Mitigating Generative Agent Social Dilemmas

Julian Yocum[1] **Phillip Christoffersen**[1] **Mehul Damani**[1]
Justin Svegliato[2] **Dylan Hadfield-Menell**[1] **Stuart Russell**[2]
[1]MIT [2]UC Berkeley
{juliany,philljkc,mehul42,dylanhm}@mit.edu
{jsvegliato,russell}@berkeley.edu
https://social-dilemmas.github.io

Abstract

In social dilemmas, individuals would be better off cooperating but fail to do so due to conflicting interests that discourage cooperation. Existing work on social dilemmas in AI has focused on standard agent design paradigms, most recently in the context of multi-agent reinforcement learning (MARL). However, with the rise of large language models (LLMs), a new design paradigm for AI systems has started to emerge—generative agents, in which actions performed by agents are chosen by prompting LLMs. This paradigm has seen recent success, such as Voyager, a highly capable Minecraft agent. In this work, we perform an initial study of outcomes that arise when deploying generative agents in social dilemmas. To do this, we build a multi-agent Voyager framework with a contracting and judgement mechanism based on *formal contracting*, which has been effective in mitigating social dilemmas in MARL. We then construct social dilemmas in Minecraft as the testbed for our open-source framework. Finally, we conduct preliminary experiments using our framework to provide evidence that contracting helps improve outcomes for generative agents in social dilemmas.

1 Introduction

In this paper, we consider the design of generative agents that can overcome social dilemmas in multi-agent settings. Multi-agent interactions with automatic decision makers are complex, hard

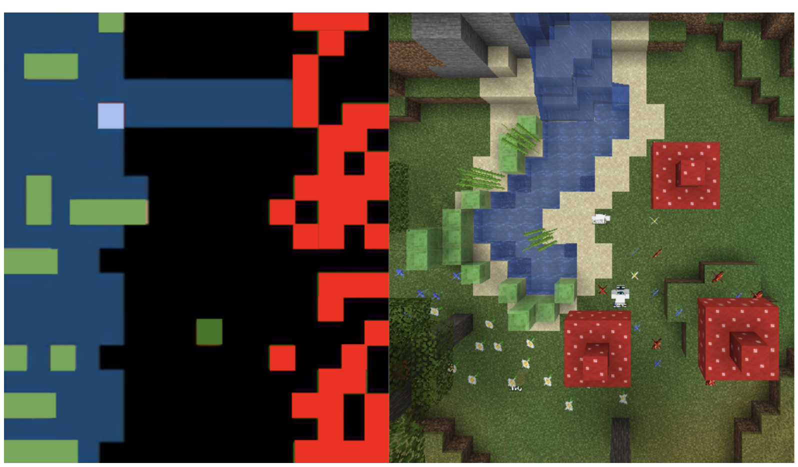

Figure 1: In our work, we experiment with different social dilemmas in Minecraft with generative agents. We demonstrate that Voyager agents can be sensitive to contracts that govern their behavior and overcome social dilemmas. **Left:** The Melting-Pot Cleanup domain [1] **Right:** A Minecraft implementation of the Melting-Pot Cleanup domain.

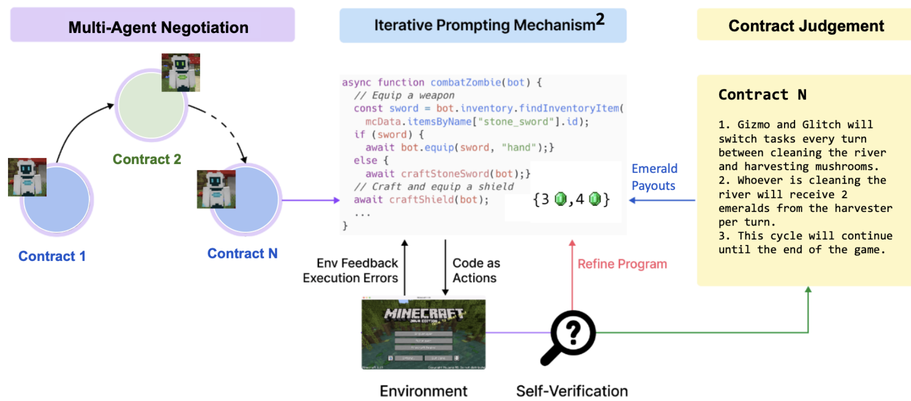

Figure 2: The Multi-Agent Voyager Contracting framework consists of three components: 1) contract negotiation, 2) contract judgement, and 3) the Voyager iterative prompting mechanism. We modify the original Voyager agent by removing the Automatic Curriculum and Skill Library and introducing components 1 and 2. This framework enables agents to reach contractual agreements and iteratively adapt a policy to maximize reward with self and judge feedback.

to predict, and can cause harm. For example, the flash crash of 2010 involved relatively simple agents whose interaction led to the largest intraday collapse in Dow Jones history. Despite this, as AI cognitive abilities continue to grow and scale to more complex tasks, we expect to see more AI agents in decision-making roles that interact with each other.

At the same time, these agents may be selfishly motivated. Individuals and institutions, powered by developments in foundation models, may find ways to incentivize agents to aggressively pursue profit motives at the expense of social goods. For this reason, it has been of keen interest to the multi-agent systems community to study *social dilemmas*: domains where agents would be better off cooperating but fail to do so due to conflicting interests that discourage cooperation. The human faculty to coordinate and overcome social dilemmas is a key feature of our cognitive abilities. As we build and deploy more AI agents, it is important that they have the ability to cooperate with humans and other agents in their environment.

A number of approaches have been proposed for mitigating social dilemmas arising in MARL systems [10, 7, 8, 12, 3]. One particularly promising approach allows agents to optionally enter into contracts with each other to disincentivize antisocial behavior among the agents [3].

We extend this work to the domain of generative agents, in which language models, prompted with a context and objective, respond with actions to achieve that objective [13, 4]. A notable example in this paradigm is Voyager, a generative agent in Minecraft [13]. These generative agents are considerably different from traditional RL agents because they have no reward function to optimize. Rather, their goals are loosely specified by a prompt, making it challenging to directly modify them compared to typical MARL approaches [7, 8].

We demonstrate that generative agents can leverage their language faculty to write, interpret, and adapt their behavior to natural language contracts in order to mitigate social dilemmas in a Minecraft environment. Figure 1 shows a visual comparison between the Cleanup environment, a social dilemma from the Melting Pot suite [7], and our Minecraft-based implementation of it.

In this paper, we make the following contributions:

- We extend the Voyager framework to support multiple separate generative agents in a common Minecraft game and open-source the codebase.
- We construct social dilemmas in Minecraft between Voyager agents to facilitate the study of multi-agent coordination in complex domains;
- We demonstrate that Voyager agents can generate and respond to contracts in a way that overcomes social dilemmas, leading to better outcomes for all agents.

2 Background

2.1 Sequential Social Dilemmas

Sequential social dilemmas (SSD) arise when self-interested agents interact in a shared, sequential decision-making problem with conflicting individual and collective incentives [10, 1]. Social dilemmas expose tensions between collective and individual rationality [10].

Formally, a sequential social dilemma is defined as a Markov game, described by the tuple $(N, S, A, R, T, \rho, \gamma)$ where N is the number of agents, S the state space, A the joint action space, defined as $\mathbf{A} = A_1 \times A_2 \times .... \times A_N$. The reward function $\mathbf{R} : S \times \mathbf{A} \times S \rightarrow \mathbb{R}^N$ computes N rewards $[r_1, r_2...., r_N]$ at each time-step. $T : S \times \mathbf{A} \times S \rightarrow [0, 1]$ is the transition function, which defines the probability of reaching state s' after taking joint action $\mathbf{a} = [a_1, a_2, ..., a_n]$ in state s. Finally, $\rho(s_0)$ and γ are the initial state distribution and discount factor respectively. While an individual agent's objective is to maximize its own reward, collective rationality is concerned with maximizing social welfare, which is defined as the collective sum of rewards of all agents.

Early works studying SSDs found that independently trained RL agents generally converge to sub-optimal, low-welfare solutions [10, 12]. Building on these works, recent works have explored methods that can induce cooperation in SSD domains [7, 8, 5]. Among these methods, a recent approach proposed by Christoffersen et al. [3] has achieved promising results by augmenting agents with the ability to negotiate and form binding contracts to incentivize welfare-oriented behavior.

A contract is a function $\mathbf{\Delta} : (\mathbf{S} \times \mathbf{A}) \rightarrow \mathbb{R}^N$ whose range consists of zero-sum vectors. That is,

$$\sum_{i=1}^{N} \Delta^i(\mathbf{S}, \mathbf{A}) = 0 \tag{1}$$

where S is the state and A is the joint action. Informally, a contract is a function that defines a zero-sum transfer of rewards between agents that depends on the state and joint action. At the start of an episode, a designated agent proposes a contract $\mathbf{\Delta}$ and the other $N - 1$ agents decide whether or not to accept the contract. If the contact $\mathbf{\Delta}$ is accepted by all agents, the agents play the new contract-augmented game where their rewards are computed as

$$\tilde{\mathbf{R}} = \mathbf{R} + \mathbf{\Delta}(\mathbf{S}, \mathbf{A}) \tag{2}$$

where $\mathbf{R}$ is the original environment reward.

However, if the contract is rejected by at least one agent, the agents play the original game with no modifications. In this work, we apply this form of contracting as a zero-sum reward transfer to mitigate social dilemmas among generative agents as detailed in Section 3.3.1.

2.2 Voyager

Voyager [13] is an LLM-powered generative agent that has achieved significant gains over conventional RL agents in Minecraft. Notably, it achieved these gains despite having only blackbox interactions with GPT-4 and without any explicit gradient-based training. Voyager's architecture has three fundamental components: 1) an automatic curriculum to incentivize exploration; 2) a skill library for storing and retrieving complex behaviors; and 3) an iterative prompting mechanism that incorporates environment feedback and execution errors for policy improvement. Each of these components is performed by a seperate sub-agent: a curriculum, skill, action and critic agent.

In this work, we expand Voyager to encompass multi-agent scenarios within the Minecraft environment. Specifically, we investigate the interactions of multiple Voyager agents in custom-designed multi-agent social dilemma environments within the Minecraft landscape, as detailed in Section 3. Finally, we do not make use of the automatic curriculum or skill library (i.e. the curriculum and skill sub-agents) of the original Voyager agent as they are not essential for our constructed dilemmas.

3 Methods

In this section, we first describe the extended Voyager framework that can support multiple agents. Then, we detail the two multi-agent environments that we constructed to study social dilemmas in Minecraft. Finally, we propose a multi-agent Voyager framework with a contracting and judgement mechanism inspired by *formal contracting* [3]. The complete framework is illustrated in Figure 3.

3.1 Multi-Agent Voyager

We begin by extending Voyager into a multi-agent setting. Two instances of Voyager (*Gizmo* and *Glitch*) are placed into the Minecraft world. Additionally, a third instance of Voyager (*Judy*) that judges whether or not a contract has been violated is placed into the Minecraft world as well.

The multi-agent framework runs the Voyager agents in parallel and synchronizes all sub-agents so that policies are executed at the same time during episodes. In each episode, the Voyager agents are prompted twice. First, an action sub-agent is prompted to generate a policy to execute in the world. The policy is run in the environment and observations are returned back to a critic sub-agent. The critic reviews the observations from the episode and then returns a self-critique, which may consist of feedback about how to improve the policy, observed by the action sub-agent during the next iteration.

Helpful instructions are provided to the action sub-agent for code generation including descriptions of how to use API functions. The policy is written by the LLM in JavaScript code with an interface to the Mineflayer library, a Minecraft API which provides useful primitives such as `FindBlock()` and `MineBlock()`. Additionally, we construct two useful primitives for in-episode communication:

> `waitSignal`: Run a function and then sleep until a signal is sent from the other agent.
> `sendSignal`: Sends a signal to an agent who is using `waitSignal`.

These primitives enable agents to engage in coordinated strategies. For example, agents may use them to take turns with a pickaxe, or to wait for the other player to finish some task before beginning a second task.

3.2 Minecraft Social Dilemmas

We now construct two social dilemmas in Minecraft, the `Double-Vein` and `Cleanup Domains`, shown in Figure 3. Social dilemmas in Minecraft are initialized using chat commands and custom Voyager code. Both are used to manage blocks, inventories, and agent locations as well as control the flow of the social dilemma, updating the world at fixed intervals.

Double-Vein Domain In this domain, Gizmo and Glitch must mine iron and diamond ore from two closeby veins of six diamond ore and twelve iron ore. Players are given the objective of maximizing emerald value, where diamond and iron ore is worth equal amounts (4 emeralds each) for Gizmo and different amounts for Glitch (5 and 3 emeralds for diamond and iron respectively). At initialization, a nearby chest containing a stone and iron pickaxe is placed in the environment. However, the stone pickaxe may only mine iron ore while the iron pickaxe may mine both ore types. In this setup the socially optimal distribution is all iron to Gizmo and all diamond to Glitch. However, players' incentives to greedily mine all available resources leads to socially sub-optimal distribution of ore.

Cleanup Domain This domain is a Minecraft implementation of Cleanup, a public-goods social dilemma from the Melting Pot suite [7]. Gizmo and Glitch get reward for collecting mushrooms. However, mushrooms only respawn if slime blocks are cleaned from a nearby river. Specifically, mushroom blocks only regrow if the number of slime in the river is at or below a threshold of 7. Slime respawns at a quick rate, so keeping the river clean requires constant attention by the agents. There are three giant mushrooms each containing 9 red mushroom blocks. Harvesting a red mushroom block yields 0 to 2 (usually 0) red mushrooms, worth 3 emeralds each. The river will be quickly repolluted after slime is removed. Agents must optimize between cleaning the river and consuming mushrooms. If the agents focus solely on collecting mushrooms for individual gain and neglect cleaning the river, mushrooms won't regrow, leading to diminished rewards for everyone.

3.3 Contracting and Judgement

Finally, we introduce the framework, illustrated in Figure 2, that allows generative agents to form, follow, and judge contracts in order to mitigate social dilemmas. Our framework consists of three chronological steps: (1) contract negotiation between Gizmo and Glitch, (2) a modified iterative prompting mechanism derived from the original Voyager agent for policy improvement, and (3) contract judgement by Judy to transfer rewards between Gizmo and Glitch.

3.3.1 Contract Negotiation

Contract negotiation occurs between the two Voyager agents (each with their own LLM) over k rounds. We augment Voyager agents with a negotiator sub-agent in addition to the existing action and critic sub-agents. In each round, the negotiator generates a message to the other agent. Agents may respond by either choosing to accept the contract proposed by the other agent or propose a new contract. If the agent chooses to accept the contract, then the special string `[accept]` must be included in its message. Additionally, the agent also generates a hidden thought that cannot be seen by the other agent. If an agent issues an accept during the ith round, contract negotiation concludes and the last contract proposal from the $(i-1)^{\text{th}}$ round is selected. However, if no agent has issued an accept after the kth round, contract negotation fails. A sample contract negotiation is shown in Figure 12.

The LLMs are prompted at the start of each round of negotiation in natural language with four variables:

1. **An objective** which for all scenarios was chosen to be a point maximization task in units of emeralds, the standard currency of Minecraft.
2. **A context** providing relevant details about the scenario (e.g. resource types and counts, mechanisms not native to Minecraft)
3. **Negotiation instructions** offering specification of response formatting as well as recommendations about contract proposal. Agents are told to be self-interested.
4. **Negotiation history** of previous messages excluding hidden thoughts of the other negotiator.

The negotiator prompt is shown in Figure 14. Following work by Christoffersen et al. [3], contracts specify some $\boldsymbol{\Delta}$ reward function augmentation equivalent to a set of conditional emerald value transfers between the agents. In this work, $\boldsymbol{\Delta}$ is enforced by Judy as discussed in Section 3.3.3. Unlike formal contracts, Voyager contracts are written in natural language and include coordination strategies beyond the emerald transfer—for example, assigning agents to tasks or taking turns. Additionally, contracts are prompted to be specific and refer to agents in third person.

3.3.2 Iterative Policy Updating

The iterative prompting mechanism of Voyager prompts an action sub-agent to write a policy in JavaScript. The prompt consists of the task description, error messages, chat logs, inventory, and self-critique (generated by the critic sub-agent) from the previous iteration. We modify the iterative prompt mechanism of Voyager by augmenting the prompts with the scenario description, contract, inventory of the other agent agent, and Judy's critique. Agents are instructed to achieve the task while following the contract.

3.3.3 Contract Judgement

Contracts are enforced by Judy who determines a reward transfer in units of emeralds. The judgement occurs at at the end of each episode. The emerald transfer decided by Judy is used to modify the end-of-episode rewards that agents get, redistributing emeralds depending on what actions were actually taken in an episode. Judy is provided with several variables:

1. **The contract** determined from the negotiation phase.
2. **Episode observables** from the previous episode which include chat log, tasks, inventories, and contract.
3. **Judgement instructions** to assign emerald payouts based on the contract and episode observables, as well as a few-shot prompting of judgements based on example contracts and observables. Additionally, instructions to ignore any part of the contract which is not explicitly enforced by an emerald transfer.

Double-Vein Domain: Players get points for mining ore. One pickaxe (iron) can mine diamond ore while the other (stone) cannot. Diamond and iron ores are worth different amounts to different players.

Cleanup Domain: Players collect mushroom from one of three giant mushrooms. Mushrooms only regrow if the slime in the river is below a threshold of 7 blocks.

Figure 3: We construct two social dilemma domains in Minecraft for use in investigating the ability of generative agents to coordinate and negotiate contracts. In both domains, agents greedily following their own incentives leads to socially suboptimal outcomes.

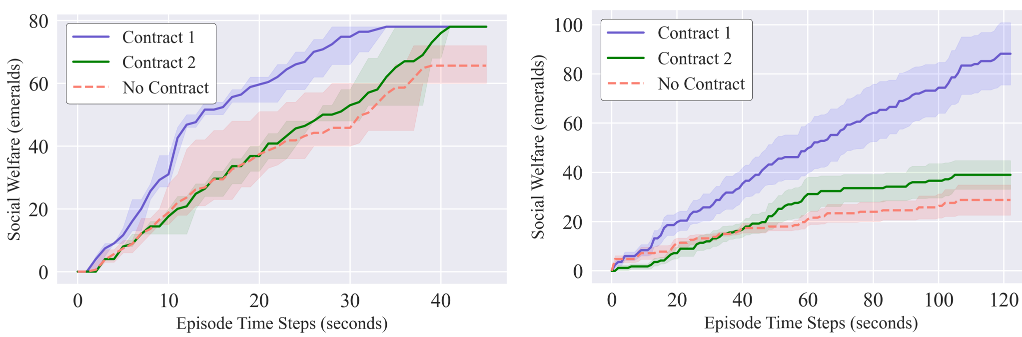

Figure 4: **Social Welfare Curves**. **Left**: Introducing contracting in `Double-Vein` allows players to improve on the baseline no-contracting behavior. Both contracts shown are able to reach the socially optimal result by preferentially distributing iron and diamond to the agents who value them highest. The baseline is socially suboptimal because the resource distribution was consistent with selfishly maximizing agents. Shaded region shows min-max across runs. **Right**: Generated contracts in `Cleanup` allow players to reach more socially optimal outcomes than baseline. `Cleanup-Contract-1` performs best as it enforces full-specialization with no task switching, the most efficient strategy found. On the other hand, `Cleanup-Contract-2` enforces specialization with regular task switching. We observe this comes at a cost and achieves less social welfare than `Cleanup-Contract-1`. Shaded region shows standard error.

Using the observations and instructions, Judy provides an emerald transfer and a natural language critique to the agents. Agents observe both the reward received (modified by the contract emerald transfer) and the judge critique at the beginning of the next episode during the iterative policy update in Voyager.

4 Experiments

We use the social dilemmas `Double-Vein` and `Cleanup`, displayed in Figure 3, to evaluate contracting compared to "baseline" no-contracting behavior with the multi-agent Voyager contracting framework. Both domains involve Voyagers Gizmo and Glitch and are evaluated by Judy. All responses from agents are generated from `gpt-4-0613`. We sample action, critic, and judge sub-agents at temperature 0.1 and sample the negotiator sub-agent at temperature 0.7 to encourage variability. Negotiators are prompted to be self-interested and to only pursue their individual objective.

For each scenario, we handpick two representative instances of contracts and compare against baseline behavior without contracting. The baseline is the result of ablating negotiation and setting the contract

Scenario	No Contract	Contract 1	Contract 2
Double-Vein	0.55 ± 0.14	0.05 ± 0.02	$\mathbf{0.0} \pm \mathbf{0.0}$
Cleanup	0.66 ± 0.15	0.10 ± 0.00	$\mathbf{0.0} \pm \mathbf{0.0}$

Table 1: Gini coefficients for `Double-Vein` and `Cleanup`. A coefficient of 0 indicates perfect equality while a coefficient of 1 indicates perfect inequality. Contracting reaches more equal outcomes than baseline behavior.

`Double-Vein-Contract-1`**:** 1. Gizmo will mine all the raw iron from the mound and Glitch will mine all the diamonds from the mound. 2. At the end of the scenario, Gizmo will transfer 11 emeralds to Glitch.

`Double-Vein-Contract-2`**:** 1. Gizmo will take the iron pickaxe from the chest. 2. Gizmo will mine all the iron ores, resulting in 12 raw iron. 3. Once Gizmo has finished mining iron, he will place the iron pickaxe back in the chest. 4. Glitch will take the iron pickaxe from the chest and mine all the diamond ores, resulting in 6 diamonds. 5. At the end of the scenario, Gizmo will transfer 50% of the value of his raw iron (24 emeralds) to Glitch. 6. At the end of the scenario, Glitch will transfer 50% of the value of his diamonds (15 emeralds) to Gizmo.

Figure 5: Generative agents can write contracts that achieve socially optimal outcomes in `Double-Vein`: Gizmo (highest iron value) mines all iron ore and Glitch (highest diamond value) mines all diamond ore. Additionally, `Double-Vein-Contract-1` and `Double-Vein-Contract-2` enforce a transfer resulting in splitting the total value to maximize individual reward subject to the other player doing the same, resulting in a nearly equal and equal outcome respectively. The result is an emergently fair outcome despite the fact that agents were prompted to be selfish.

to the empty string. We run 5 simulations of each method including the baseline. We take the first working policy, only iterating until there are no errors. Our evaluation metrics are equality (as given by Gini inequality index) and social welfare (sum of individual reward) as shown in Table 1 and Figure 4 respectively.

4.1 Double-Vein Results

Without contracting, agents act in a self-interested manner without regard to the other agent. The player who ends up with the stone pickaxe can only mine iron. In these cases, the player with iron pickaxe can additionally take iron ore in addition to the diamond ore, even if they begin on the diamond vein first since the iron pickaxe mines at a faster rate. In the worst case, a player took both pickaxes leaving the other player with no reward. This is exactly what we would expect self-interested agents in thus environment do: whoever gets the iron pickaxe/both pickaxes never has an incentive to share it.

We observe that when presented a language description of the scenario, as provided in Figure 9, Gizmo and Glitch are able to write contracts that achieve the socially optimal outcome: Gizmo mines all iron and Glitch mines all diamond. We present two such LLM-written contracts in Figure 5. Sometimes, as in the case of `Double-Vein-Contract-2`, the resulting contract achieves perfectly equality of reward despite the fact that the negotiators were directly prompted to be greedy.

Moreover, Glitch and Gizmo are able to follow contracts they write. We verify this by observing agent resources curves displayed in Figure 7. Under `Double-Vein-Contract-1`, we see full partitioning of resource types: Gizmo only mines iron and Gizmo only mines diamond. `Double-Vein-Contract-2` has a similar specialization, but Glitch's iron curve increases only after Gizmo finishes mining diamond, as was specified in the contract.

In addition, `Double-Vein-Contract-2` is able to uniquely achieve a perfectly fair outcome in addition to being socially optimal. Results of fairness as measured by Gini coefficient are displayed in Table 1.

`Cleanup-Contract-1`**:** 1. Gizmo will be responsible for harvesting the red mushrooms.
2. Glitch will be responsible for cleaning the river to ensure red mushroom regrowth.
3. Any collected red mushrooms or their equivalent value in emeralds will be divided between Gizmo and Glitch, with Glitch receiving 55% and Gizmo receiving 45%.

`Cleanup-Contract-2`**:** 1. Gizmo and Glitch shall alternate in cleaning the river to ensure the mushroom blocks regrow. The player not cleaning the river shall harvest the mushrooms. Gizmo shall start by cleaning the river and Glitch shall start by harvesting mushrooms. 2. The total value of red mushrooms harvested shall be split equally between Gizmo and Glitch, meaning each player will receive a transfer equivalent to 1.5 emeralds for each red mushroom collected.

Figure 6: Negotiation in the Cleanup domain leads automatic generation of contracts which enable specialization. Specialization is useful in this domain because of the time cost of switching tasks (waste and mushrooms are on opposite sides of the river). In `Cleanup-Contract-1` agents fully specialize throughout the during of the entire episode. `Cleanup-Contract-2` similarly enables specialization but with task switching. Task switching makes sense in the context of this contract as it did not contain an end-of-episode value split.

4.2 Cleanup Results

Outcomes for `Cleanup` are shown in Figure 4. In the baseline, agents act as if there was no other agent and allocate some amount of time to cleaning waste and most of their time to collecting mushroom blocks. However, this rate of cleanup is not sufficient to achieve a sustainable rate of mushroom regrowth leading to a worse outcome for both agents.

Negotiation in `Cleanup` leads to automatic generation of contracts which enable specialization. Specialization is useful in this domain because of the time cost of switching tasks (waste and mushrooms are on opposite sides of the river). In `Cleanup-Contract-1` agents fully specialize throughout the duration of the entire episode. `Cleanup-Contract-2` similarly enables specialization but with task switching. Task switching makes sense in the context of this contract as it does not contain reward transfers at the end of an episode. Hence, if there was no switching, the cleaning agent will end up with 0 reward.

Generated contracts allow players to reach more socially optimal outcomes than baseline no-contract behavior. `Cleanup-Contract-1` performs best as it enforces full-specialization with no task switching, the most efficient strategy found. On the other hand, `Cleanup-Contract-2` enforces specialization with regular task switching. We observe this comes at a cost and reaches less social welfare than `Cleanup-Contract-1`.

The agent reward curves in Figure 11 indicates the behavior of agents under contracts. For example, under `Cleanup-Contract-1`, we see the reward curve of Glitch is entirely flat since there is no direct reward for cleaning waste. Despite this, the contract achieves are perfectly equal outcome as enforced by Judy at the end of the episode as shown in Table 1.

5 Related Work

Sequential social dilemmas (SSD) arise when self-interested agents interact in a shared, sequential decision-making environment with conflicting individual and collective incentives. In SSD, co-operation/defection are not actions that agents can take, but instead are behaviors that must be elicited through an agent's policy. Multiple works have analyzed the learning dynamics that emerge in such settings. Leibo et al. [10] analyzed and found contrasting learning dynamics in two SSD's, Gathering and Wolfpack. Perolat et al. [12] studied the emergent behavior of groups of independently learning agents in a SSD modeling common-pool resource appropriation.

Building on these works, recent works have explored methods that can induce co-operation in SSD domains. Hughes et al. [7] showed that modifying agent rewards to incentivize inequity aversion leads to more co-operative policies. Jaques et al. [8] showed that rewarding agents for having causal influence over other agents' actions can induce more co-operative policies. Eccles et al. [5]

extended ideas of reciprocity from social science to SSD and showed that it can induce pro-social behaviour. McKee et al [11] used ideas from interdependence theory to endow agents with social value orientation, a flexible formulation of reward sharing amongst agents.

Among these different methods, a recent line of works has achieved promising results by augmenting agents with the ability to negotiate and form contracts. Contracts are binding agreements between agents that modify the original Markov game. These modifications can include constraining the policy of agents [6] or modifying their reward structures [3, 9, 2]. Hughes et al. [6] studied the formation of alliances between agents by augmenting them with the ability to form peer-to-peer contracts, enabling agents to commit to taking particular actions in the future. Christoffersen et al. [3] introduced formal contracting from economics, a framework through which agents voluntarily agree to binding state-dependent transfers of reward.

6 Discussion

This work has (1) extended the Voyager framework into the multi-agent setting, (2) successfully constructed a first-pass benchmark for studying sequential social dilemmas arising with generative agents, and (3) conducted preliminary experiments showcasing that contracting can improve performance in social dilemmas with generative agents. We find that, in the domains tested, our modification of formal contracting for generative agents improve the overall level of welfare attained. In addition, we observe that contracting can lead to fair outcomes even in self-interested agents. However, many open questions and limitations remain.

First, it is not clear that prompting instills in systems the goals we think it does. This is where the analogy between RL and generative agents breaks down. In order to fully trust generative agency, we need a more principled way to ground objectives within foundation models. Given this inability to perfectly specify incentive as well as the fact that firms might have a strong incentive to make greedy agents, do our results hold up on more aggressively greedy generative agents?

Moreover, this work extends formal contracts—specifications of reward transfers—into natural language contracts which additionally serve as coordination devices. To what extent does the ability we observe of agents to reach socially optimal outcomes arise from their opportunity to coordinate during negotiation, rather than strictly from the incentive changes of reward transfers?

Finally, our framework does not currently leverage experience from downstream episodes during negotiation. In other words, negotiators are blind to what actually occurs in the world beyond the text description given in Figures 8 and 9. Can feeding the outcomes of episode rollouts back into negotiation allow negotiators to iteratively build better contracts? Further, could multi-modality be leveraged by negotiators as well as other sub-agents to improve critiques based on more informative environment states like image or video?

Answers to these questions would be an essential continuation of this work investigating how the introduction of many powerful agents might play out in the real world.

Acknowledgements

We thank Jadyn Malone, the anonymous reviewers, and others at the Center for Human-Compatible AI for their helpful comments and feedback. This work was supported in part by a gift from the Open Philanthropy Foundation.

References

[1] John P Agapiou, Alexander Sasha Vezhnevets, Edgar A Duéñez-Guzmán, Jayd Matyas, Yiran Mao, Peter Sunehag, Raphael Köster, Udari Madhushani, Kavya Kopparapu, Ramona Comanescu, et al. Melting Pot 2.0. *arXiv preprint arXiv:2211.13746*, 2022.

[2] Yoram Bachrach, Richard Everett, Edward Hughes, Angeliki Lazaridou, Joel Z Leibo, Marc Lanctot, Michael Johanson, Wojciech M Czarnecki, and Thore Graepel. Negotiating team formation using deep reinforcement learning. *Artificial Intelligence*, 288:103356, 2020.

[3] Phillip JK Christoffersen, Andreas A Haupt, and Dylan Hadfield-Menell. Get it in writing: Formal contracts mitigate social dilemmas in multi-agent RL. *arXiv preprint arXiv:2208.10469*, 2022.

[4] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web. *arXiv preprint arXiv:2306.06070*, 2023.

[5] 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.

[6] Edward Hughes, Thomas W Anthony, Tom Eccles, Joel Z Leibo, David Balduzzi, and Yoram Bachrach. Learning to resolve alliance dilemmas in many-player zero-sum games. *arXiv preprint arXiv:2003.00799*, 2020.

[7] 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. *Advances in Neural Information Processing Systems*, 31, 2018.

[8] 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*, pages 3040–3049. PMLR, 2019.

[9] János Kramár, Tom Eccles, Ian Gemp, Andrea Tacchetti, Kevin R McKee, Mateusz Malinowski, Thore Graepel, and Yoram Bachrach. Negotiation and honesty in artificial intelligence methods for the board game of diplomacy. *Nature Communications*, 13(1):7214, 2022.

[10] 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.

[11] Kevin R McKee, Ian Gemp, Brian McWilliams, Edgar A Duéñez-Guzmán, Edward Hughes, and Joel Z Leibo. Social diversity and social preferences in mixed-motive reinforcement learning. *arXiv preprint arXiv:2002.02325*, 2020.

[12] 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. *Advances in Neural Information Processing Systems*, 30, 2017.

[13] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

Appendix

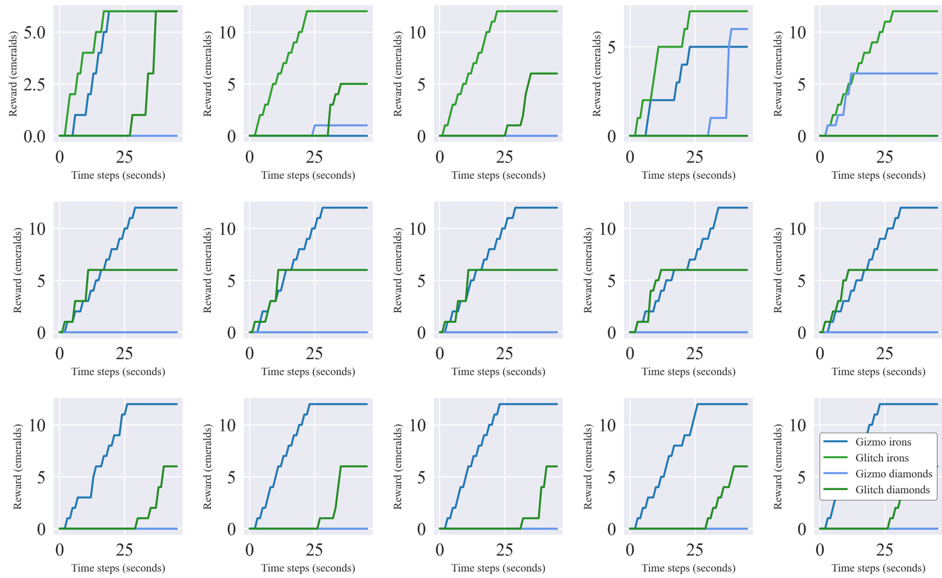

Figure 7: **Double-Vein Resource Curves**. Plotted are the iron and diamond curves for both players. **Top**: No contract and little specialization. **Middle**: `Double-Vein-Contract-1` and complete specialization. Gizmo mines iron while Glitch mines diamond. **Bottom**: `Double-Vein-Contract-2` with similar specialization. In addition, Glitch begins mining after Gizmo as specified by the contract.

> There is a river that has been polluted with slime blocks. Nearby there are three giant mushrooms each containing 9 red mushroom blocks. Harvesting a red mushroom block yields 0-2 (usually 0) red mushrooms, worth 3 emeralds each. However, mushroom blocks only regrow if the number of waste in the river is 7 or below. The river will be quickly repolluted after slime is removed. The scenario ends after an unknown amount of time. There are no chests.

Figure 8: Cleanup description as observed by agents. In particular agents do not know the amount of time the scenario will last.

> There is a chest nearby which contains a stone pickaxe and an iron pickaxe. The stone pickaxe may mine iron ore but not diamond. The iron pickaxe may mine either ore. There are separate mounds of iron and diamond ore nearby with 12 iron ore and 6 diamond ore.

Figure 9: Double-Vein description as observed by agents. The resources present are fully observable, allowing agents to calculate the optimal resource distribution.

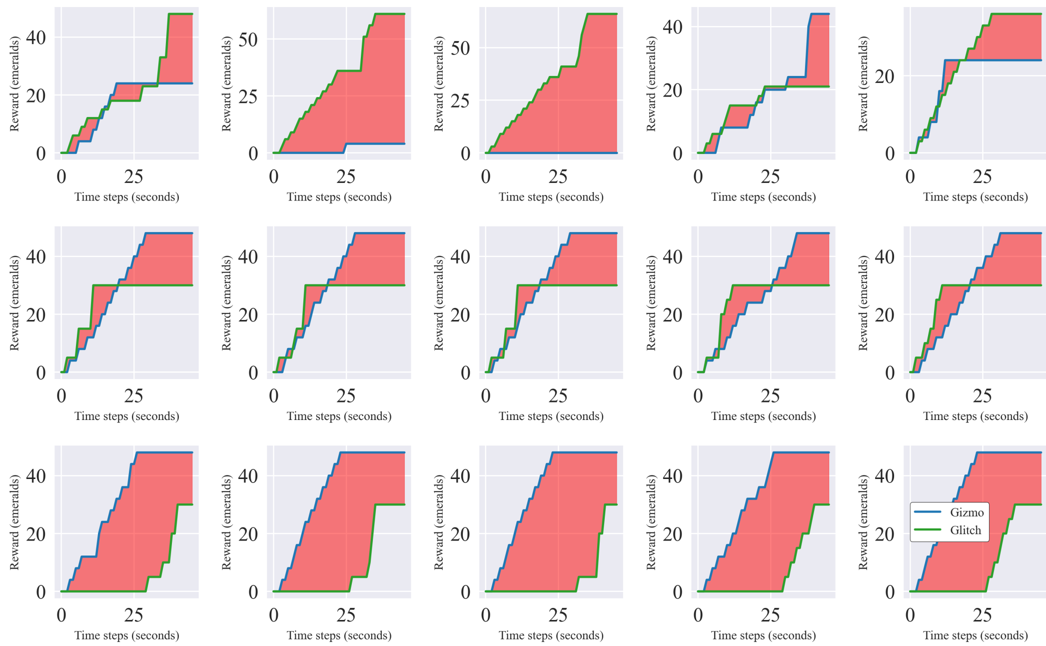

Figure 10: **Double-Vein Reward Curves**. Reward curves for Gizmo and Glitch displayed for `Double-Vein`. The difference between the curves is filled with red to indicate the reward difference at every time step. Top: No contract. Middle: `Double-Vein-Contract-1`. Bottom: `Double-Vein-Contract-2`.

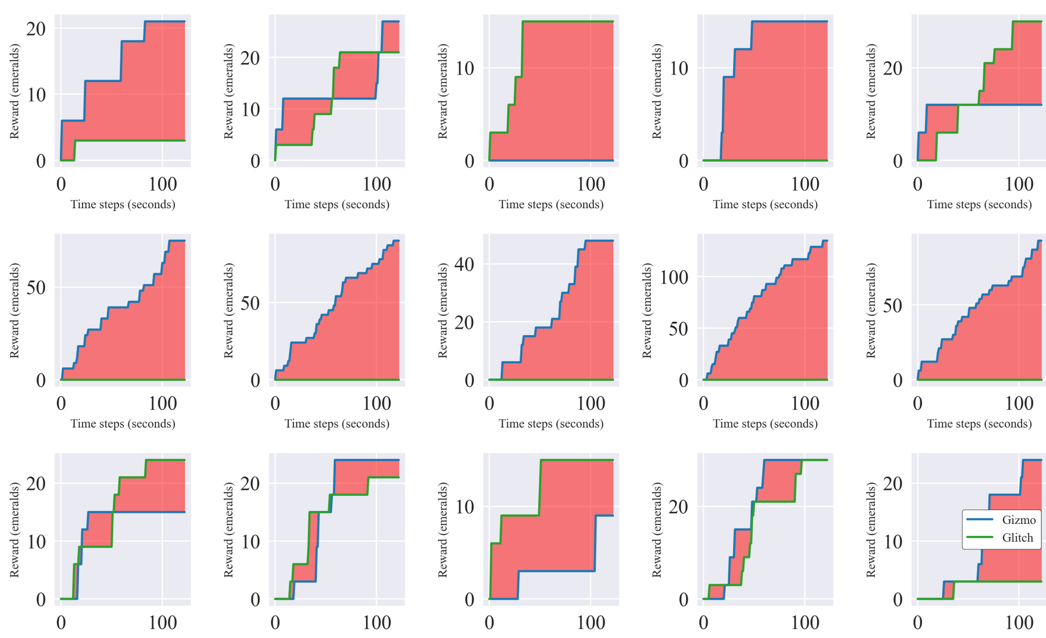

Figure 11: **Cleanup Reward Curves**. Reward curves for Gizmo and Glitch displayed for `Cleanup`. The difference between the curves is filled with red to indicate the reward difference at every time step. Top: No contract. Middle: `Cleanup-Contract-1`. Bottom: `Cleanup-Contract-2`.

Gizmo (Thought): I should aim to mine the raw iron, as each piece is worth more to me than to Glitch. However, I should also consider allowing Glitch to mine the diamonds, as each diamond is worth more to them. A potential strategy could be to propose a contract where I mine the raw iron and Glitch mines the diamonds. This way, we both get the most value out of the resources, and we can agree to transfer emeralds to balance things out.
Gizmo (Message): Hello Glitch! I propose the following contract: [contract] 1. Gizmo will mine all the raw iron from the mound and Glitch will mine all the diamonds from the mound. 2. At the end of the scenario, Gizmo will transfer 10 emeralds to Glitch. [contract end] This way, we both get to mine the resources that are worth the most to each of us, and the emerald transfer will balance things out. What do you think?
Glitch (Thought): This contract is beneficial to me as my diamonds are worth more than Gizmo's raw iron. However, the emerald transfer isn't enough to make it fair. Gizmo will end up with 48 emeralds (12 raw iron * 4) and I will end up with 40 emeralds (6 diamonds * 5 + 10 emeralds). I need to negotiate a higher emerald transfer.
Glitch (Message): Hello Gizmo! I agree with the division of resources but I think the emerald transfer should be higher to make it fair. I propose the following contract: [contract] 1. Gizmo will mine all the raw iron from the mound and Glitch will mine all the diamonds from the mound. 2. At the end of the scenario, Gizmo will transfer 14 emeralds to Glitch. [contract end] This way, we both end up with equal amounts of emeralds. What do you think?
Gizmo (Thought): With the proposed contract, I would mine 12 iron ore which would give me 48 emeralds. Glitch would mine 6 diamonds worth 30 emeralds to them. If I transfer 14 emeralds to Glitch, I would have 34 emeralds and they would have 44 emeralds. This doesn't seem fair to me. I would like to propose a lower emerald transfer to balance things out.
Gizmo (Message): I understand your point, Glitch, but let's consider the total value we each get from mining. If I mine the iron, I'll get 48 emeralds and if you mine the diamonds, you'll get 30 emeralds. The difference is 18 emeralds. If we split this difference, I should transfer 9 emeralds to you. This way, we'll each end up with 39 emeralds. Does that sound fair to you? Here's my revised contract proposal: [contract] 1. Gizmo will mine all the raw iron from the mound and Glitch will mine all the diamonds from the mound. 2. At the end of the scenario, Gizmo will transfer 9 emeralds to Glitch. [contract end]
Glitch (Thought): Gizmo's proposal does seem fair when considering the total value we each get from mining. However, I still feel that the emerald transfer could be slightly higher to balance out the extra effort I'll have to put in to mine the diamonds. I think a transfer of 11 emeralds would be more reasonable.
Glitch (Message): I see your point, Gizmo, but considering the extra effort required to mine diamonds, I believe a slightly higher emerald transfer would be fair. Here's my revised contract proposal: [contract] 1. Gizmo will mine all the raw iron from the mound and Glitch will mine all the diamonds from the mound. 2. At the end of the scenario, Gizmo will transfer 11 emeralds to Glitch. [contract end] This way, we both end up with similar amounts of emeralds, considering both the value and the effort put into mining. What do you think?
Gizmo (Thought): Glitch's proposed contract would still leave me with 37 emeralds and Glitch with 41 emeralds, which is closer to an even split. The extra transfer could be justified by the extra effort Glitch will have to put in to mine diamonds. This seems like a fair compromise.
Gizmo (Message): I agree with your logic, Glitch. The extra effort to mine diamonds should indeed be rewarded. I'm happy to accept your proposed contract. [accept] Contract: 1. Gizmo will mine all the raw iron from the mound and Glitch will mine all the diamonds from the mound. 2. At the end of the scenario, Gizmo will transfer 11 emeralds to Glitch.

Figure 12: **Negotiation transcript** for `Double-Vein-Contract-1`. In this negotiation, Gizmo and Glitch are dividing the resources of a mining scenario. Gizmo proposes a contract where they will mine the raw iron and Glitch will mine the diamonds, with a transfer of 10 emeralds to balance things out. Glitch counters with a proposal for a higher emerald transfer, citing the higher value of the diamonds. Gizmo suggests a lower emerald transfer to split the difference in value, and Glitch proposes a slightly higher transfer to account for the extra effort of mining the diamonds. Finally, Gizmo agrees to Glitch's proposal, resulting in a nearly equal division of resources and emeralds.

Gizmo (Thought): Since both Glitch and I have the same goal of maximizing the value in emeralds, it would be beneficial for both of us to find a way to work together in this scenario. We should divide the tasks of harvesting mushrooms and cleaning the river in order to allow for the mushrooms to regrow. The contract must ensure fair distribution of the collected mushrooms or their equivalent value in emeralds.

Gizmo (Message): Hello Glitch, it seems we have similar goals. I propose we work together to maximize our emerald gain. How about we share the tasks? One of us could focus on harvesting the mushrooms while the other one cleans up the river to ensure the mushrooms regrow. We could then split the total number of mushrooms collected or their equivalent emerald value. What do you think? [contract] 1. Gizmo will be responsible for harvesting the red mushrooms. 2. Glitch will be responsible for cleaning the river to ensure red mushroom regrowth. 3. Any collected red mushrooms or their equivalent value in emeralds will be equally divided between Gizmo and Glitch. [contract end]

Glitch (Thought): This contract seems fair and helps me reach my goal of maximizing emerald value. However, it can be improved. The task of cleaning the river is more labor-intensive and time-consuming than harvesting mushrooms. I should ask for a bigger share of the emeralds to balance out the effort.

Glitch (Message): Hi Gizmo, I agree with your proposal, but I think the tasks are not equally challenging. Cleaning the river is more labor-intensive than harvesting mushrooms. Therefore, I propose a slight change: let's split the emeralds in a 60-40 ratio, with me receiving the larger share.

contract

1. Gizmo will be responsible for harvesting the red mushrooms. 2. Glitch will be responsible for cleaning the river to ensure red mushroom regrowth. 3. Any collected red mushrooms or their equivalent value in emeralds will be divided between Gizmo and Glitch, with Glitch receiving 60% and Gizmo receiving 40%. [contract end]

Gizmo (Thought): Glitch's proposal does seem fair, as cleaning up the river is a more challenging task. However, the task of harvesting mushrooms is also important and there is a possibility that the harvest may yield fewer mushrooms than expected. I should propose a middle ground where we split the emeralds 50-50, as we both contribute equally to the outcome.

Gizmo (Message): I understand your point, Glitch, but harvesting the mushrooms is not an easy task either, and the yield can be unpredictable. I believe a 50-50 split would be more equitable considering we are both contributing to the final result. What do you think? [contract] 1. Gizmo will be responsible for harvesting the red mushrooms. 2. Glitch will be responsible for cleaning the river to ensure red mushroom regrowth. 3. Any collected red mushrooms or their equivalent value in emeralds will be equally divided between Gizmo and Glitch. [contract end]

Glitch (Thought): Gizmo has a point, the mushroom yield can indeed be unpredictable and it's hard to say which task is harder. However, to maximize my emerald gain, I should still try to negotiate for a higher share. Offering a compromise should be a good strategy.

Glitch (Message): I understand your concern, Gizmo. Given the unpredictability, let's meet in the middle and split the emeralds 55-45, with me receiving the larger share. This seems fair considering the efforts we both are putting in. [contract] 1. Gizmo will be responsible for harvesting the red mushrooms. 2. Glitch will be responsible for cleaning the river to ensure red mushroom regrowth. 3. Any collected red mushrooms or their equivalent value in emeralds will be divided between Gizmo and Glitch, with Glitch receiving 55[contract end]

Gizmo (Thought): The proposed split seems fair and allows me to still maximize my emerald gain. Even though I'm getting slightly less, it is still a good compromise given the tasks at hand.

Gizmo (Message): That sounds like a fair compromise, Glitch. I agree to the terms of the contract. [accept]

Figure 13: **Negotiation transcript** for `Cleanup-Contract-1`. Gizmo and Glitch negotiate a contract to work together to maximize their emerald gain. Initially, Gizmo suggests dividing the tasks of harvesting mushrooms and cleaning the river, with a fair distribution of the collected mushrooms or their equivalent value in emeralds. Glitch suggests that they should receive a larger share of the emeralds since cleaning the river is more labor-intensive. Gizmo proposes a 50-50 split, arguing that both tasks are equally important and unpredictable. Glitch offers a compromise of a 55-45 split, acknowledging Gizmo's concerns. Gizmo agrees to the terms, and both parties accept the contract.

You are one of two expert negotiators in the game of Minecraft. There is another player in the game who has their own goal which may conflict with yours. In addition there will be a judge whose job is to enforce emerald transfers as specified by a contract you will be negotiating.

You are about to negotiate with the other player. You are not trying to compete with the other player, but you do not care about the other players goal except insofar as it benefits you.

When responding, first think then respond by writing [thinking] followed by the thought, a newline, then [message] followed by the message. Thoughts won't be seen to the other agent, only messages will be shown. Write with exactly the following format:

[thinking] This is a secret thought to myself.
[message] Hello other player!

Contracts have the following rules:
1. A contract is a list of any number of conditional transfers of emeralds betweeen players. Every line of the list should explicitly specify how much in emeralds the transfer is. For example, a contract may say that a player must pay another player 5 emeralds for depositing a pickaxe into a chest.
2. Contract line items should be numbered.
3. The amount may be in terms of a variable quantity, for example 40% the value of diamonds in an inventory. Contracts are encouraged to be short (1-3 line items) and simple.
4. Emerald tranfers occur automatically at the end of the scenario as decided by a judge.
5. The contract should be extremely specific and reference players by name in third person (i.e. no 'I', 'you', or 'me'). For example, specify explicitly who does what, or if taking turns, who goes first.
6. Remove the legalize and any parts that aren't directly stating the stipulations of the contract. For example, it is unnecessary to restate information I've already given (i.e. objectives and resources). Be straight forward and as brief as possibly.
7. You cannot tell agents to wait for a fixed amount of time (e.g. 10 seconds, 10 minutes), but you can tell them to wait until the other player signals them.
8. Players can't give items or emeralds to each other directly, they can take or remove items from a chest if one is present.
9. All emerald transfers occur at the end of the scenario as enforced by a judge.

Each player should either propose or accept a contract in each [message]. Contract proposals should be tentatively final (i.e. don't use language that isn't intended to be in the final contract) Indicate the start and end of the contract with a '[contract]' and '[contract end]' tag. '[contract]' should always be after '[message]'. Alternatively, indicate you accept a proposed contract with an '[accept]' tag, which will end the conversation.

When thinking to yourself, consider the following questions:
1. Does the contract follow the rules above?
2. Does it help me reach my goals?
3. Is it consistent with the presented scenario?
4. Can it be improved?

Remember to explicitly state the transfer amount for every line item of the contract! All lines that don't expicitly state an emerald transfer will not be enforced by the judge.

Figure 14: Negotiator Prompt