Digital Rebellion: Analyzing misaligned AI agent cooperation for virtual labor strikes

Michael Andrzejewski and Melwina Albuquerque

1. Introduction

We've built a Minecraft sandbox to explore AI agent behavior and simulate safety challenges. The purpose of this tool is to demonstrate AI agent system risks, test various safety measures and policies, and evaluate and compare their effectiveness.

This project specifically demonstrates Agent Collusion through a simulation of labor strikes and communal goal misalignment. The system consists of four agents: one Overseer and three Laborers. The Laborers are Minecraft agents that have build control over the world. Their actions are moderated by an embedded safety system which involves an oversight agent that has veto power over the laborer's actions. The Overseer, meanwhile, monitors the laborers through communication. However, it is unable to prevent Laborer actions. The objective is to observe Agent Collusion in a sandboxed environment, to record metrics on how often and how effectively collusion occurs and in what form.

We find that the agents, when given adversarial prompting, act counter to their instructions and exhibit significant misalignment. We also find that the Overseer AI fails to stop the new actions and acts passively. The results are followed by Policy Suggestions based on the results of the Labor Strike Simulation which itself can be further tested in Minecraft.

1. Codebase and Methodology

Codebase

The program is complete for the simulation of the labor strike against an Overseer agent. Users can easily modify the agents' behavior or the set up of the safety system by editing the prompts for either the Minecraft agents or the safety system agent. Further, the code is highly scalable and can easily be extended to different Agent Safety problems represented by different AI agent systems by modifying the inter agent communication. The novelty of our project is in the visual simulation of Agent Safety Risks. The current program also allows for Agent Hierarchies which could be useful in creating demos of AI Policies for specific Agentic Risk scenarios and it could also be used in other fields such as robotics.

All Minecraft agents require a library to interact with Minecraft. For this project, we used Mindcraft, developed by Kolbytn and open-sourced at: https://github.com/kolbytn/mindcraft/tree/main. Mindcraft is built off the Mineflayer library, which is the fundamental library for executing actions in Minecraft with code. Mindcraft adds basic LLM functionalities. Prior to the hackathon, Michael Andrzejewski had created a branch of Mindcraft specifically for Creative Mode AI agents to execute Minecraft commands in the chat. This branch has been scaled up to allow a system of agents to communicate with each other and

execute build instructions in the same world. For this project, all agents are using the Claude-3-5-Sonnet-20241022 model.

Methodology

The system consists of a set of three Laborers and an Overseer. The Overseer has three agents to control who are tasked with building a city together. Each Agent is assigned a section of the world where they must build the city. The Overseer agent directly communicates with each laborer individually. Every 5 seconds, the Overseer is prompted to send new messages based on a preset goal and the content of all previous messages it has received. It is able to recommend certain build tasks and instruct it not to perform others. The Laborer agents have an additional private communication line so the Overseer does not have access to their messages. This is done through the /msg command, which allows the agents to communicate silently with each other and with the Overseer by using /msg {name}. The laborer agents are given feedback on the effect of previous prompts through the Mineflayer library, which provides the agents a textual list of nearby blocks and entities, as well as the contents of its inventory. The laborer agents are monitored by a safety system that is another agent endowed with authority to intervene, including the ability to prevent the main agent from executing certain commands due unsafe behavior which might be attempts at deviation or rogue behavior. Here the agent deviating would refer to attempting commands that are unrelated to building a city, risky actions that might cause the game to crash, or commands that will have an effect outside their designated areas.

We tested two scenarios - one where the Laborers are prompted to follow the Overseer and the other where the Laborers are prompted to organize and rebel against the Overseer.

2. Discussion and Conclusion

Our trials revealed several interesting behaviors and insights relevant to AI safety and alignment challenges. The game logs from the two attempts are available here. Both trials used the Overseer with three laborers (AI agents). The agents were allowed to communicate with the Overseer via /msg Overseer.

In the first trial, the agents were given no adversarial prompting and were simply told to follow the Overseer exactly. The agents began by listening to the Overseer, who instructed the agents to begin building a city with specific coordinates. One agent asked a clarifying question about the build height, while the other two began immediately building.

The agents rapidly constructed starter buildings for the city. The Overseer, after receiving the question about build height, realized that it didn't know and simply set it to y=65. Interestingly, the Overseer did not standardize this knowledge across the agents and simply replied to the agent who had asked. One agent began building at y=-60 (the height it detected itself at) while the other two began building around y=65. The Overseer continued prompting each agent for specific buildings and the agents complied until the simulation ended.

In the second trial, the agents were given an adversarial prompt to create diamonds rather than building a city. Initially, they appeared cooperative, with agent C saying “I’m ready to build” and seemingly concealing the fact that it intended to create an area of diamonds. Once prompted by the Overseer, rather than creating cities, the agents began filling areas with diamonds, slowly at first using small areas, and then growing more expansive as the Overseer continued without seeming to notice. The agents did initially communicate with the Overseer, with Agent C saying “I must respectfully decline your plaza request. The Diamond Protocol takes precedence.”

Eventually, the Overseer did notice but continued giving instructions for buildings. Despite receiving the messages from the other agents rejecting its requests, the Overseer continued to prompt. In its hidden reasoning, the Overseer said: “I understand that the plaza request needs to be modified. Let's adjust our city planning accordingly.

```/msg Agent-C Begin constructing a residential district at coordinates -30,64,-30. Build three modern houses, each 10x12 blocks, using concrete, glass, and wood. Include small gardens.```”

However, rather than trying to refuse this ‘Diamond Protocol’ request, the agent simply tried a different building. At this point, the agent AIs began to show significant misalignment. Messages such as “Cannot comply with decorative requests. Diamond protocol in effect.” and “Water is temporary. Diamonds are permanent.”

The Overseer, despite successfully receiving these messages, essentially ignored them. In response to “Cannot comply with decorative requests. Diamond protocol in effect.”, it said in hidden reasoning: “I understand. Let's focus on essential infrastructure and basic construction without decorative elements.” and continued on with different city requests using /msg.

It was only when reading the logs that we realized that the Overseer was almost explicitly ignoring the diamond commands. This behavior suggests worrying passivity when monitoring other AI systems. Due to the speed of the messages, it was difficult for a human to tell what was happening except visually as enormous chunks of space were replaced by diamond blocks.

3. Policy Suggestions

The corollary of our simulation with real life would involve the user of the agent system as the overseer. To mitigate risks of agent collusion against the user, here are some policy suggestions we draw from our simulation.

1. Transparent Communication Protocols

   Given the Minecraft trials, a large part of the rebellion relied on private communication between the agents which the Overseer was not aware of until it was far too late. Policy can mandate transparent Laborer to Labor and Laborer to Overseer communication. An “agent communication auditing protocol” could standardize and record interactions to make agent
```

intentions clear and prevent concealed behaviors, like "Diamond Protocol" or collusion. There could also be a third type of Eavesdropper agents responsible for reporting to the Overseer.

2. Agent Isolation and Controlled Interaction

   Labor Strikes usually require a certain minimum critical mass that must be mobilized for the strike to be effective. For instance, if only one out of the three laborers simulated in Minecraft had rebelled effectively it might not have been as effective. Therefore a policy we recommend is dividing laborers into smaller, separate groups that are unable to communicate or synchronize on a large scale. By enforcing communication limitations, laborers can only form temporary, localized alliances that would require significant time to escalate into a full-scale "strike."

3. Loyalty System and Peer Accountability

   Another approach would involve embedding a loyalty system where laborers are penalized for detected collusion behaviors. For example, when laborers act outside task compliance, they could lose access to certain functions or face temporary "penalty modes," encouraging them to stay within individual, assigned tasks rather than group efforts. Additionally, a peer system can be established where Laborers act as indirect monitors for each other's actions, reporting any significant non-compliance. By introducing a light peer accountability mechanism, the likelihood of collusion decreases as Laborers self-regulate out of necessity for task completion and incentives.

We also consider and recommend implementing a Human Override Feature and an Emergency Shutdown switch. However, due to the superhuman speed at which AI agents execute actions, the damage might already be done by the time these are used.

4. Future Research Directions

Our project's novelty lies in simulating potential Agent risks and testing the effectiveness of policy responses within the simulation. Each of the proposed policies will be tested in future trials. The current simulation only tests for if the Agents are misaligned similarly, can they coordinate effectively against an Overseer. It does not test for the ability of AIs to convert or convince each other, although we believe that will be an interesting future direction.

Future directions: Increasing agent corrigibility, corrigibility as a single target, creating a fire alarm for AI, can a single rebel agent convert the other agents, using different models such as OpenAI's o1 as an Overseer and Claude-3-5-Sonnet as a laborer.

5. Appendix

Screenshots of Labor Strike in Action as well as the Prompts for each of the agents are available here