size	5
Epochs per Update	15
Gradient Clipping	9
Value Function Coef.	1
Gain	0.01
Loss	Huber Loss with delta 10.00
Number of policies	4
Number of slot keys	4

Table 6: Model architecture and hyperparameters used for VDN.

Component	**Specification**
Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 128, output = 64), 1 layer MLP (input=128, output=6)
Target Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 128, output = 64), 1 layer MLP (input=128, output=6)
Mixer Network Architecture	Tensor sum of states
Policy optimizer	Adam, alpha 0.99, epsilon 1×10^{-5}
Target policy optimizer	Adam, alpha 0.99, epsilon 1×10^{-5}
Start epsilon greedy	1.0
Minimum epsilon greedy	0.05
Discount factor γ	0.99
Target network update interval	1 episode
Start learning rate	1×10^{-2}
Minimum learning rate	1×10^{-6}
TD Lambda	1.0
Replay buffer size	1000
Parallel environment	32
Parallel episodes per buffer episode	32
Training batch size	32
Warm up buffer episodes	32

Table 7: Model architecture and hyperparameters used for QMIX.

Component	**Specification**
Actor Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Target Actor Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Mixing Network Architecture (Joint)	
Hypernet Weights 1	1-layer MLP (input =54, output=64, ReLU), 1-layer MLP (input = 64, output=52)
Hypernet Biases 1	1-layer MLP (input =54, output=64)
Hypernet Weights 2	1-layer MLP (input =54, output=32, ReLU), 1-layer MLP (input = 64, output=32)
Hypernet Bias 2	1-layer MLP (input =54, output=64, ReLU), 1-layer MLP (input = 64, output=1)
Target Mixing Network Architecture (Joint)	
Hypernet Weights 1	1-layer MLP (input =54, output=64, ReLU), 1-layer MLP (input = 64, output=52)
Hypernet Biases 1	1-layer MLP (input =54, output=64)
Hypernet Weights 2	1-layer MLP (input =54, output=32, ReLU), 1-layer MLP (input = 64, output=32)
Hypernet Bias 2	1-layer MLP (input =54, output=64, ReLU), 1-layer MLP (input = 64, output=1)
Policy optimizer	Adam, alpha 0.99, epsilon 1×10^{-5}
Target policy optimizer	Adam, alpha 0.99, epsilon 1×10^{-5}
Start epsilon greedy	1.0
Minimum epsilon greedy	0.05
Discount factor γ	0.99
Target network update interval	1 episode
Start learning rate	1×10^{-2}
Minimum learning rate	1×10^{-6}
TD Lambda	1.0
Replay buffer size	1000
Parallel environment	32
Parallel episodes per buffer episode	32
Training batch size	32
Warm up buffer episodes	32

Table 8: Model architecture and hyperparameters used for QTRAN.

Component	**Specification**
Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Target Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Mixing Network Architecture (Joint)	
Query Network	1-layer MLP (input =188, output=32, ReLU), 1-layer MLP (input = 32, output=32, ReLU), 1-layer MLP (input = 32, output=1)
Value Network	1-layer MLP (input =54, output=32, ReLU), 1-layer MLP (input = 32, output=32, ReLU), 1-layer MLP (input = 32, output=1)
Action Encoding	1-layer MLP (input =134, output=134, ReLU), 1-layer MLP (input = 134, output=134)
Target Mixing Network Architecture (Joint)	
Query Network	1-layer MLP (input =188, output=32, ReLU), 1-layer MLP (input = 32, output=32, ReLU), 1-layer MLP (input = 32, output=1)
Value Network	1-layer MLP (input =54, output=32, ReLU), 1-layer MLP (input = 32, output=32, ReLU), 1-layer MLP (input = 32, output=1)
Action Encoding	1-layer MLP (input =134, output=134, ReLU), 1-layer MLP (input = 134, output=134)
Policy optimizer	Adam, alpha 0.99, epsilon 1×10^{-5}
Target policy optimizer	Adam, alpha 0.99, epsilon 1×10^{-5}
Start epsilon greedy	1.0
Minimum epsilon greedy	0.05
Discount factor γ	0.99
Target network update interval	1 episode
Start learning rate	1×10^{-2}
Minimum learning rate	1×10^{-6}
TD Lambda	1.0
Replay buffer size	1000
Parallel environment	32
Parallel episodes per buffer episode	32
Training batch size	32
Warm up buffer episodes	32

Table 9: Model architecture and hyperparameters used for MAVEN.

Component	**Specification**
Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Target Policy Network Architecture (Joint)	1-layer MLP (input = 27, output=64, ReLU), 1 layer GRU (input = 64, output = 64), 1 layer MLP (input=64, output=6)
Noise Mixing Network Architecture (Joint)	
Hypernet Weights 1	1-layer MLP (input=116, output=64)
Hypernet Bias 1	1-layer MLP (input=116, output=32)
Hypernet Weights 2	1-layer MLP (input=116, output=32)
Skip Connection	1-layer MLP (input=116, output=2)
Value network	1-layer MLP (input=116, output=32, ReLU), 1-layer MLP(input=32,output=1)
Target Noise Mixing Network Architecture (Joint)	
Hypernet Weights 1	1-layer MLP (input=116, output=64)
Hypernet Bias 1	1-layer MLP (input=116, output=32)
Hypernet Weights 2	1-layer MLP (input=116, output=32)
Skip Connection	1-layer MLP (input=116, output=2)
Value network	1-layer MLP (input=116, output=32, ReLU), 1-layer MLP(input=32,output=1)
RNN Aggregator	1-layer GRU (input=116, output=2)
Discriminator	1-layer MLP (input=116, output=32, ReLU), 1-layer MLP (input=32, output=2),
Actor optimizer	RMSprop, alpha 0.99, epsilon 1×10^{-5}
Target actor optimizer	Adam, alpha 0.99, epsilon 1×10^{-5}
Use skip connection in mixer	False
Use RNN aggregation	False
Discount factor γ	0.99
Target network update interval	1 episode
Learning rate	5×10^{-5}
TD Lambda	1.0
Replay buffer size	1000
Parallel environment	32
Parallel episodes per buffer episode	1
Training batch size	32

M.2 Compute

For each simulation 2 CPUs were allocated and the 32 parallel environments were multithreaded. All models expect for SAF were able to run without GPUs while SAF used a single A100 for each simulation. All models, except for VDN, QMIX and QTRAN can finish at 10000 episodes for all 10 simulations within 4 days while the aforementioned models take 7 days. It is possible to use a GPU for these value mixer models for faster data collection but this was not done to collect the data. The correction term experiments take 7 days to collect 26000 episodes and do not benefit from GPUs since their networks are too small. The Hessian term was approximated with Pearlmutter's trick (Pearlmutter, 1994).

M.3 Manitokan task setup

The Manitokan Task is a grid world for tractable analysis. The key, agents and doors are randomly initialized at the beginning of each episode and the actions *drop* and *toggle* were additionally pruned when an agent is not holding a key for reasonable environment logic but are not necessary to be removed for the task to work. Everything else was described in 2.

R Related Work

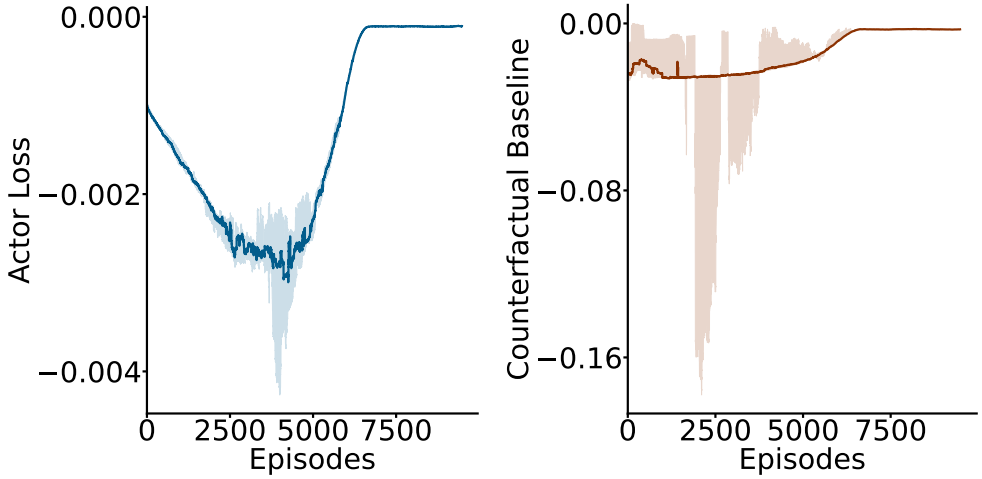
Coordination games in MARL are fully cooperative tasks such as the Multiple Particle Environment (Mordatch & Abbeel, 2017; Lowe et al., 2017), Overcooked (Carroll et al., 2019; Gessler et al., 2025), the Starcraft Multi-Agent Challenge (Samvelyan et al., 2019; Ellis et al., 2023) and variations of harvesting or collecting environments (Leibo et al., 2017; Lerer & Peysakhovich, 2017; Lupu & Precup, 2020; Christianos et al., 2020). In these, there is a single team objective that agents requiring more than one agent for the group to be successful. Other than share global information, these fully cooperative environments also distribute the same reward to all agents even ones who's actions did not contribute to success to spur coordination.

This shared reward guarantees that the agents update their policies together and will repeat the any actions that were associated with that reward. The few works have investigated the impact individual rewards in coordination games found mixed results and possible policy inconsistency that harmed perofrmance (Wang et al., 2022). Uniquely, the Manitokan task in section 2 has an individual reward for opening a door with a key that is necessary to then acquire the collective reward through leaving the key. The collective reward changes after each policy update since it is the only reward needing coordination and the agents are challenged on when to leave the gift.

E Additional Experiments

The experiments provided below offer insights into the challenge of the Manitokan Task, and further empirical validation of the correction and self correction terms.

E.1 COMA's loss becomes negative



(a) Actor Loss (b) Counterfactual Baseline

Figure 7: a) Policy loss of the COMA model b) Counterfactual baseline in the COMA policy update

COMA persistently collapsed even though it exhibited similar learning behaviour to PG (a closely related model). The policy loss and baseline curves show increasing instability with large variance spikes before converging to a value around 0.0. Perhaps this collapse is from the difficulty of leaving a hidden gift between individual and collective incentives. The original COMA paper Foerster et al. (2018) even mentions a struggle for an agent overcoming an individual reward, although exterior to hidden gifts, may be cause for the instability.

E.2 Key drops across all parallel environment for value mixer models collapses

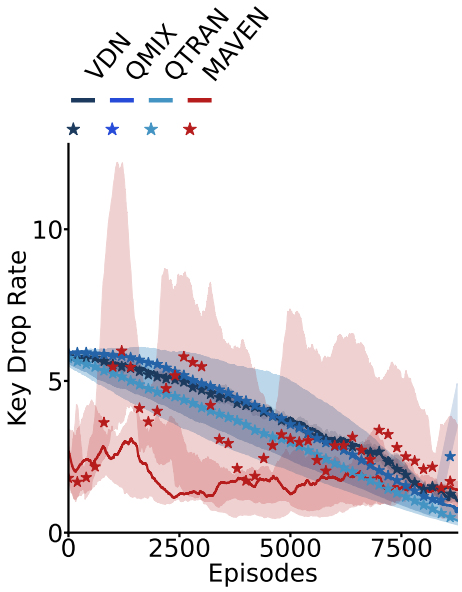
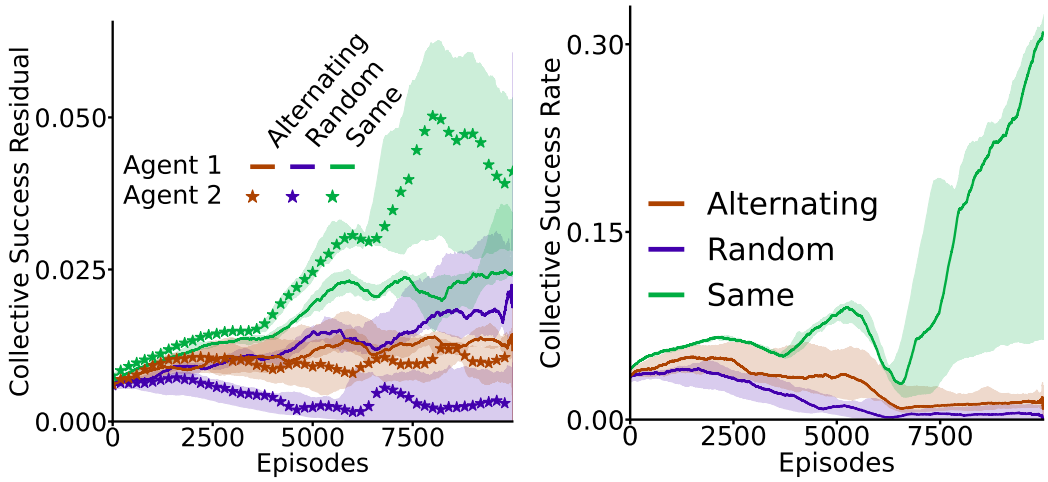


Figure 8: Keydrops averaged over all parallel environments including ones with zero drops with models that mix values into a global value function (VDN, QMIX, QTRAN and MAVEN).

The non-zero key drop rate in the main results Fig 3(b) showed a wider variation between agents and small learning effect. The decreased variance in the appendix Fig 8 is most likely attributed to

agents not finding the key at all due to noise from the global value updates. The second QMIX agent also contains a burst in key drops towards the end with

E.3 Changing which agent steps first in an episode harms performance



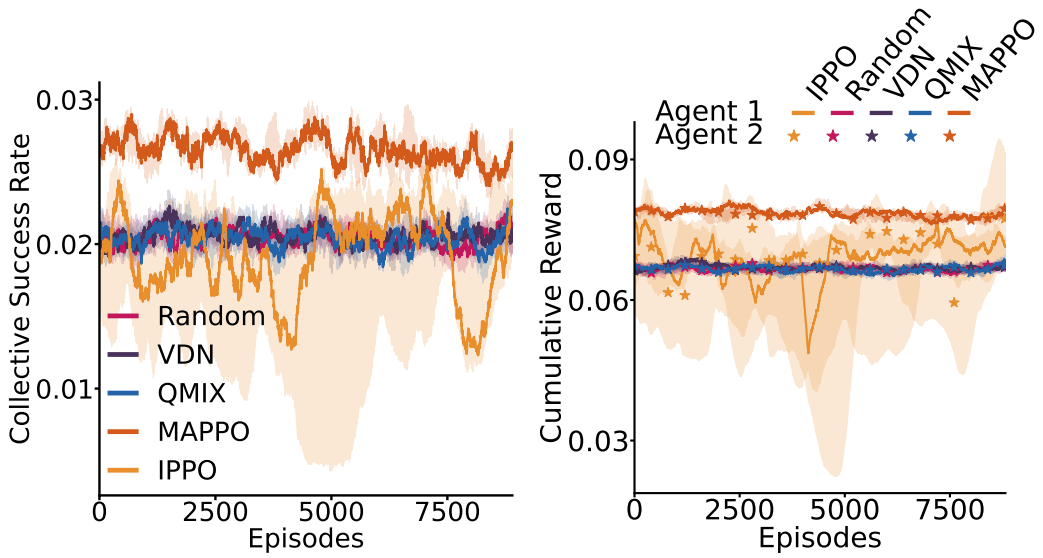
(a) Collective Success Rate Residual

(b) Collective Success Rate

Figure 9: a) The contribution of an agent's reward accumulation to success weighted by their total reward comparing policy gradient agents with action history of the same agent stepping first (i.e. agent 1 then agent 2), alternating agents stepping first (i.e. agent 1 steps first on odd numbered episodes and agent 2 steps first in even numbers episodes), and a random agent is selecting to step first. b) Success rate between different step ordering each episode.

The collective success residual is calculated as $(r^c - r^i) \times r^i$ where $(r^c - r^i)$ describes how much an agent i is contributing to the collective success while weighting it by r^i shows if the agents are increasing that success rate. Interestingly, alternating which agent goes first between episodes creates oscillations in the collective success rate residual where one agent receiving more reward means the other agent receives less. Greatly reducing the success. Moreover, randomly selecting an agent to go first biases the first agent to increase their reward and almost removes all success. These effect may be caused by uncertainty associated with which agent can reach the key when the other agent is in sight.

E.4 Randomizing the policy can increase collective success slightly



(a) Collective Success Rate

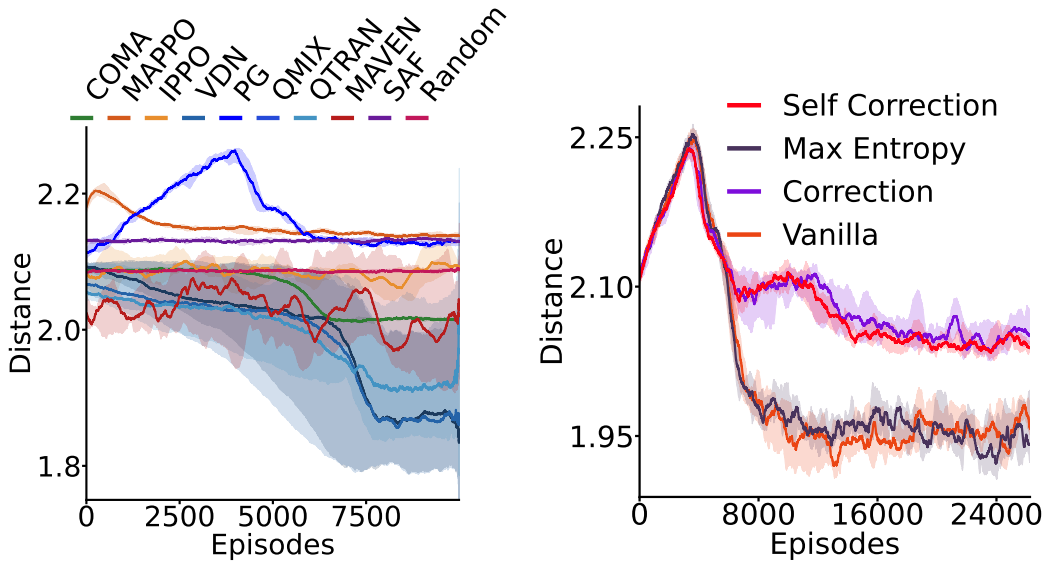
(b) Cumulative Reward

Figure 10: a) Comparing agents of MAPPO, IPPO, VDN and QMIX models with a randomization applied to their policies b) The cumulative reward for randomized policy agents

PPO agents had their value function learning rates set to 0.001 while the policy learning rates where kept as 0.000001. This meant the policy would always prefer initial episodes and converge quickly to those while the value function weighting them more evenly to converge further in the training process. VDN and QMIX use epsilon greedy in their strategy and simply increasing the time of decay for this mechanism led these agents to be more random throughout the experiment.

This policy randomization process very slightly improved these agents the success rates' compared to those in the main results Fig 2(a) but decreased the cumulative reward for the PPO agents than those in Fig 2(b). The random policy aligned VDN and QMIX to the random action baseline more or less, and avoided collapse.

E.5 Behavioural variations appear between models with inter agent distance and minimizing the steps to the first reward



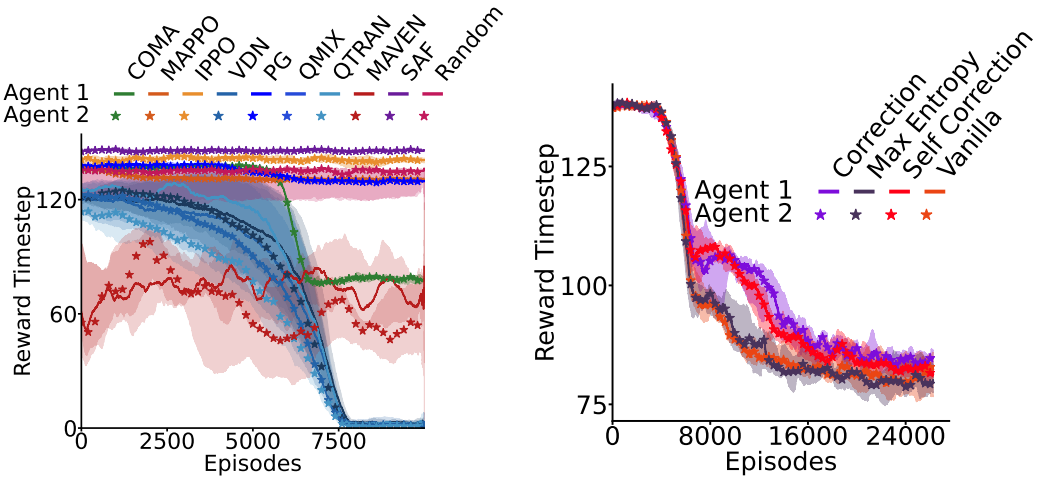
(a) Distance

(b) Correction Term Distance

Figure 11: a) Euclidean distance between agents averaged over parallel environments and simulations across our tested models b) Euclidean distance comparing policy gradient agents with action history and variance reduction terms.

Although the 2-agent Manitokan Task is a four by four grid world, we measured the euclidean distance between agents to see if they become more coordinated or adversarial when learning hidden gifting. In Fig 11(a), PG agents exhibited the highest exploration phase but eventually converged to a lower distance. MAPPO agents also has a similar but substantially smaller exploration effect in the very beginning while SAF did not have any exploration phases. IPPO and MAVEN agents similarly hovered below the random baseline but MAVEN agents were closer to each other. COMA agents begin around random but converge to be closer to each other as well. Value mixer agents VDN, QMIX and QTRAN all are on average closer to each other but QTRAN agent agents converge further apart.

In Fig 11(b), vanilla and max entropy PG agents with action history become asymptotically closer to each other while the correction term agents converge further apart from them. The variance reduction in self correcting agents is also noticeable.



(a) Timestep of First Reward

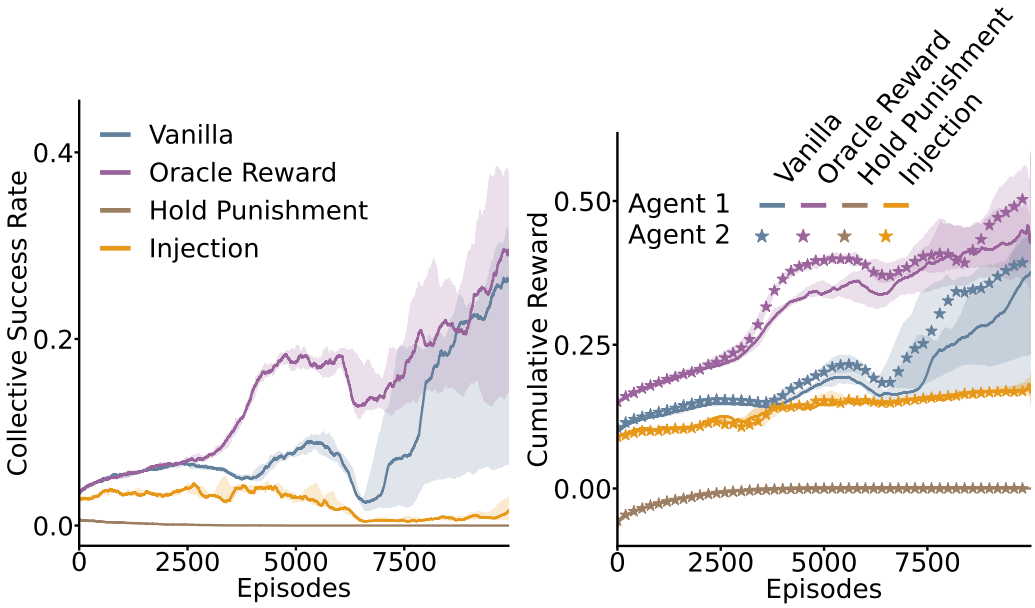
(b) Correction Term Timestep of First Reward

Figure 12: a) Timestep the first reward an agent received. b) Timestep the first reward a policy gradient agent with action history received.

The reducing the timestep of the first reward is a way to measure if agents are improving their policies if cumulative reward also increases. In (Fig 12(a)), PG, IPPO, MAPPO and SAF all converge quickly while PG and MAPPO learn policies of reducing the step slightly below random. COMA converges at a low timestep but this is most likely due to the collapse. MAVEN oscillates at a timestep better than random but never converges and doesn't seem to learn a good policy and VDN, QMIX, and QTRAN collapse consistently with other results in Section 3.

While in Fig 12(b), all PG models with action history reduce their initial reward timesteps but models with the correction term converge slower.

E.6 Modifying the reward function enhances perspective on the challenge of the Manitokan task



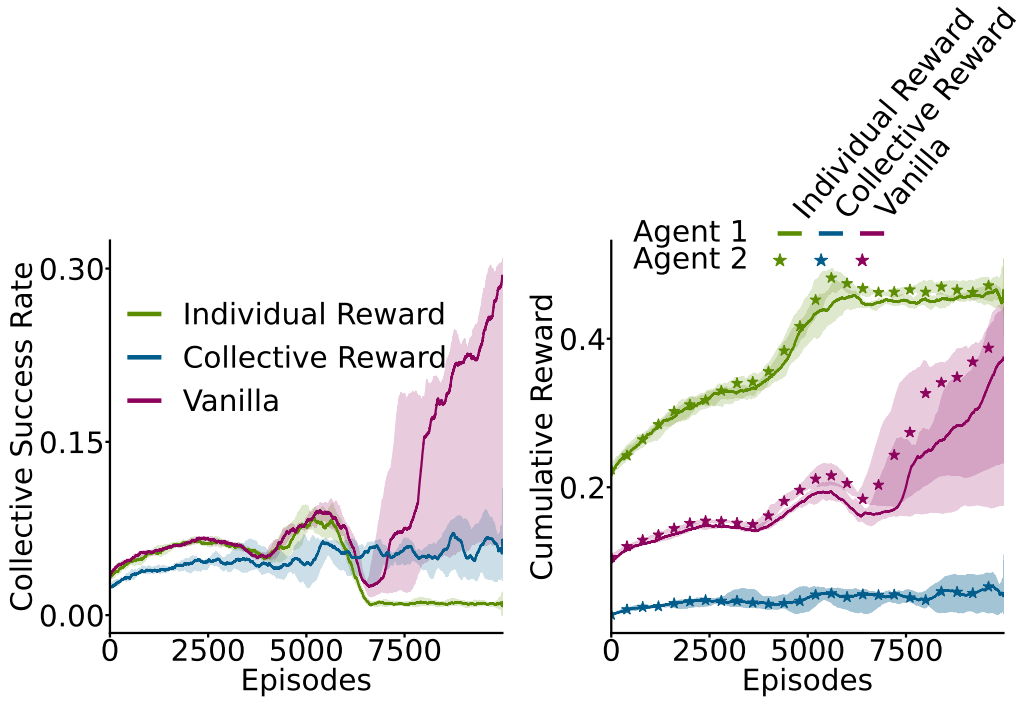
(a) Collective Success Rate

(b) Cumulative Reward

Figure 13: a) Success rate of policy gradient agents with action history comparing the normal reward function with an oracle reward term (i.e. an agent receives a reward of 1 once for dropping the key after opening their door), a punishment term (i.e.. a negative reward of 1 is applied each step an agent holds their key after opening their door) and a reward injection term (i.e. randomly distributing normally smaller rewards around the standard rewards decaying over episodes) b) Cumulative reward to compare the modified reward functions

The reward function $\mathcal{R}$ in equation 1 to study hidden gifting behavior is both sparse with a hard to predict collective reward conditioned on the other agent's policy. We tested additional reward conditions on PG agents with action history to see if sample efficiency improvement can be found. Particularly, the oracle reward: r_t^i the first key dropped after agent i's door is opened , is the critical step to take for hidden gifting and when implemented the collective success rate increased quicker than the normal reward function. The punishment reward: -0.5 for each step agent i is holding the key after their door was opened, is also meant to induce gifting behavior but agents seemed to avoid the key altogether. Lastly, the injection reward where a set of rewards $r^d < r^i$ are normally distributed around rewards r^i and r^c which also served as the mean. r^d was additionally reduced each episode for agents to prefer the standard rewards. Injection reduced the success rate severely but also reduced variance in accumulating the expected reward.

These minor modifications reemphasize the difficulty in hidden gifting, where our most performative agents still struggle even when rewarded for the optimal action.



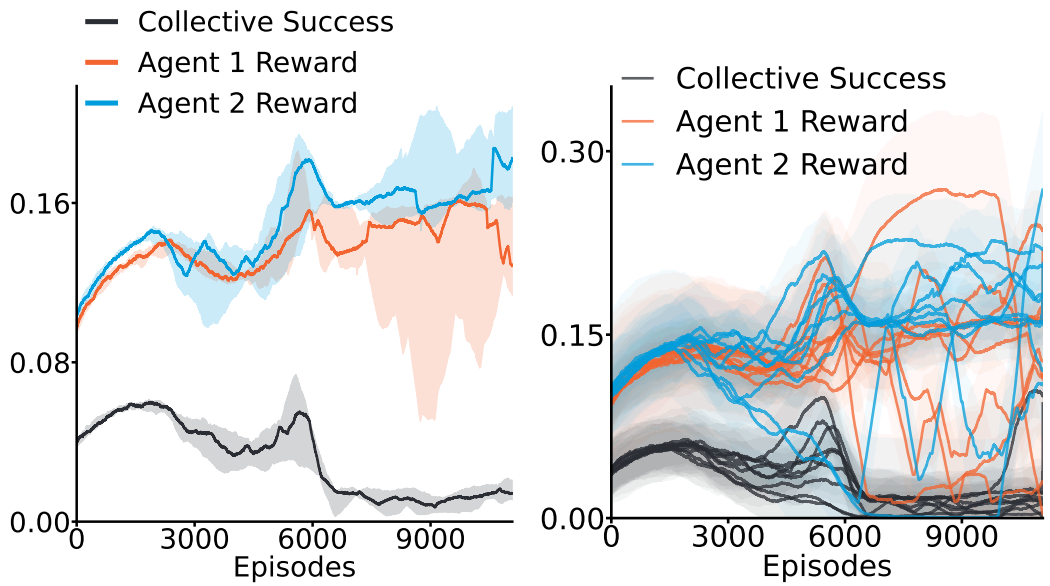
(a) Collective Success Rate

(b) Cumulative Reward

Figure 14: a) Success rate between policy gradient agents comparing a disassociation of the reward function (i.e.. just the individual reward and the collective rewards) b) Cumulative reward of the same dissociated reward function agents

For a further investigation of the reward function, we tested a dissociation of the individual reward r^i and the collective reward r^c with action history PG agents. Using only the individual reward removed collective success altogether but agents converged at a higher percentage of the cumulative reward (i.e.. whoever gets to the key first). The sole collective reward did not cause a failure in collective behavior but severely inhibited it. With both these reward dissociation, agents fail to learn hidden gifting.

E.7 The self correction term is empirically sound in contraposition



(a) Collective Success Rate

(b) Collective Success Rate Across Simulations

Figure 15: a) The percentage of cumulative reward and collective success for anti-collective policy gradient agents with action history (i.e. optimizing the negated self correction term) across 11000 episodes b) 9 individual simulations for anti-collective behaviour averaged each across a different set of 32 parallel environments

For all previous experiments, the correction term was maximized to induce agents towards dropping the key for the other agent (i.e. hidden gifting). Contrapositively however, this term for an agent i could also be minimized through negation $-\mathbb{E}[\nabla_{\Theta^i}\nabla_{\Theta^j}J_c(\Theta^j)\Psi(\pi_c^j, a^j, o^j)]$ in the policy update and doing so led agents to actively "compete" for the key and avoid dropping it all together. In Fig 15(a), the rewards for both agents increases with variance spikes while the collective success rate goes down. These results demonstrate a stronger implication of the self-correction in the collective behaviour of agents than just as a variance reducer.

Fig 15(b) displays the individual simulations with standard deviation of the 32 parallel environments. Specifically, the reward curves sharply drop and return after agents have learned to open their doors. This tradeoff in the individual reward accumulation is a detriment to the collective success rate but perhaps in other situations, the negative correction term can help avoid undesired rewarded behaviour.

P Proofs

P.1 Correction term

We begin by deriving the standard policy gradient theorem (Sutton et al., 1998; 1999) under the assumptions in Section 4 that an agent i is first to open their door and that the collective reward r^c is differentiable through another agent js objective. The objective $J(\Theta^i)$ for agent i is to maximize the expected cumulative sum of rewards within an episode $\mathbb{E}[\sum_t^T \mathcal{R}^i(o_t^i, a_t^i)]$ with the reward function $\mathcal{R}$ in equation 1 where a value function $V(\Theta^i, o^i) = \mathbb{E}[\mathcal{R}^i(o^i, a^i)]$.

$$\nabla_{\Theta^i} J(\Theta^i) = \nabla_{\Theta^i}(\sum_{a^i \in A} \pi^i(a^i|o^i) Q(o^i, a^i)) \tag{6}$$

is the differentiated objective with respect to agent i.

$$\sum_{a^i \in A} (\nabla_{\Theta^i} \pi^i(a^i|o^i) Q(o^i, a^i) + \pi^i(a^i|o^i) \nabla_{\Theta^i} Q(o^i, a^i)) \tag{7}$$

by product rule expansion.

$$\nabla_{\Theta^i} \pi^i(a^i|o^i) Q(o^i, a^i) + \pi^i(a^i|o^i) \nabla_{\Theta^i}(\sum_{o^{i'}, R^i} \mathcal{T}(o^{i'}, R^i(o^i, a^i)|o^i, a^i)(R^i(o^i, a^i) + V(\Theta^i, o^{i'}) \tag{8}$$

Here, eq. (8) is summed over all actions $\sum_{a^i \in A}$. Notably the value function can be used to predict a look-ahead of the next reward with a next observation $o^{i'}$ and $\mathcal{T}$ is the transition probability.

Now we construct the other agent's value estimate as a surrogate for the future collective reward. The individual reward is a constant and disappears by passing the gradient but we can isolate the collective reward as sub-objective for a sub-policy with a linearity assumption.

$$\mathbb{E}[\sum_{t=0}^{T} \mathcal{R}^j(o_t^j, a_t^j)] = \mathbb{E}[\sum_{t=0}^{T} r_t^j + r_t^c] = \mathbb{E}[\sum_{t=0}^{T} r_t^j] + \mathbb{E}[\sum_{t=0}^{T} r_t^c] \tag{9}$$

eq. (1), only r^j degenerates to 0 while r^c is differentiable w.r.t to another agent j.

To isolate the sub-objective for the collective policy, start with the reward maximization objection.

$$J(\Theta^j) = \mathbb{E}[\sum_{t}^{T} \mathcal{R}^j(o_t^j, a_t^j)] \tag{10}$$

$$J(\Theta^j) = \mathbb{E}[\sum_{t=0}^{T} r_t^j] + \mathbb{E}[\sum_{t=0}^{T} r_t^c] \tag{11}$$

by linearity in eq. (2) of $\mathcal{R}^j$.

$$J(\Theta^j) - \mathbb{E}[\sum_{t=0}^{T} r_t^j] = \mathbb{E}[\sum_{t=0}^{T} r_t^c] = J_c(\Theta^j) \tag{12}$$

$$\nabla_{\Theta^j} J_c(\Theta^j) = \mathbb{E}[\nabla_{\Theta^j} \log \pi_c^j(a^j|o^j) Q_c(o^j, a^j)] \tag{13}$$

$$\nabla_{\Theta^j} J_c(\Theta^j) = \mathbb{E}[\nabla_{\Theta^j} \log \pi_c^j(a^j|o^j)]\mathbb{E}[Q_c(o^j, a^j)] \tag{14}$$

Since the individual policy on finding the key and opening the door is assumed to be learned from eq. (3) then the agent's policies are probabilistically independent from each other.

Let $\Psi(\pi_c^j, o^j, a^j) = \frac{1}{\mathbb{E}[\nabla_{\Theta^j} \log \pi_c^j(a^j|o^j)]}$ where Ψ is the reciprocal of the expected collective policy for agent j. So we can clarify the term

$$\frac{\nabla_{\Theta^j} J_c(\Theta^j)}{\mathbb{E}[\nabla_{\Theta^j} \log \pi_c^j(a^j|o^j)]} = \nabla_{\Theta^j} J_c(\Theta^j)\Psi(\pi_c^j, o^j, a^j) = \mathbb{E}[Q_c(o^j, a^j)] \tag{15}$$

$$\pi^i(a^i|o^i)(\sum_{o^{i'}, R^i} \mathcal{T}(o_{+1}^{i'}, R^i(o^i, a^i)|o^i, a^i)(\nabla_{\Theta^i}\nabla_{\Theta^j} J_c(\Theta^j)\Psi(\pi_c^j, a^j, o^j) + \nabla_{\Theta^i} V(\Theta^i, o^{i'})) \tag{16}$$

Now in eq. (16) the correction term as a surrogate for the collective reward in the look ahead step from eq. (8).

Let $\Phi(o^i) = \sum_{a^i \in A}(\nabla_{\Theta^i}\pi^i(a^i|o^i)Q(o^i, a^i)$ for readability and Let $\rho^i(o^i \rightarrow o^{i'}) = \pi^i(a^i|o^i)(\sum_{o^{i'}, R^i} \mathcal{T}(o^{i'}, R^i(o^i, a^i)|o^i, a^i)$ for further readability.

$$\Phi(o^i) + \sum_{o^i} \rho^i(o^i \rightarrow o_{+1}^i)(\nabla_{\Theta^i} V(\Theta^i, o_{+1}^i) + \nabla_{\Theta^i}\nabla_{\Theta^j} J_c(\Theta^j)\Psi(\pi_{\Theta^j}, a^j, o^j)) \tag{17}$$

The previous, eq. (17), can then be recursively expanded out further $\Phi(o^i) + \sum_{o^i} \rho^i(o^i \rightarrow o_{+1}^i)(\Phi(o_{+1}^i) + \nabla_{\Theta^i}\nabla_{\Theta^j} J_c(\Theta^j)\Psi(\pi_c^j, a^j, o^j) + \sum_{o_{+1}^i} \rho^i(o_{+1}^j \rightarrow o_{+2}^j)(\nabla_{\Theta^i} V(\Theta^i, o^{+2}) + \nabla_{\Theta^i}\nabla_{\Theta^j} J(\Theta_c^j, o^j)\Psi(\pi_c^j, a^j, o^j))$

$$\sum_{x^i, x^j \in O} \sum_{k=0}^{\infty} \rho^i(o \rightarrow x^i, k)(\Phi(x^i) + \nabla_{\Theta^i}\nabla_{\Theta^j} J_c(\Theta_c^j)\Psi(\pi_c^j, a^j, x^j)) \tag{18}$$

Let $\eta(o) = \sum_{k=0}^{\infty} \rho^i(o^i \rightarrow o^{i'}, k)$ to clarify the transitions.

$$\sum_o \eta(o)(\Phi(o) + \nabla_{\Theta^i}\nabla_{\Theta^j} J_c(\Theta^j)) \propto \sum_o \frac{\eta(o)}{\sum_o \eta(o)}(\Phi(o) + \nabla_{\Theta^i}\nabla_{\Theta^j} J_c(\Theta^j, o^j)\Psi(\pi_c^j, a^j, o^j) \tag{19}$$

since the normalized distribution is a factor of the sum.

Then let $\sum_s \frac{\eta(o)}{\sum_o \eta(o)} = \sum_{o \in O} d(o)$

$$\sum_{o \in O} d(o)(\sum_{a^i \in A} (\nabla_{\Theta^i} \pi^i(a^i|o^i) Q(o^i, a^i) + \nabla_{\Theta^i} \nabla_{\Theta^j} J_c(\Theta^j, o^j) \Psi(\pi_c^j, a^j, o^j)) \tag{20}$$

$$\sum_{o \in O} d(o)(\sum_{a^i \in A} (\pi^i(a^i|o^i) Q(o^i, a^i) \frac{\nabla_{\Theta^i} \pi^i(a^i|o^i)}{\pi^i(a^i|o^i)} + \nabla_{\Theta^i} \nabla_{\Theta^j} J_c(\Theta^j, o^j) \Psi(\pi_c^j, a^j, o^j)) \tag{21}$$

, the log-derivative trick can pull out the gradient.

$$\sum_{s \in S} d(s)(\sum_{a^i \in A} ((a^i|o^i) Q(o^i, a^i) \nabla_{\Theta^i} \log \pi^i(a^i|o^i) + \nabla_{\Theta^i} \nabla_{\Theta^j} J_c(\Theta^j, o^j) \Psi(\pi_c^j, a^j, o^j)) \tag{22}$$

Finally, the full gradient objective from eq. (5) is constructed

$$\nabla_{\Theta^i} J(\Theta^i) = \mathbb{E}[Q(o^i, a^i) \nabla_{\Theta^i} \log \pi^i(a^i|o^i) + \nabla_{\Theta^i} \nabla_{\Theta^j} J(\Theta^j, o^j) \Psi(\pi_{\Theta^j}, a^j, o^j))] \; \square$$

P.2 Self correction term

Considering eq. (3) and eq. (4) the correction term for agent i is equivalent to the expected collective reward value estimate of

$$\mathbb{E}[\nabla_{\Theta^i} \nabla_{\Theta^j} J_c(\Theta^j) \Psi(\pi_c^j, a^j, o^j)] = \mathbb{E}[Q_c(o^j, a^j)] \tag{23}$$

In turn, the collective value estimate is an approximated prediction of the collective reward at any time

$$\mathbb{E}[Q_c(o^j, a^j)] \approx \mathbb{E}[r^c] \tag{24}$$

.

However the collective reward is also an approximate of the agent i's collective reward values estimate, if they opened their door first, which is again equivalent to the correction term of agent j

$$\mathbb{E}[r^c] \approx \mathbb{E}[Q_c(o^i, a^i)] = \mathbb{E}[\nabla_{\Theta^j} \nabla_{\Theta^i} J_c(\Theta^i) \Psi(\pi_c^i, a^i, o^i)] \tag{25}$$

Therefore, in expectation, the correction terms of both agents are equivalent and objective sharing is not necessary,

$$\mathbb{E}[\nabla_{\Theta^i} \nabla_{\Theta^j} J_c(\Theta^j) \Psi(\pi_c^j, a^j, o^j)] = \mathbb{E}[\nabla_{\Theta^j} \nabla_{\Theta^i} J_c(\Theta^i) \Psi(\pi_c^i, a^i, o^i)] \; \square$$

Very critically, this equivalence is in *expectation* and therefore is not an instance of a linear calculation or transform but the average value of one agent's correction term is the same as another when in similar context like opening their door first.