Analysing Impostor Strategies in Among Us Using Agent-Based Modelling

B. J. Hoogland, P. L. van der Jagt, I. M. Reitsma, M. J. Sprinkhuizen, R. R. W. van Woerkom

Abstract
This research investigates the optimal player strategy for the impostor character in the popular multiplayer game Among Us. This is done by applying Agent-Based Modelling, which is scarcely used in gameplay or player modelling. Four strategies for the impostor are analysed. These strategies are a combination of an active or passive tactic, and an aggressive or careful behaviour. The analysis of these strategies is done for both single games, as well as iterated games with the same player set in order to find emerging social effects. The strategy that resulted in the highest impostor win-rate in the single games was the strategy where the impostor is aggressive and active. For iterated games the win-rate of this strategy declined significantly suggesting a long-term negative effect for a less social strategy. Another interesting result is that the the win-rate of the strategy where the impostor is careful and passive significantly increased for iterated games. However, this increase is not enough to make a significant difference to the other strategies, so no superior strategy for iterated games was found.

Keywords
Among Us — Agent Based Model — Mesa — Player Modelling — Iterated games

Contents

1. Introduction

One of the most popular multiplayer games at the moment is Among Us, hitting 3 million concurrent players in September 2020 [1]. Among Us was released in 2018 on multiple platforms and gained significant popularity around the world due to the COVID-19 lockdowns, leading up to 60 million players a day at the end of September 2020. Even well known American politicians, such as Alexandria Ocasio-Cortez and Ilhan Omar, participated in the game [2]. In Among Us, players need to communicate with one another in order to succeed, which gives the game a strong social aspect.

Player modelling is a research-field focusing on detection, modelling, prediction and expression of player characteristics. It often uses computer models to analyse different player strategies and experiences. Understanding which strategies work based on gameplay can have multiple applications, such as creating believable computer agents who play the game and determining how to alter the games mechanics. Altering the mechanics can be done to either fit better to these strategies, or to fit worse to the strategies, forcing players to adopt different strategies to still be able to win the game, changing the overall gaming experience [3].

To gain these insights for Among Us, the game needs to be modelled. A modelling technique that so far has not been used often is Agent-Based Modelling. Very few, if any, literature is available using Agent-Based Models (ABMs) in gameplay or player modelling. Most of the ABMs related to gaming are targeted towards two player games, and almost not to multiplayer games. Most models with a large number of agents are not related to gaming, such as the well-known Schelling's segregation model [4]. As mentioned by Szilagyi [5], most of the multiplayer ABM simulations are narrowly defined, with

limited difference in variables. In addition, the authors themselves assume the agents have no goals in their own research, and focus on different cooperation or non-cooperation strategies. Therefore, this research focuses on implementing a more complicated game, Among Us, using Agent-Based Modelling, to determine optimal player strategies of the impostor both for single games as well as multiple games with the same player set.

1.1 Gameplay

The default environment of the game is on a space-ship. At the start of the game each player is assigned a role. The majority of the players will be 'crewmates' and the remaining players will be 'impostors'. The goal of the crewmates is to find out who the impostors are and to complete tasks (in the form of minigames) on the map before the time runs out. The goal of the impostors is to kill all the crewmates before they are able to complete their tasks. After an impostor kills a crewmate, he is assigned a kill cooldown time in which he temporarily cannot kill other crewmates. Players can start meetings to discuss who the impostor is by either calling an emergency meeting in the Cafeteria, or by reporting a dead body that is left after the impostor kills a crewmate. In these meetings, players can vote on who they think is the impostor, and the player with the most votes is eliminated from the game. During the meetings it can also be decided to skip the voting, such that no player is eliminated.

The impostors win if the number of impostors is equal to the number of crewmates. The crewmates win if they eliminate all impostors or when they complete all their tasks. The impostor can also 'vent', which is travelling through the air vents in the map, creating short-cuts to different areas in the map. However, he needs to be careful that this venting is not observed by crewmates, as it will give away his role. All players have limited vision, only allowing them to see other players that are not blocked by walls or certain objects.

For the implementation of the ABM, Mesa is used [6]. Mesa is a framework for ABM in Python, which makes it possible to visualise the model. In addition, Mesa has a build-in grid with functions that make it easier to retrieve information about the grid.

2. The Model

This report makes use of the ODD+D (Overview, Design concepts and Details + Decision making) protocol for describing the ABM [7]. In addition, the article of Grimm et al. [8] has been taken into account to alleviate the shortcomings of the ODD format, especially regarding the description of the rationale behind modelling decisions.

2.1 Overview

2.1.1 Purpose

The purpose of this report is three-fold. First, ABM has been used scarcely as a method for analysing computer-games. When this report was written, there was no record of other papers analysing Among Us using ABM. Therefore, this report explores the novelty of ABM modelling in video games. Secondly, another aim of this report is to determine the strategy of the impostor that leads to the most wins of games. The analysis of the game can be used by both the creators of the game, to alter the gameplay so different strategies work better or adjust the game mechanics to change the ratio of games won by either party, and by the game players, to determine which strategy to apply when they are the impostor or change their strategy to make catching the impostor more likely when they are a crewmate. Finally, the different impostor tactics result in different social relations between the players of the game, comparing these different tactics for single games and iterated games can elucidate the importance of taking into account social factors while determining the best strategy.

2.1.2 Entities, state variables and scales

The main agent entities are the crewmates and the impostor, which can move around the grid. The crewmates and impostors are characterised by the strategy of the impostor, the tasks the crewmates need to execute, the kill cooldown time, vision range, and their respective trustworthiness and suspiciousness. These last two variables are defined in the 'sus' and 'trust' matrix, containing scores of each agent and how much they trust or suspect each of the others. At each step, the sus matrix is updated, following equation 1, and after each voting round the trust list, according to equation 3. The trust list is saved over multiple games (if they are iterative), while the sus matrix is reset every game. Specific for the impostor are the kill cooldown time which prevents the impostor from being able to kill in quick succession, and the strategy. The strategy of the impostor consists of two parts, the tactic and the behaviour. The behaviour can be either aggressive or careful, and the tactic can be active or passive. These strategies are further explained in the design concepts 2.2.1. Specific for the crewmates are their tasks. Each crewmate is assigned four random short tasks at the start of a game which he will need to complete by going to the location of the tasks and spending between 5 and 7 in-game seconds here. Most tasks require the crewmate walking to one location, but there are also some tasks that require the player to visit two or three rooms subsequently. The game also has several 'common tasks'; tasks that need to be executed either by all crewmates, or by none of them. There are two common tasks, one that only requires the crewmates to travel to one location, the other requires three locations in a certain order. These locations are different for different crewmates, but the task remains the same.

Both aforementioned agent types also have a vision range, the range within which they can see each other. A dictionary containing the vision range of the crewmates and impostor for each cell of the grid is created beforehand, and can be loaded in. This way, the vision of a crewmate or impostor on its current position can easily be accessed.

An overview of all state variables and their types and scales are shown in table 1.

Regarding the entities, there are four additional agent types, Dead Crewmate, Wall, Obstruction, and Vent. Whenever a crewmate gets killed, he will be replaced by a 'Dead Crewmate', which cannot move around the grid and will stay on the position at which he was killed. Other players can find this body when it is in their vision, and report it in order to start a meeting. After the meeting, all dead bodies are removed from the model.

Agents of type Wall and type Obstruction are used to recreate the map on a rectangular grid. The map used is the most popular map in the game, a spaceship called The Skeld. The types are similar, since the agents cannot walk over either of them, however, the agents can see through an Obstruction, but not through a Wall. Certain objects in the original map are modelled as Wall, to accurately model impostors and crewmates being unable to see through these objects. The impostor can use a Vent agent as a shortcut to linked vents.

The map is loaded in by using an image of the map of 138x242 pixels with distinct colours for the agent types. Each timestep in the simulation corresponds to 0.07 seconds in the game on the standard settings.

2.1.3 Process overview and scheduling

In the model, first the Crewmates are updated, and then the Impostor. At the end of each step, the 'sus matrix' is increased with 0.0005 for each Agent. The steps of the crewmates are given in algorithm 1. Here, 'findpath' finds the path towards the task in the tasklist using the A* pathfinding algorithm [9]. One step of the impostor is represented by algorithm 2. Here, the 'findpath' function picks a random location of any task, and moves there using A* pathfinding [9]. The 'euclidean' function determines the distance between the impostor and a crewmate, and if the distance is sufficiently short, the crewmate is within the kill range of the impostor.

Algorithm 1: Pseudocode for a step of a crewmate agent

```
if path == None then
    if currpos in goallist then
        complete task, possibly add next part of task;
        if all tasks are done then
            Go to main room, run around;
    find path;
else if path ≠ None then
    If in task, stand still else, move towards goal;
Detect neighbouring agents;
for agents in vision do
    Change 'sus' matrix;
```

Certain actions of the crewmates and impostor affect their suspiciousness during the game. These actions and their respective effects are described with equation 1 and the values of 2. These equations describe how suspicious player i believes player j is, and is updated at every time step in the

Algorithm 2: Pseudocode for a step of the impostor agent

```
Decrease cooldown and kill timers if they are >0
if path == None then
    find path depending on tactic;
Detect crewmates;
for crewmate in detected crewmates do
    if kill cooldown == 0 and euclidean ≤ 6 then
        if behaviour == agressive then
            Kill crewmate, set cooldown and
             justkilled timers;
        else if behaviour == careful then
            set waiting countdown, set behaviour to
             waiting;
        else if behaviour == waiting then
            if waiting countdown > 0 then
                decrease waiting countdown
            else
                if only 1 crewmate in vision then
                    kill crewmate, set cooldown and
                     justkilled timers, set behaviour
                     to careful;
                else
                    set cooldown timer, set behaviour
                     to careful;
if path ≠ None then
    move towards goal;
```

Variable	Explanation	Type	Scale
N_c	number of Crewmates	Integer	> 0
N_i	number of Impostors	Integer	$0 < N_i < N_c$
T	number of tasks every Crewmate gets	Integer	> 0
S_m	suspicion level between players	Matrix of floats	$[0,1]$
T_v	trust in a players opinion	Vector of floats	$[0,1]$
V_r	vision range of agents	Integer	> 0
S_p	all possible starting positions	Set of coordinates	(X,Y)
T_T	the range of how long a task can take	Integer	$[A,B], A > 0, B > A$
K_c	kill cooldown of the impostor	Integer	> 0
I_b	impostor aggressive behaviour	Boolean	
I_t	impostor active tactic	Boolean	
I_v	impostor vents active	Boolean	
σ_1	suspicion score of witnessing kill	Float	$[0,\infty]$
σ_2	suspicion score of witnessing vent	Float	$[0,\infty]$
σ_3	suspicion score of witnessing task	Float	$[-1,0]$
σ_4	suspicion score of walking together	Float	$[-1,0]$
σ_5	suspicion score change with time	Float	$[0,1]$
γ_1	trust score change when voted correctly	Float	$[0,1]$
γ_2	trust score change when voted wrongly	Float	$[-1,0]$

Table 1. Overview of all variables in the model with their explanation, type and scale

model. They are reset to their default values every new game.

$$\Delta Sm_{ij} = \sum_{n=1}^{5} \alpha_n \sigma_n \tag{1}$$

$$\alpha_1 = \begin{cases} 1 & \text{if } j \text{ killed someone in vision of } i \\ 0 & \text{if else} \end{cases}$$
$$\alpha_2 = \begin{cases} 1 & \text{if } j \text{ took a vent in vision } i \\ 0 & \text{if else} \end{cases}$$
$$\alpha_3 = \begin{cases} 1 & \text{if } j \text{ killed performed a task in vision } i \\ 0 & \text{if else} \end{cases} \tag{2}$$
$$\alpha_4 = \begin{cases} 1 & \text{if } j \text{ is walking in vision of } i \\ 0 & \text{if else} \end{cases}$$
$$\alpha_5 = 1$$

And the equations representing the changes in the trustworthiness of one player according to an other are:

$$\Delta Tv_i = \lambda\gamma_1 + (1-\lambda)\gamma_2 \tag{3}$$

$$\lambda = \begin{cases} 1 & \text{if player } i \text{ voted right} \\ 0 & \text{if else} \end{cases}$$

2.2 Design concepts

2.2.1 Basic principles

This model tests the strategies that players can have when they are assigned the role of impostor. Their strategy can influence their odds to win the game. A more active role of quickly trying to kill all crewmates can result in getting caught more easily, increasing their 'sus' score and thereby their chance of being voted out, but being too passive might give the crewmates enough time to finish their tasks. These different strategies are modelled by giving an impostor an 'active' or a 'passive' strategy, and 'aggressive' or 'careful' behaviour. In the aggressive strategy, the impostor continuously walks from task to task, 'seeking out' crewmates by showing up near their tasks. With the passive strategy, the impostor walks to one task, and stays there until a crewmate shows up. When a crewmate appears in the vision of the impostor, the impostor kills the crewmate and moves to a different task location, to avoid suspicion. The difference between the 'aggressive' and 'careful' behaviour is twofold: with the aggressive behaviour, the impostor kills a crewmate immediately after the initial killing cooldown ends, if one or more are in sight, while the 'careful' impostor will wait 5 more steps after kill cooldown ends, and then only kill if there is exactly one crewmate in sight, and not more, to avoid being spotted killing. The sus value between agents within their vision range decreases at each step, based on the assumptions that if the other agent is the impostor the first agent would be killed, and that the other agent can not be killing another crewmate at that moment, thus making it less believable they are the impostor. The trust value between agents is updated after a voting procedure based on if a player voted correctly on the impostor. The parameters of the model are based on data collected by the game developers, as mentioned in 2.3.2.

2.2.2 Emergence

Adapting the strategy mostly concerns the behaviour of the impostor. In contrast, the general behaviour of the crewmates is more constant throughout the model, always aiming to complete all the tasks they are assigned as quickly as possible. However, when the impostor is aggressive for example, the crewmates might have a higher chance of observing a murder or finding a body, triggering more meetings. This in turn will lead to games lasting shorter, and less tasks being completed.

2.2.3 Adaptation

When the impostor strategy is careful, it will depend on whether there is another crewmate in the vicinity whether he plans to kill a crewmate.

2.2.4 Objectives

The objective for all players is to win the game. In order to achieve this, other intermediate objectives are given, such as completing tasks and killing crewmates. The game runs until either the impostor or the crewmates reach this goal. If the number of crewmates is equal to the number of impostors, the impostor wins. If the crewmates complete all their tasks or vote out the impostor, all the crewmates win, even the ones who were voted out or died.

2.2.5 Individual decision making

The crewmates and impostor are all both the subjects and objects of decision-making. They each vote the moment a meeting is called after the body of a crewmate has been found. Each agent votes for the agent they consider the most suspicious. The product is taken of the 'trust' and 'sus' scores for each agent, and these scores are added together and normalised using a tangent hyperbolic function. Whichever agent ends up having the highest score gets voted out, resembling a voting process in which the agents discuss their respective trusted and suspected agents, and deciding on the agent with the most suspicion and least trustworthiness. By definition, an agent has zero suspicion of himself and of dead agents. An example of this decision making process is given in the supplementary data S1.

2.2.6 Learning

During the game, the 'trust' and 'sus' scores are updated, which changes the voting behaviour of the players. The trust score is updated after each vote. It is based on how often an agent suspects the wrong agent of being the impostor, or more precisely, whether the agent had a high 'sus' score regarding the impostor and a low score regarding other crewmates during the last vote or in the previous game(s). These scores change the decision whom to vote out. The overall decision making process does not change over the simulations.

2.2.7 Individual sensing

The crewmates are able to sense nearby other crewmates and the impostor, and can sense dead bodies surrounding them. They can also find and complete the tasks nearby. It is possible for the crewmates to incorrectly assume that a nearby agent is completing a task, which increases the score of the trust list for that crewmate, when it is in fact the impostor who is standing on the task, thus leading to an erroneous assumption that the agent is innocent. The crewmates can also sense whether the impostor has just killed within their vision.

The vision of the agents is implemented by using Bresenham's line algorithm for ray tracing [10]. The maximal vision of agents is within a radius of 20 grid cells, if not blocked by a wall or opaque object. In this vision field, the agent can sense the other agents and perceive their actions. The mechanisms by which agents obtain information are modelled explicitly. During each step of an agent, the algorithm iterates over all positions visible to them. These visible positions are calculated before the model runs and saved in a dictionary to reduce the computational cost.

For the impostor, this space is only checked for potential crewmates to kill. For crewmates, the space is checked for dead bodies to report, and for other agents that are either a fellow crewmate or the impostor. Observing or not observing other players will then decrease or increase their trust scores. The crewmates can also perceive if another player is in the vicinity of a dead body, if a player uses a vent and if a player is on a location of a task. The latter decreases the 'sus' score of the observed player, while the first two instantly raise this score to the highest value.

2.2.8 Interaction

The most important and direct interaction among agents is when the impostor kills a crewmate. This removes the crewmate agent, changing it into a dead agent. This dead agent is not able to move anymore, so it will stay on its place. Interaction also takes place between a task and a crewmate, when the crewmate executes a task and this task is removed from the 'to-do list' of the crewmate. However, these interactions are mainly indirect, since the Task agent remains present in the map, to be able to be executable by another agent. Each agent has a list with goals, or tasks to be executed, from which the task gets removed once arriving at its location. The agent also remains at this location for a random amount of steps between 70 and 100. A form of interaction exists during voting, where the agents compare their trust and sus scores, modelled indirectly by multiplying and adding up the scores, representing a discussion during the actual game.

2.2.9 Collectives

All crewmates belong to the crewmate aggregation. They work together in trying to win the game, and win together, meaning that a dead or voted out crewmate can win if the remaining crewmates reach their goal.

2.2.10 Heterogeneity

The crewmate agents exhibit almost the same state variables and processes. The difference between these agents is which tasks they need to execute and how many of them they have completed. Also the 'sus' and 'trust' scores assigned to other players are different between the crewmates. The decision

making of all crewmates is according to the same model, only the decision on who to vote for will differ between agents since their 'sus' and 'trust' scores are different. The dead crewmate agents are former crewmates, all dead crewmates are the same except for their position. The game has only one impostor agent, which differs from the crewmates because he can use the vents and can kill crewmates. The impostor and crewmates also differ in their pathfinding goals. For the crewmates, these goals are the tasks they need to complete. The path goal of the impostor is a random task, and the vents are considered in the pathfinding.

2.2.11 Stochasticity

Each crewmate is assigned a list of random chosen tasks. Also, if the crewmates execute the 'fix wires' common task, the locations in which the wires need fixing are chosen randomly and ascribed to crewmates. The impostor walks randomly to different tasks, to either await a crewmate, or to randomly start walking towards another tasks. During the initialisation of the model, the impostor can be randomly chosen from the group of crewmates. However, in the model the cycle of agents becoming an impostor is deterministic in order to compare multiple runs with one another.

2.2.12 Observation

Five of the state variables mentioned in table 1 are analysed using the OFAT method. These variables are γ_1, γ_2, σ_3, σ_4 and σ_5. These variables are chosen since they represent the social factors, where there is no single logical value. For each variables, 5 values are chosen, and each value is tested on the model for 160 iterations, totalling 800 iterations per variable. These 800 iterations are then repeated two times, once with no iterative games and once with 32 iterative games. There are 4 crewmates and 1 impostor at all times. Each of the four different strategy combinations is assigned to a player, and each round one of the players with a strategy becomes the impostor, iterating over the players so all players become the impostor an equal amount of games. The fifth player never becomes the impostor. After each simulation the trust list is saved, as well as a matrix that stores how often each player wins as the impostor or as a crewmate. In addition, how many crewmates finished their tasks is saved, as well as how many crewmates died, the amount of iterations that the game lasted, the amount of tasks that are finished and whether the impostor or the crewmates won the game is stored. An analysis of the results can be seen in section 3.

2.3 Details

2.3.1 Implementation details

The model has been implemented using Python (version 3.8.5) and uses the Mesa package (version 0.8.8.1). The main model has been implemented using Jupyter Notebook. The model, along with the input files, is accessible in GitHub [11].

2.3.2 Initialization

The players are all always spawned in the same area of the map: the cafetaria, in both our model as well as in the actual game. All walls, obstructions, vents and tasks are loaded into the map, recreating the actual map.

The model takes the amount of crewmates, impostors, the two values between which a random value is chosen to determine the amount of time a crewmate spends doing a task, the kill cooldown time, the number of tasks for the crewmates, the impostor tactic and behavior, whether the impostor can enter vents, how many steps it remains visible that the impostor just killed, and with how much the trust and sus scores change during the different events as input. These can be varied.

During the model testing, we chose to have four crewmates and one impostor, with the time chosen for the tasks being between 70 and 100 model steps. These values are based on the Among Us data, which has been analysed by a data analyst working for InnerSloth, the game developer [12]. Their analysis shows that 4 or 5 player games are the most common. In games with 5 players or less, it is suggested to have one impostor [13]. The time to spend in task has been chosen based on our own research, completing 20 short tasks in Among Us, each of which took us anywhere between 5 and 11 seconds. We decided to keep the time to complete one task on the short side of the range, so between 5 and 7 seconds.

Each crewmate is assigned four random tasks at the start of the game. The maximum tasks possible in the settings of the game are two common tasks, three long tasks and five short tasks. We decided to implement the two common tasks and four random short tasks, to take half of the maximum amount of short and long tasks combined, and adding the common tasks to make up for the lack of long tasks. The impostor is given a tactic, behaviour and kill cooldown. The tactic and behaviour are varied during testing, to determine the best strategy for the impostor. The kill cooldown is set to 214 timesteps, which corresponds to 15 seconds. This favours the impostor slightly, as mentioned in [13], but is justifiable considering the ratio impostor/crewmates, which is on the low side, making it difficult for the impostor to win. In addition, 15 seconds kill cooldown is the standard setting of the game, thereby making it likely that it is the most used setting, making our gameplay more similar to the actual gameplay.

The changes in trust and sus scores depending on whether they are observed, and what the agents are doing when observed, are set fixed at an increase in suspiciousness, and the decrease in suspiciousness when performing a task or walking together are varied over the simulations. For the exact values, see section 3. Initially, all players trust and suspect one another with a value of 0.5.

2.3.3 Input data

The input for the model are files that contain the locations of the walls, obstructions, vents and tasks. A dictionary containing a list of visible grid cells from each grid cell is loaded in beforehand. The model also stores the trust list, for use in the next game. The A* pathfinding algorithm [9] is implemented in a separate Python file ('A_star_pathfinding.py' on GitHub [11]) and imported at the start of the model.

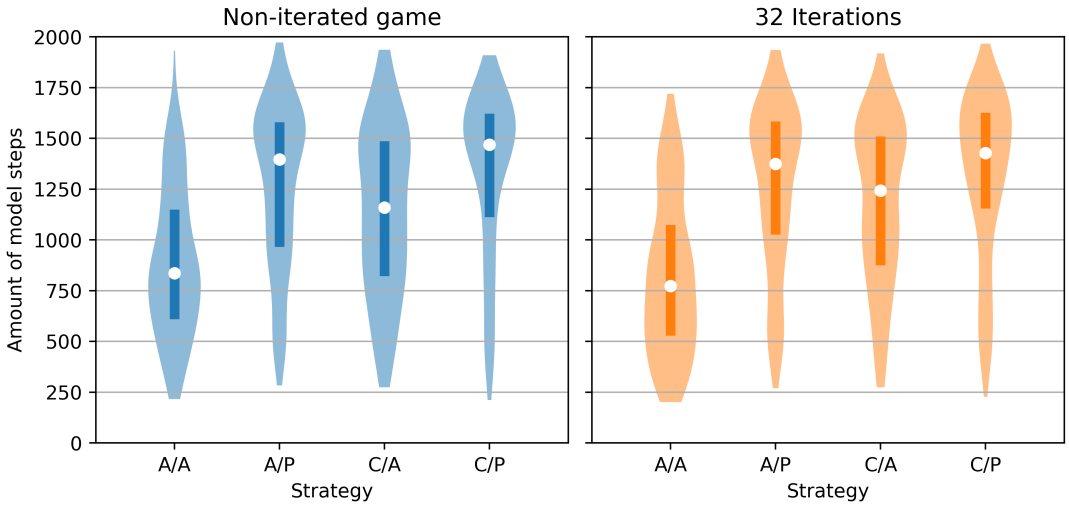


Figure 1. The amount of steps in the model before the game ends. One step equates to 0.07 seconds in-game time. The white circles are the medians, the dark coloured bars show the first and third quartiles and the surrounding shade shows the distribution density of the data.

2.3.4 Submodels

Bresenham's line algorithm is used to determine the vision range on each grid cell [10]. This algorithm was chosen because it can approximate a straight line on a grid. Enough lines were created to fully cover all grid cells within a circle with a radius of 20 grid cells. The centre of this circle is then transformed to a certain position on the grid, from where ray tracing can be done. For full implementation, see 'vision.py' on GitHub [11].

3. Results

3.1 Setup and Settings

Unless otherwise specified, all the variables were set to the standard values in order to achieve the results; $N_c = 4$, $N_i = 1$, $T = 4$, $S_m = 0.5$, $T_v = 0.5$, $V_r = 20$, $S_p = 8$ positions, $T_T = [70, 100]$, $K_c = 214$, $I_b =$ False, $I_t =$ True, $I_v =$ True, $\sigma_1 = \infty$, $\sigma_2 = \infty$, $\sigma_3 = -0.1$, $\sigma_4 = -0.01$, $\sigma_5 = 0.0005$, $\gamma_1 = 0.04$, $\gamma_2 = -0.02$. The standard values arise from the standard game setting values, with the exception of the "social" variables σ and γ, which will be further explored in section 3.5.

3.2 Typical run

In order to visualise a single run of the model, a visualisation was made. This model was run with the impostor having an active tactic and careful behaviour.

Figure 1 shows the amount of steps necessary to complete a single game in a violin plot. It can be seen that an Aggressive/Active strategy results in shorter games, while the Careful/Passive strategy seldom ends early. There is no significant difference for the length of a game between the non-iterated and the iterated game.

3.3 Win-rate per strategy

In order to determine the win-rate for each strategy, the model was run a total of 800 times in batches of 32 runs. Four agents could become the impostor, each with a unique strategy. For the non-iterated games, the trust matrix was reset between each run, whereas for the iterated games it was updated throughout the 32 runs. In Figure 2, the average win-rates for the four different strategies are shown for non-iterated and iterated games. It can be seen that an Aggressive/Active strategy results in a significantly higher win-rate than any other strategy when the games are independent, but also becomes significantly worse applied in an iterated game. Reversely, the Careful/Passive strategy has the lowest win-rate for the independent games, but results in a significantly higher win-rate in the iterated game. This could be an effect of the changing trust values between players.

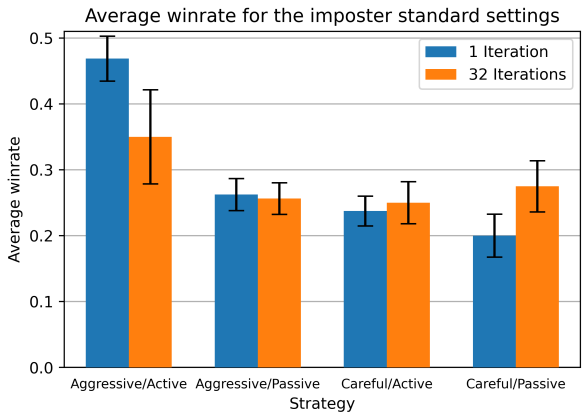


Figure 2. The average win-rate for the four different strategies, with non-iterated (left) and iterated games (right)

3.4 Change in trust in iterated games

In order to verify the hypothesis that the trust values were the underlying cause for the significant shift in win-rates, the mean trust that an agent had was calculated for the duration of the 32 games. As seen in figure 3, the average trust an agent has when his strategy as an impostor is Aggressive/Active becomes lower as the session progresses, whereas the other strategies result in the agent being more trustworthy. It is interesting to note that although in figure 2, the win-rate for the Careful/Passive strategy increased significantly with the iterated games, the average trust score is comparable to that of the remaining strategies. This means that the increase in the win-rate is not directly explainable solely from the trust score.

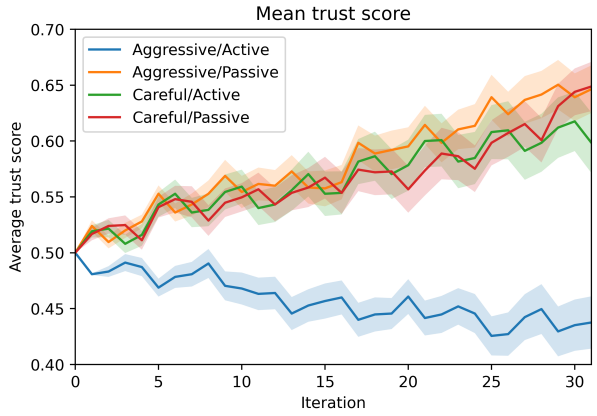
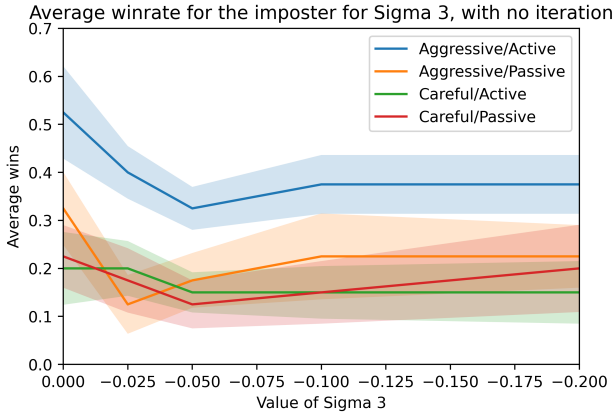


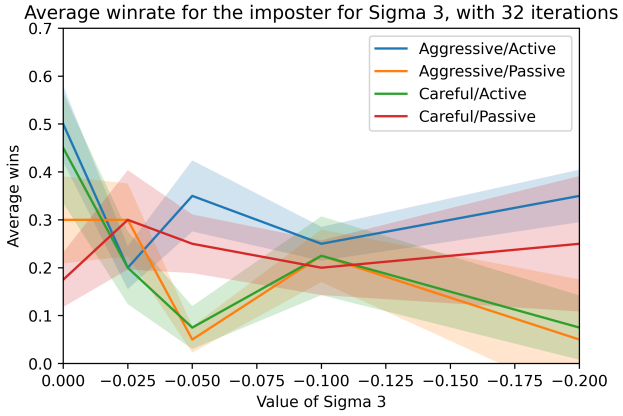
Figure 3. The average trust value an agent received for the four different strategies.)

3.5 Sensitivity of social variables on win-rates

To reiterate, σ and γ indicate respectively the result of certain behaviour on how much a player suspects another player, and how much a player trusts another player. σ_1 and σ_2 are cases in which it is certain that the impostor is found, thus are set to ∞. However, the other values (see table 1) are more subjective. In order to investigate the effect of different social variable values, an OFAT sensitivity analysis was performed. This analysis was done by individually varying parameters σ_3, σ_4, σ_5, γ_1 and γ_2, while keeping the other parameters fixed at their default values. For each of these five parameters, five different values were chosen and used during 160 model runs, 40 model runs per strategy. This was done with both iteration turned on and iteration turned off, resulting in $(5 \times 160 \times 2 =)1600$ model runs per investigated parameter. Figure 4 shows the average impostor wins for different values of σ_3, the 'suspicion score of witnessing task' parameter, for iterated and non-iterated games. Interestingly, the non-iterated variant as seen in Figure 4 (a), suggests that the Aggressive/Active strategy is by far the most effective for the impostor compared to the other strategies. Meanwhile, there does not seem to be a clearly superior strategy for the impostor in the iterated variant shown in Figure 4 (b). Further results from the sensitivity analysis are given in the supplementary data S3., with the trust parameters γ having the most impact when they are set to 0, meaning the trust score does not change. When this is done, the win-rate for the impostor is somewhat higher. It also seems that the win-rate for the impostor with the Aggressive/Active behaviour is higher when σ_4 has a larger negative value.



(a) Effect of a change in σ_3 without iterating the game



(b) Effect of a change in σ_3 with iterating the game

Figure 4. Sensitivity analysis for the σ_3, the reduction in the suspicion score when witnessing a task.

4. Conclusion and discussion

The goal of this research was to determine the optimal impostor strategy for a simplified version of the game AmongUs in a single game and in iterated games with the same players in order to find emerging social implications of (anti-)social strategies using Agent-Based Modelling.

The results showed that the Aggressive/Active strategy resulted in the highest impostor win-rate for independent games. For iterated games the win-rate of this strategy declined significantly, which could be addressed to a change in trust between players during iterated games where the antisocial playing style of the Aggressive/Active strategy resulted in less influence on the group on the long-term. However, it should be

noted that during these games the impostor role was fulfilled in a sequential manner by the players, where the player with Aggressive/Active strategy was always the first impostor during a series of 32 iterated games. This might have resulted towards a bias since the impostor's trust score always declines and that this 'back start' resulted in an enhanced effect on the trust score. Repeating this experiment partly showed indeed that if the Agressive/Active player was not the first impostor, the long term effect on the trust score was more mild. The Careful/Passive strategy had the reverse effect: the impostor win-rate increased for iterated games. This effect could be explained directly by the increasing trust-score of the Careful/Passive strategy. However, no increase in win-rate was observed for Aggressive/Passive and Careful/Active, while their trust score increased almost equally with Careful/Passive and thus it is unsure that the increasing win-rate of Careful/Passive is increased due to the increasing trust-scores. The most effective impostor strategy for non-iterated games is thus the Aggressive/Active strategy, while for iterated games no significant superior strategy was found.

4.1 Suggestions for further research

In our model all the agents are controlled by a computer, interesting for future research is the incorporation of human agents into the model, so that there are both human players and computer agents playing simultaneously. This could be used to determine whether the strategies that are optimal against computer agents work as well against human players. Or, an even more challenging experiment would be creating such a human like agent that it becomes indistinguishable from the human players. However, social behaviour of a computer controlled entity which is indistinguishable from a real human's behaviour could have enormous societal implications and should require the necessary ethical debate, but this is too much outside of the scope of this report to address further [14].

Another suggestion for improving the model is to include more factors in the game. In the real game, players can also use a monitor in the security room on the map to observe other locations than their own vision range. In addition, dead players can still help to complete tasks, while in our model they can not do anything after they are killed, and all their tasks are marked as done. It would be interesting to analyse the effect of these additional factors on the model in order to observe possible emerging effects in a more complicated environment.

Lastly, instead of having just four different possible strategies for the impostor, incorporating more tactics and or behaviours into the model for both the impostor as for the crewmates could lead to overall more realistic behaviour of the agents, up until the point where a players tactic and behaviour is the result of multiple continuous factors instead of the Boolean system now used. Machine learning techniques could even be applied to dynamically alter the behaviour of agents depending on the scenario, in order to optimise their win-rate. This could be done with evolutionary algorithms, to optimise the behaviour of an agent over multiple games, in a similar way as was done for the EvoMan game [15].

References

[1] Innersloth, [Twitter user `@InnerslothDevs`], "We hit 3 million players across all platforms over the weekend! Thank you everyone for enjoying the game!" 29 September, 2020.

[2] A. Fisher, "Among us: the video game that has shot 100 million players into outer space," Nov, 2020. https://www.theguardian.com/games/2020/nov/29/among-us-video-game-100-million-outer-space.

[3] G. N. Yannakakis, P. Spronck, D. Loiacono, and E. André, "Player modeling,".

[4] T. C. Schelling, *Micromotives and macrobehavior*. WW Norton & Company.

[5] M. N. Szilagyi, "Investigation of n-person games by agent-based modeling," *Complex Systems* **21** (2012) no. 3, 201–243.

[6] D. Masad and J. Kazil, "Mesa: an agent-based modeling framework," in *14th PYTHON in Science Conference*, pp. 53–60, Citeseer.

[7] B. Müller, F. Bohn, G. Dreßler, J. Groeneveld, C. Klassert, R. Martin, M. Schlüter, J. Schulze, H. Weise, and N. Schwarz, "Describing human decisions in agent-based models–odd+ d, an extension of the odd protocol," *Environmental Modelling & Software* **48** (2013) 37–48.

[8] V. Grimm, S. F. Railsback, C. E. Vincenot, U. Berger, C. Gallagher, D. L. DeAngelis, B. Edmonds, J. Ge, J. Giske, J. Groeneveld, *et al.*, "The odd protocol for describing agent-based and other simulation models: A second update to improve clarity, replication, and structural realism," *Journal of Artificial Societies and Social Simulation* **23** (2020) no. 2, .

[9] X. Cui and H. Shi, "A*-based pathfinding in modern computer games," *International Journal of Computer Science and Network Security* **11** (2011) no. 1, 125–130.

[10] J. E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems journal* **4** (1965) no. 1, 25–30.

[11] Hoogland, van der Jagt, Sprinkhuizen, van Woerkom, and Reitsma, "Among us." https://github.com/reitsmairis/amongus.

[12] Innersloth, "The data among us," Oct, 2018. https://innersloth.itch.io/among-us/devlog/50755/the-data-among-us.

[13] "Rules and settings: How to create the best among us game - among us wiki guide," Sep, 2020.

https://www.ign.com/wikis/among-us/Rules_and_Settings:_How_to_Create_the_Best_Among_Us_Game#How_many_Impostors_should_I_have.3F.

[14] T. Hagendorff, "The Ethics of AI Ethics: An Evaluation of Guidelines," *Minds and Machines* **30** (2020) no. 1, 99–120. https://doi.org/10.1007/s11023-020-09517-8.

[15] K. d. S. M. de Araújo and F. O. de França, "An electronic-game framework for evaluating coevolutionary algorithms," *arXiv preprint arXiv:1604.00644* (2016) .

Supplementary data

S1. Example calculation of sus and trust matrices

In order to elaborate on the working of the social interactions within this model here we provide example calculations of the sus and trust matrices and the result of a voting procedure. In this game there are four players, respectively player 1, player 2, player 3 and player 4. Player 3 is the impostor and player 4 has been killed already. After n steps we have the following sus matrix (S_m): $\begin{bmatrix} 0 & 0.4 & 0.2 & 0 \\ 0.25 & 0 & 0.55 & 0 \\ 0.7 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, and the following trust list (T_m): $\begin{bmatrix} 0.3 & 0.4 & 0.2 & 0.5 \end{bmatrix}$, both after normalisation.

Sus matrix change after a step

The first player to move is player 1, who sees player 2 doing a task and cannot see player three. Thus the change in sus-score towards player 2 will be reduced with $\sigma_3(=-0.1)$ and the sus-score towards player 2 and 3 will be increased with $\sigma_5(=0.0005)$. The resulting matrix after player 1 moved is as follows:

$$\begin{bmatrix} 0 & 0.3005 & 0.2005 & 0 \\ 0.25 & 0 & 0.55 & 0 \\ 0.7 & 0.25 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Of course, player 2 and 3 still have to move during this step and adjust sus scores from their perspective, player 2 can see player 1 walking and can't see player 3, and player 3 can neither see player 1 or 2. The final sus matrix at step $n+1$ will be:

$$\begin{bmatrix} 0 & 0.3005 & 0.2005 & 0 \\ 0.24 & 0 & 0.5505 & 0 \\ 0.7005 & 0.2505 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Voting procedure

When the body of a death player is found, in this case player 4, a voting procedure starts. Here the living players communicate to each other whom they suspect, how much a players suspections contribute towards the voting outcome depends on how much this players suspections are trusted. This is calculated as $S_m * T_m =$

$$\begin{bmatrix} 0.3 & 0.4 & 0.2 & 0.5 \end{bmatrix} \begin{bmatrix} 0 & 0.3005 & 0.2005 & 0 \\ 0.24 & 0 & 0.5505 & 0 \\ 0.7005 & 0.2505 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$= \begin{bmatrix} 0.2361 & 0.14025 & 0.28035 & 0 \end{bmatrix}$. Simply, the player with the highest trust-sus score will be voted out, in this case player 3. Although player 3 suspected player 1 heavily, the trust score of player 3 was relatively low and thus the resulting contribution was low. In contrast, player 2's suspicion towards player 3 was not as high, but was enhanced due to the high trust score of player 2 and his 'opinion' weighing heavy in the voting procedure.

Trust list change after voting

After a vote was cast the players know if they voted right (on the impostor) or wrong. If a player votes right the trust score of this player will change with $\gamma_1(=0.04)$ and if wrong $\gamma_2(=-0.02)$. A player voted right if the player which they suspected the most was indeed the impostor and wrong otherwise. In this case player 1 suspected player 2 the most, which was not the impostor and thus the trust score of player 1 will become 0.28. Player 2 correctly suspected player 3 and thus their new trust score will become 0.44. The impostor never votes for itself and a dead player cannot contribute which is regarded as a wrong answer. The resulting trust list after this vote will thus be: $\begin{bmatrix} 0.28 & 0.44 & 0.18 & 0.48 \end{bmatrix}$.

S2. The visualised model

In figure 5 a screenshot of the visualisation is given. Here, the black cells represent walls, grey obstructions, orange cells short tasks, blue cells common tasks and green cells vents. A visualisation of the run can be seen here.

S3. Further sensitivity analysis results

The sensitivity analysis of the other social variables are given in figure 6. For each of these four parameters, five different values were chosen and used during 160 model runs, 40 model runs per strategy. This was done with both iteration turned on and iteration turned off, resulting in $(5 \times 160 \times 2 =)1600$ model runs per investigated parameter. The shaded areas in the graph represent the standard deviation. It can be seen that the trust parameters γ have the most impact when they are set to 0, meaning the trust score does not change. When this is done, the win-rate for the impostor is somewhat higher. It also seems that the win-rate for the impostor with the Aggressive/Active behaviour is higher when σ_4 has a larger negative value. This could be explained that the Active part makes sure the impostor walks around, and therefore is often in vision of other players, thus decreasing his suspiciousness value more strongly. Interestingly, sometimes changing a variable seems to affect a specific trait of the strategy. For instance, in the iterated games of γ_2 the different behaviour traits (Aggressive/Passive) seems to follow similar patterns when γ_2 is changed. The same goes for the non-iterated σ_5 games, where not the behaviour but the tactic trait (Active/Passive) seems to be explanatory in how the win-rate is affected by a change in σ_5. However, for conclusive results, more data is needed in order to give statistically significant findings.

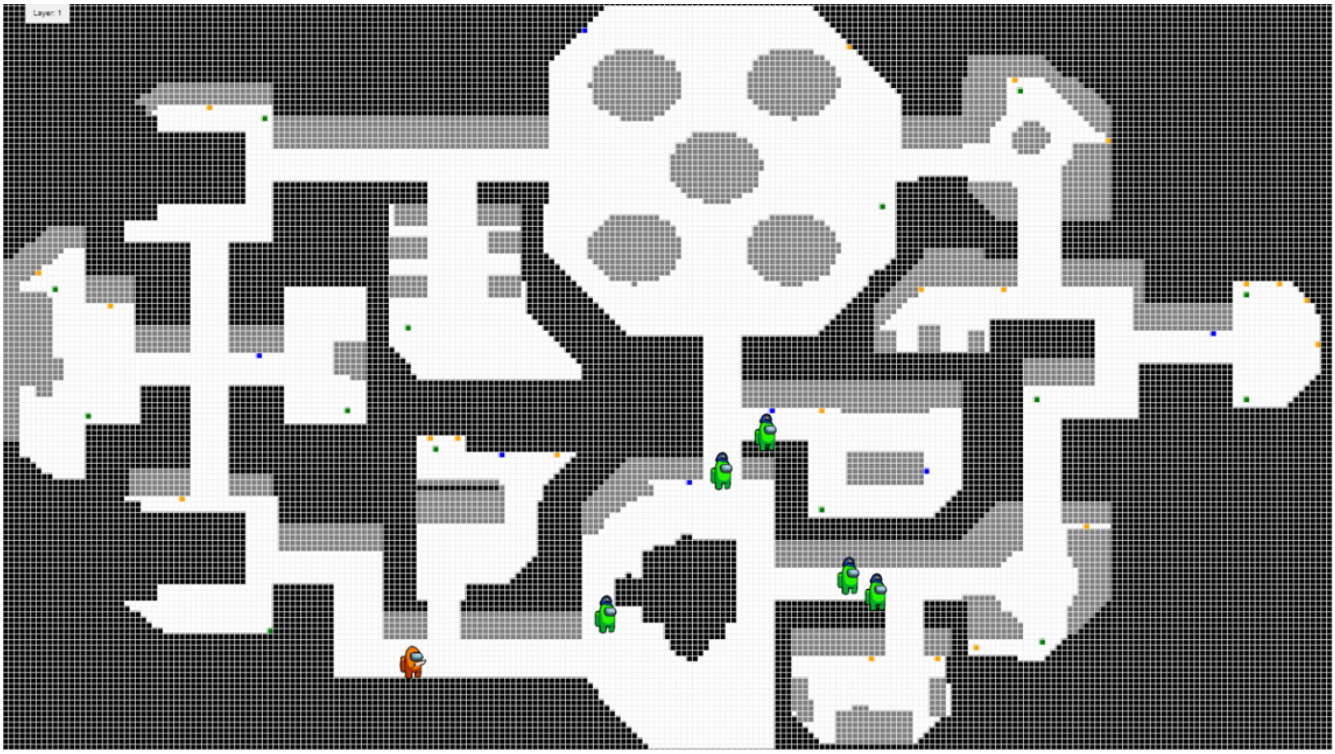
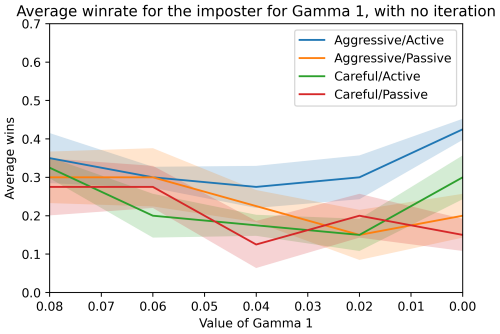
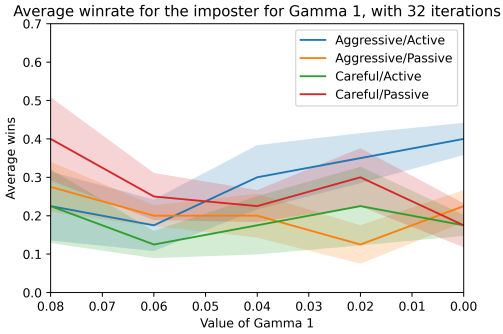


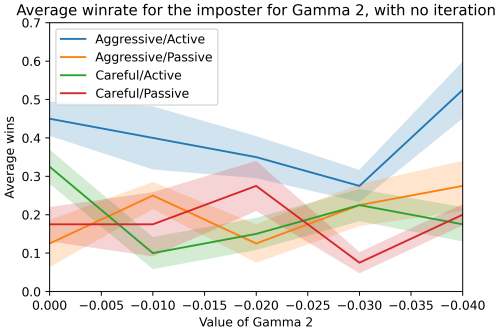
Figure 5. Screenshot of the visualisation, where the black cells represent walls, grey cells obstructions, orange cells short tasks, blue cells common tasks and green cells vents. The crewmates and impostor are also shown.



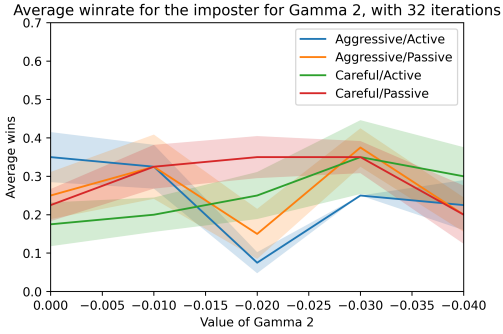
(a) γ_1 no iteration



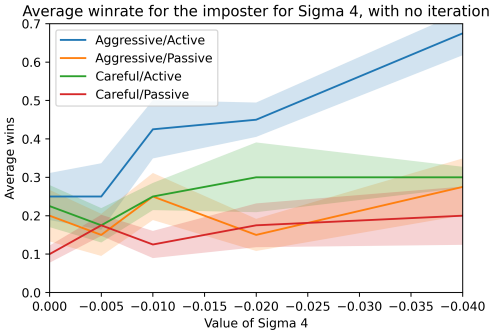
(b) γ_1 with iteration



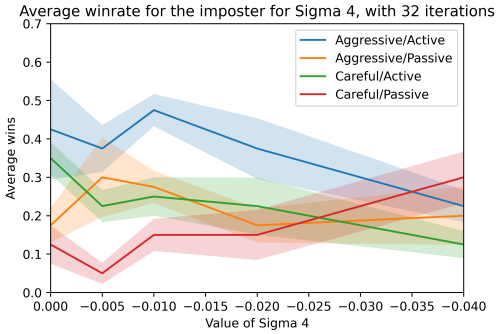
(c) γ_2 no iteration



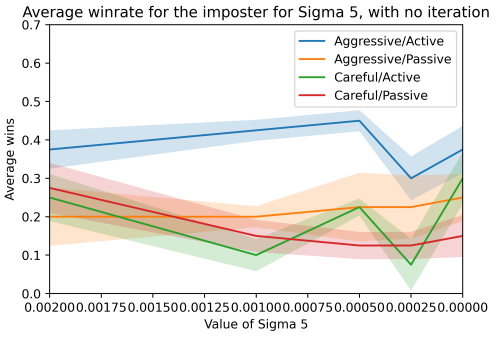
(d) γ_2 with iteration



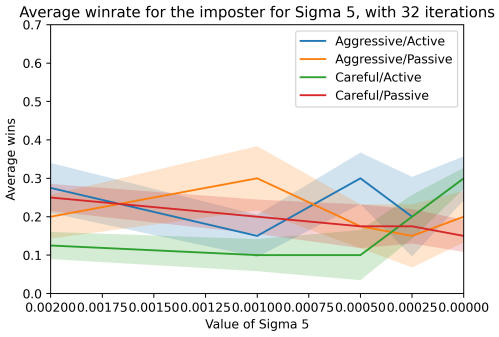
(e) σ_4 no iteration



(f) σ_4 with iteration



(g) σ_5 no iteration



(h) σ_5 with iteration

Figure 6. Sensitivity analysis of the social variables γ_1 (increase of trust for correct vote), γ_2 (decrease of trust for wrong vote), σ_4 (decrease of sus for seeing other agent), and σ_5 (increase of sus for seeing other agentduring game). The shaded area represents the standard deviation.