The Impact of Social Norms in the Among Us Game

ANDREIA PEREIRA, Instituto Superior Técnico, Portugal
PEDRO NUNES, Instituto Superior Técnico, Portugal
VASCO RODRIGUES, Instituto Superior Técnico, Portugal

The Among Us game had a popularity increase in 2020, when friend groups joined in video calls to play together. The game explores ideas of Cooperation and Defection in a spaceship environment, where an impostor agent surges among the spaceship crew, with the goal of sabotaging the space mission. In this work, we will attempt to analyse how different social norms within the spaceship can affect the survival of the spaceship crew.

1 INTRODUCTION

1.1 Motivation

The simulation of different social norms is a widely studied topic, mainly in game theory applications. It seems interesting to join that area to that of multi-agent systems, where agents have goals and can achieve them by cooperating with each other.

The Among Us game [1] became widely known last year. Success in the game requires that players build trust in each other to complete the game objectives.

The importance of **trust** in the game makes it interesting to study with this approach. In Game Theory, social norms are used to define agent behaviour in a multi-agent setting, and are closely related with *reputation maintenance*, with the premise that agents with better reputation succeed while others may fail. Our motivation is to *simulate* a multi-agent game that resembles the Among Us game, and to study how different social norms may lead to success or defeat.

1.2 Problem Definition

Among Us is an online multiplayer game that takes place in a spaceship. The spaceship has several rooms where tasks are to be executed. A *space crew* inhabits the ship and must complete tasks to maintain the spaceship and the mission. In this game, players can have one of *two roles*: **Impostor** or **Crewmates**. There is only one Impostor - but the Crewmates have no idea who he is, as all players look the same.

1.2.1 Crewmates. Crewmates are the majority of the players. Their goals are:

- To survive, by avoiding being killed by the Impostor.
- To complete tasks to sustain the mission. These tasks are assigned to each Crewmate and are to be completed in several rooms of the spaceship.
- To find the Impostor in order to eliminate them.

Crewmates win the game if they complete all the tasks or if the impostor is eliminated.

Authors' addresses: Andreia Pereira, Instituto Superior Técnico, Lisbon, Portugal; Pedro Nunes, Instituto Superior Técnico, Lisbon, Portugal; Vasco Rodrigues, Instituto Superior Técnico, Lisbon, Portugal.

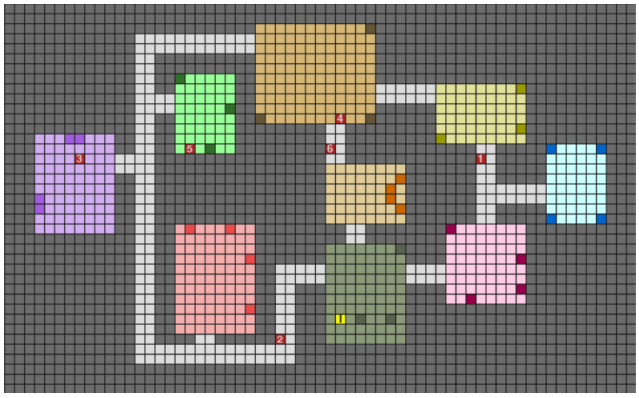

Fig. 1. Our simulated spaceship.

1.2.2 Impostor. The Impostor's goal is to kill all the Crewmates before they can complete all the tasks. The Impostor must be careful to do it in order to not be caught. They win if there is only one Crewmate left alive.

1.2.3 Elimination Mechanics. Regular space crew meetings take place to vote someone out. Each player votes on who they think the Impostor is, and the player with the most votes is eliminated.

1.3 Requirements

The environment of this project is the grid presented in Fig. 1, and the agents will need to know how to navigate it. To do so, we're using a *path-finding* algorithm - *Dijkstra* - so agents that do not know the entire map can find the shortest path to their destiny.

1.4 Objectives

Our main goal is to implement a simulation where we can define what social norm the players should use - this is, how an agent should update the reputations of the other agents.

2 APPROACH

2.1 Environment

Our environment is the spaceship, which is represented in Fig. 1. The spaceship has *nine rooms*, which are interconnected via *tunnels*, represented in white. The grey zones outside of the tunnels and rooms are the walls of the spaceship, so agents cannot move across them. There are also specific highlighted zones which represent the tasks in each room.

The environment is:

- **Semi-Accessible** - The Crewmates know about the full map of the spaceship but don't know who the Impostor is or where a dead Crewmate might be.

- **Deterministic** - Each action has a single guaranteed effect.
- **Dynamic** - The Impostor may kill a Crewmate while another Crewmate is updating their internal state.
- **Discrete** - There is a finite number of possible actions and sensors for both Crewmates and the Impostor.

2.2 A Multi-Agent System

We propose the following agents for this problem:

- *Crewmates*, represented as *red squares*. *Crewmates* have *limited visibility* of the environment - they can only see in a radius of 2 squares around him, which means that they do not know where other *Crewmates* can be.
- An *Impostor*, represented as a *yellow square*. The *Impostor* has full visibility of the space ship and can always see where the *Crewmates* are.

Both *Crewmates* and the *Impostor* are identified by a number (*ID*). This will allow each agent to identify other agents and accordingly update reputations. Impostor's *ID* is always the number of agents playing the game (*eg*: if there are 7 agents playing the game, the *Impostor*'s *ID* is the number 7, even though they are represented in the grid as a yellow square with an *I* in the center). In terms of agent properties, we have:

- **Reactivity** - If Crewmates see another agent being killed, they call a meeting in order to vote the Impostor out.
- **Proactiveness** - The Impostor acts as a Crewmate and identifies where opportunities to eliminate Crewmates arise.
- **Social Ability** - Crewmates coordinate themselves in the spaceship meetings, by sharing their reputation beliefs and then using that information to vote (or not) some agent out.
- **Rationality** - Crewmates will act in a way that minimizes their chance of being falsely accused as well as the chances of being killed.

2.3 Architecture

We predict that the best results will be achieved by an *Hybrid Architecture*, that combines *Stateful* and *Utilitarian* architectures.

The agents keep an *internal state* that combines two modules: the *Tasks* an agent still has to fulfill and the *Reputations List* an agent maintains about the other agents. The two following sections will go into more detail about these two models that define the *internal state* of an agent.

In Table 1 we sum up what our *Sensors* and *Actuators* will be, that follow the *Hybrid Architecture* we will implement.

2.4 Reputations and Social Norms

Agents will maintain a *list of reputations*, or *beliefs*. They will have one belief for each other agent. These beliefs values are supposed to represent how much the agent trusts other agents, and are then used to decide on how to act. In practice, the internal state of a Crewmate is formed by their beliefs about other agents as well as their uncompleted tasks.

Reputation updates can happen in multiple moments, depending on the chosen *social norm*. it is the *social norm* of each simulation that defines what agents value the most, which will reflect on the kind and severity of *reputation* updates they perform. These *reputations* mimic intelligent behaviour and make the simulation more interesting than a random one. This is explained in more detail in section 3.4.1.

Agents	Sensors	Actuators
Crewmate	isImpostor isTask scanGround	move do_task die callVoting vote update_reputation
Impostor	isImpostor isClose scanGround	move kill vote fake_task update_reputation

Table 1. Agent Sensors and Actuators

3 ARCHITECTURE DETAILS

Our architecture consists of *two blocks*: the main execution block, and the classes block. The classes module, describes the following *three classes*: *Agent*, *Crewmate*, and *Impostor*. *Agent* is a superclass that holds methods and information that are common to both Crewmates and the Impostor.

In Fig. 2 we present a brief scheme of the classes, useful for the next section, where we describe the main workings of the simulation. we will present how the *belief updates* work and in which situations they occur. we will also explain the voting mechanism as well as the global functioning of both types of agents.

3.1 Tasks

A *task* will consist of going to a specific zone in a certain room for a certain amount of time, and can be defined as $\{taskId, roomId, numSeconds\}$.

Each room in the spaceship has a set of *highlighted positions*, in which a task specific to that room can be completed. When a Crewmate selects a task to complete, it selects one of the possible positions where that task can be completed, and goes there. If that position is occupied, then it selects another possible location for said task.

Each *Crewmate* will be assigned a set of tasks, and we will vary the number across different simulations. Once all *Crewmates* finish their tasks, the game ends and the *Crewmates* win, even if they have not found who the *Impostor* is.

An *Impostor* may have a set of **fake tasks** to achieve. This is done when the Impostor cannot kill any Crewmate. The functioning of the fake tasks is the same as the Crewmates' tasks.

3.2 Crewmates' Planning

The crewmates' *plan* function simply consists of picking the task they want to complete next, and finding a route to the room in which the task takes place.

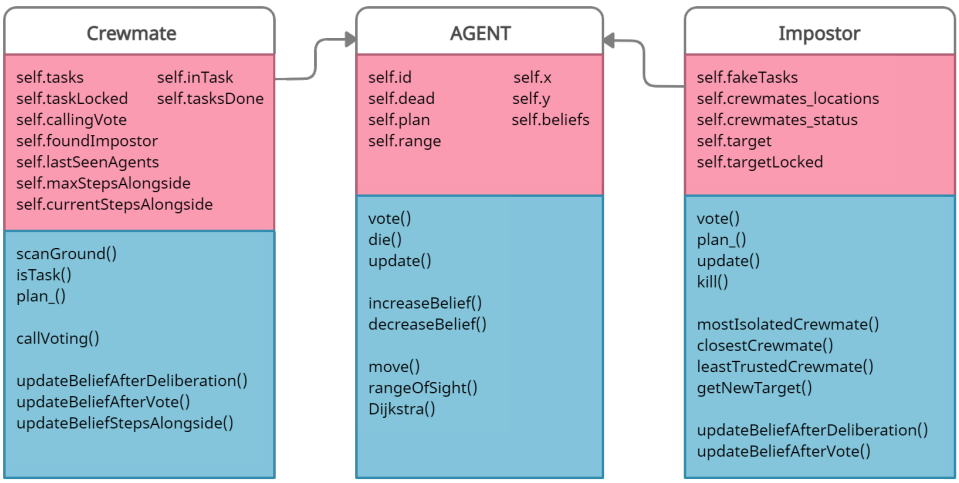

Fig. 2. A scheme of class attributes (pink) and methods (blue)

3.3 Impostor's Planning and Heuristics

We propose *three distinct heuristics* according to which the Impostor decides their *next target*. The heuristics are:

- The **Most Isolated** Crewmate
- The **Closest** Crewmate
- The **Less Trusted** Crewmate

The implementation of this decision is done in the *getNewTarget* method. What we chose to do was to have all the *heuristic functions* return an ordered list of Crewmate's ID's. Then, we simply use the position in each of the three lists to calculate a *global score* for each Crewmate. The Impostor chooses the Crewmate with the *smaller score* to be their new target. This is more explicit in Fig. 3:

Fig. 3. An example of the functioning of the Impostor's planning logic

Once the Impostor fixes a target, it plans a route to reach said target. However, as said previously, the Impostor might not be able to select a new target yet. This happens when the Impostor enters their *kill cooldown period*. We implemented this cooldown period so that the Impostor would lay low after killing another agent. In this situation, the Impostor's plan switches to completing **fake tasks**.

3.4 Belief Updates as a Deliberative Component

Every agent, including the Impostor, has a set of *beliefs*, one for each *other* agent. We represent *beliefs* as a value in the range of $[0, 1]$. For example, if agent 1 has the belief of agent 2 as 0, then we can say that agent 1 does not trust agent 2 at all.

In practice, an agent's belief list will look something like this. We assume there are 7 agents, and that this agent's ID is 1. Then:

$$beliefs_1 = \{2 : 0.5, 3 : 0.5, 4 : 0.7, 5 : 0.8, 6 : 0.3, 7 : 0.3\}$$

These beliefs are the main **deliberative component** of our agents. The intelligent behaviour will emerge from the updates they can make and how they use the knowledge to find the Impostor.

3.4.1 Crewmates' Belief Updates. In the Crewmate's logic, the beliefs are used to find the Impostor and vote them out. There are three situations in which a Crewmate can update their beliefs of other agents:

- In the **Deliberation**: Here, the Crewmates update their own beliefs using the other agents' beliefs. They value more the opinion of an agent they trust the most, and vice versa. The increase factor is 0.01 in the base version.
- In a **Voting Session**: Here, the Crewmates increase the trust of an agent that agrees with them (by a factor of 0.1), and decrease the belief of an agent that votes for them (by a factor of 0.2 as well).
- When **walking alongside** another agent: Crewmates keep a *max steps alongside* list, with the information of the longest paths they have walked alongside every other agent. They

then regularly increase the beliefs of agents that have accompanied them, as they are most trustworthy. This increase is done by a factor of 0.01, in the base version.
- When they **find an agent killing** another one: They immediately update the *Impostor's* belief to the lowest value possible (0) and call a spaceship meeting to kick the *Impostor*.
- When they find an isolated **dead body**: They lower the belief(s) of the last agent(s) they have seen (by a factor of 0.1), as they are the most suspicious.

One important note is that, as previously mentioned, the environment is only *semi-accessible* for the Crewmates. This means that although they know the map, they do not know where other agents are, unless they are within their range of sight. To do this, we manually implemented a *rangeOfSight* method that ignores vision beyond the walls of the spaceship and returns all the positions a Crewmate can observe around them, at a certain time.

The base value for the range of sight is a *radius* of 2 grid cells around the current position of a Crewmate.

3.4.2 Impostor's Belief Updates. In the Impostor's logic, the beliefs are used to find who trusts them the least. There are two situations in which an Impostor can update their beliefs of other agents:

- In the **Deliberation**: When the Crewmates share their beliefs, the Impostor calculates its average reputation amongst the Crewmates, using their shared beliefs. After that, it checks every Crewmate's beliefs of them. If they are above average, the Impostor increases their belief concerning that Crewmate; If they are below average, the Impostor decreases their belief concerning said Crewmate.
- In a **Voting Session**: The Impostor will simply reduce the belief it has of any Crewmate that votes for them.

3.5 Voting Sessions as a Reactive Component

A voting session occurs when a **dead body** is found. A Crewmate that finds said dead body will stop pursuing the task it was going for, and immediately calls a *voting session*. This sudden change of plans mimics the normal functioning of the game.

Before the actual voting, a *deliberation session* takes place, as explained before.

After this, the voting itself works the same for both Crewmates and the Impostor: they vote in the agent they trust the least, according to their saved beliefs. One detail is that they may choose **not to vote**, in case there is more than one agent with the lowest belief - they'll never choose randomly as they tipically do not want to create *unnecessary enemies*.

Then, if there is a *majority* of votes, an agent is ejected from the spaceship.

In a voting session, we take some of the tasks of the Crewmates that had been killed in that round and *re-distribute* them among the alive agents. This is done to keep the game rolling, and corresponds to a similar behaviour of the Among Us game.

4 EXECUTION MODES

We defined four different execution modes to highlight four different *social norms*. Each execution mode is increasingly more sophisticated in the ways agents both plan and react. We will use these execution modes to evaluate our work, in section 5.

4.1 Dummy

A very basic implementation, with very little interaction between agents. This will be our baseline model.

- **Impostor**: follows the Crewmate closest to them and attempts to kill them, using the *Closest Crewmate* heuristic.
- **Crewmates**: only update their beliefs when they **see** the Impostor killing another Crewmate.

It is expected that the Crewmates will mostly choose not to vote, as they don't update the beliefs in nearly any situation.

4.2 Mode #1

A slightly more complex implementation, where the Impostor focuses on killing Crewmates that don't trust them and the Crewmates have some cooperation.

- **Impostor**: decides what Crewmate to kill based on the *Closest Crewmate* and *Least Trusted* heuristics.
- **Crewmates**: the Crewmates now also use the *Update Deliberation* and *Update Vote* heuristics, to share their beliefs among eachother.

4.3 Mode #2

The Impostor will now also try to fool other agents, and Crewmates will now suspect agents that they've last seen.

- **Impostor**: will now complete **fake tasks** to try and fool Crewmates into trusting them.
- **Crewmates**: Crewmates will now use the *Last Seen Agents* heuristic, which will decrease the belief of the last agent(s) they've seen once they find a dead body.

4.4 Mode #3

The Impostor now can find out more vulnerable targets and Crewmates will now pay attention to who's been near them.

- **Impostor**: can now use the most *Isolated Heuristic* to pick a target. This heuristic finds out how isolated each Crewmate is.
- **Crewmates**: Crewmates now keep a record of how many steps each agent has taken near them, and trust those who've stayed by their side the longest, using the *Update Steps Alongside* function.

5 EMPIRICAL EVALUATION

we will conduct three different yet complimentary experiment approaches to showcase our results. All of our experiments will run with 7 agents: 6 Crewmates and 1 impostor.

5.1 Comparing Execution Modes

we will use **four metrics** to compare simulations with different social norms:

- **Crewmate's Success**, which will be a *Boolean value*.
- **Number of False Accusations**, which is the number of accusations in the voting meeting that failed to find the impostor.
- **Number of Voting Sessions**, which is the number of voting rounds that take place before the game ends.
- **Number of Iterations** until the game finishes. This is a way to evaluate the game's duration.

With this set of metrics we hope to be able to distinguish and evaluate different social norms and how they can stimulate cooperation among the Crewmates.

5.1.1 Methodology. We ran each execution mode 100 times to extract the aforementioned evaluation metrics. Having the 100 experimental results for *each* execution mode, we will:

- Present the distribution of Crewmates and Impostor victories, as a percentage.
- The average number of false accusations, across all 100 runs.
- The average number of voting sessions, across all 100 runs.

Our goal is to evaluate each of the metrics for all execution modes.

5.1.2 Crewmate's Success. Fig. 4 shows how different execution modes affect the success of both kinds of agents.

The Dummy model serves as a baseline. Here, the agents can't be considered remotely intelligent.

Mode #1 is the one with the biggest discrepancy in results. This is interesting but should be taken with a grain of salt: Although the Crewmates win much more often, this is still a semi-intelligent model in which the Impostor simply cannot compete with the Crewmate's collective deliberation. This is still exciting as it gives us the hint that, as suspected, promoting trust among the Crewmates has a good effect in their overall success.

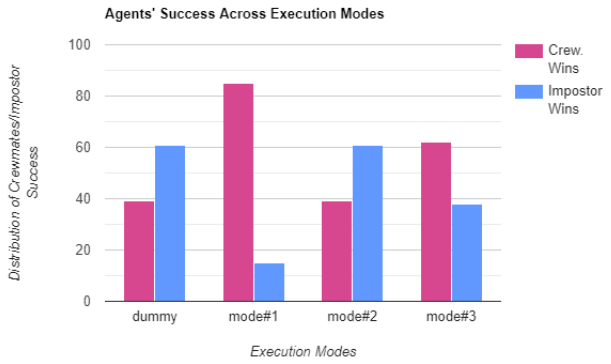

Fig. 4. The distribution of crewmate's success across the different execution modes

In mode #2, we can see that the Crewmates' win rate is lower than the Impostor's. The main reason for this is that the Impostor now can influence the Crewmates both from completing fake tasks and the Crewmate's last seen agents, while the Crewmates do not have enough intelligence in order to cooperate and find the impostor.

Finally, mode #3 is the most interesting to analyse as it is the mode where the Impostor's intelligence in targeting Crewmates and the Crewmates' ability to collectively share opinions are balanced. As expected, the Crewmates are at a clear dominance of the game, meaning that they can successfully assess who the most trustworthy agents are, and thus single out the Impostor.

5.1.3 Number of False Accusations. When analyzing the number of false accusations, one result is clear. The dummy version has no false accusations, since agents will only lower their belief of another agent if they see them killing someone. In this version, a Crewmate never risks voting an agent out unless they are **certain** that it is the Impostor.

In mode #1, false accusations are still lower than in other modes, since agents will only start suspecting someone of being an Impostor, and therefore vote for them, if they either see them killing someone, or if their belief is influenced by other agents. And, once one agent finds the impostor, by sharing their belief with the other agents, the impostor will end up being voted out.

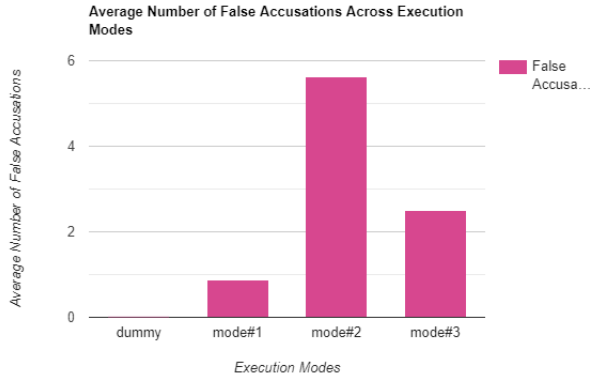

Fig. 5. The average number of false accusations across the different execution modes

In mode #2, false accusations are extremely high compared to other modes, since now the Impostor tries to emulate Crewmate behaviour. That combined with the fact that now Crewmates are more doubtful in general of other agents, by using the *Last Seen Agent* reasoning, makes it so that many voting sessions end up with the ejection of a fellow Crewmate.

In mode #3, false accusations get lower again, although they don't get as low as they did in mode #1. Even though the Impostor makes more calculated kills, with the *Most Isolated* heuristic, Crewmates now are extremely careful, updating their beliefs of someone that stays near them for a long time without killing them, helping them find out who the impostor is by trusting other Crewmates.

5.1.4 Number of Voting Sessions. In Fig. 6, we show how the number of voting sessions varies across the four execution modes.

The Dummy mode is the one with the highest average number of voting sessions. This is because, as stated before, the Crewmates don't really update their beliefs except when they find the Impostor

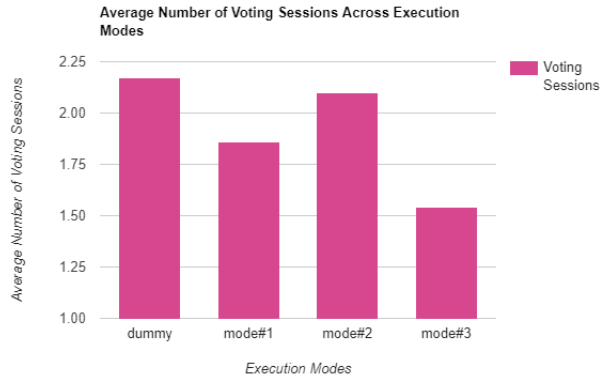

Fig. 6. The average number of voting sessions across the different execution modes

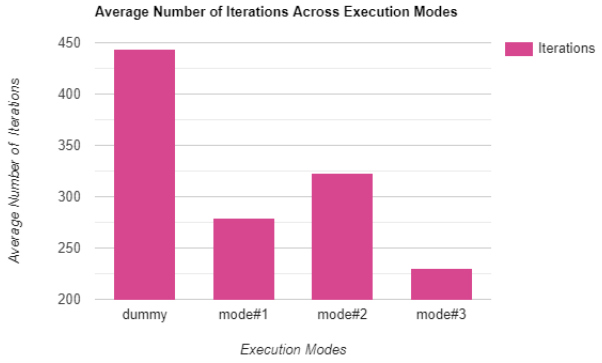

Fig. 7. The average number of iterations across the different execution modes

killing some other crewmate. This means that the game prolongs itself while the Crewmates keep finding dead bodies.

In mode #1, the relatively lower average number of voting sessions is explained by the addition of the belief sharing mechanism in the Deliberation phase of a voting session. Crewmates become more expert at sharing impressions and thus require way less voting sessions to find the Impostor.

In mode #2, the fact that the Impostor starts faking tasks leads to an increased difficulty in the Crewmates finding them, thus increasing the number of voting sessions required.

Finally, in mode #3, the average number of voting sessions is at its lowest, showing that adding the Crewmate deliberative behaviour of trusting more an agent that walks alongside them the longest has its perks in how fast they can find the Impostor.

5.1.5 Number of Iterations. In Fig. 7 we show the variation of the number of iterations according to the execution mode.

Starting by the dummy version, we can easily understand why it has such high values. It happens because the agents in this version do not update their beliefs and for that reason, the game will take longer to end since the voting sessions will be *useless* in this version.

In mode #1 there is large decrease in the number of iterations since we introduce both the *closest* and *leastTrusted* heuristics to the Impostor and the Crewmate's beliefs update in the voting session, before and after the voting. For that reason, agents tend to vote people out in the voting sessions, which means that the game will end faster, i.e. in less iterations.

In mode #2, the number of iterations increases a little bit in comparison with mode#1. This happens since this mode allows the Impostor to do fake tasks, which increases the time that Crewmates need to find him.

Finally, in mode #3, the number of iterations reaches its lowest value. Crewmates now update beliefs in every iteration and the Impostor uses the *mostIsolated* heuristic, allowing both to, respectively, find the Impostor in a faster and better way, and to select the best Crewmate to kill. For this reason, the game will be faster (since agents learn faster) and the number of iterations will be lower, as observed.

5.2 Varying Impostor's Heuristics Importance

In this section, we will evaluate what is the impact of changing the **weights** (or importance) of the Impostor's heuristics, explained in section 3.3. Our goal is to find out if any heuristic is better than the others.

We will depart from the Execution Mode #3 and define four experiments:

- The *baseline* version, corresponding to the version where all the three heuristics weigh the same.
- The *least-trusted* version, where we triple the importance of the least trusted crewmate heuristic.
- The *most-isolated* version, where we triple the importance of the most isolated crewmate heuristic.
- The *closest* version, where we triple the importance of the closest crewmate heuristic.

We ran each experiment 100 times and will present the resulting distribution in Crewmates/Impostor wins.

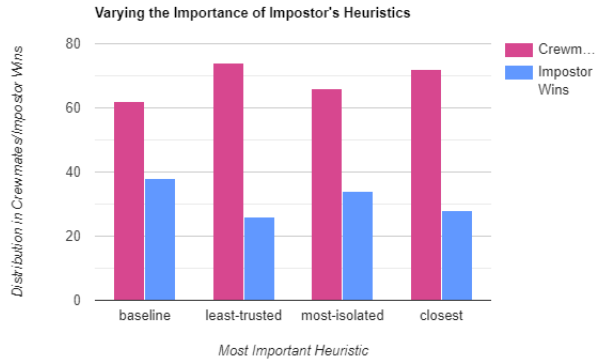

Fig. 8. Varying the Impostor's heuristics importance

We can see in Fig. 8 that *no heuristic* is more impacting in the Impostor's success. The best chance the Impostor has to win the

game is to run the *baseline* version, using all heuristics in the same proportion when deciding a new target.

5.3 Varying Crewmates' Trust Factor

In this section, we will evaluate the impact of changing how the Crewmates increase their beliefs when walking alongside other agents in the spaceship. This kind of update is described in Section 3.4.1.

We will depart from the Execution Mode #3 again and define three experiments:

- The *baseline* model, where the Crewmates increase the beliefs with a factor of 0.01 when walking alongside other agents.
- The *middle* model, where the Crewmates increase the beliefs with a factor of 0.02 when walking alongside other agents.
- The *trust* model, where the Crewmates increase the beliefs with a factor of 0.03 when walking alongside other agents.

We ran each experiment 100 times and will present the resulting distribution in Crewmates/Impostor wins.

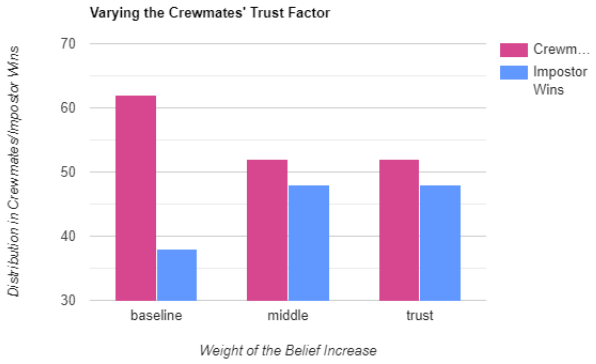

Fig. 9. Varying the Crewmates' trust factor

We can see in Fig. 9, that although the Crewmates always have a better winrate than the Impostor, the baseline model is much better for the Crewmates. Crewmates, in general, have a better chance of winning when they don't put that much trust on other Agents. The best chance the Crewmate has to win is to run the *baseline* version, increasing the beliefs by 0.01 when walking alongside other agents.

It is also interesting to see that there is no difference in the *middle* and *trust* models. This might indicate the existence of a cap in how much we can increase the trust factor before it starts harming the Crewmates' success.

6 CONCLUSION

We were successful in implementing distinct social norms and assessing that indeed they change the probability of Crewmates' success or defeat. We found out that the Crewmates can profit out of trusting one another when walking around the spaceship, as well as building a collective notion of *reputations* in voting sessions.

Furthermore, we also discovered that several heuristics contribute to the success of the Impostor, seeming that the best results are achieved when the Impostor gives equal priority or importance to all three heuristics.

Some ideas were left unexplored and could have been further developed to make both agents behaviour even more sophisticated. Even though most of these ideas were not implemented due to time restraints, we next disclose some of then, in what can constitute some future work:

- Once a Crewmate finds the Impostor *in action*, i.e., killing another agent, they should try to stay away from the Impostor in their future routes.
- When using the *max steps alongside* reasoning, the Crewmates should actively plan their routes to each uncompleted task in a way that allows them to walk alongside other agents they trust. That is, the Crewmates should actively make themselves more trustworthy in order to fully take advantage of the other agents' reasoning.
- On a more complex level, we wished to add a whole new way of interacting with the game on the Impostor end. While on their mission to kill Crewmates, the Impostor could also **sabotage** certain aspects of the spaceship. For example, the Impostor could kill the lights (reducing the range of sight for Crewmates), force Crewmates to complete specific tasks at the same time on opposite sides of the ship (to force them to be separated) or even close entrances to certain areas to isolate certain Crewmates.

All in all, we consider this work was well accomplished and executed, and we hope to have advanced knowledge in the application of Game Theory and social norms in a widely-known game.

REFERENCES

[1] Marcus Bromander. [n.d.]. *Among Us.* https://innersloth.com/gameAmongUs.php