The University of Western Australia

Resistance AI Report

A CITS3001 Project

Jin Yoaw 22412148

Nicholas Choong 21980614

Resistance Agent

Introduction

The Resistance is a role-play party game played between 5 to 10 players. Each player is assigned randomly to 2 roles which are the resistance and the spies. The role of the resistance is to overthrow the government by going on and completing missions while the spies are hidden amongst the resistance to sabotage the missions. *The resistance* players do not know the roles of other players while the spies know everyone's roles; it is up to the resistance players themselves to deduce other player's roles and play the game according to win. Every mission has a required number of sabotages to fail the mission, if not enough sabotages were acquired during the mission, the mission becomes successful. For every successful mission, the resistance team gets a point and for every mission failure, the spy team gets a point. The first team to reach 3 points wins the game. Mission members are chosen by a randomly appointed team leader at the start of the game.

This report will look at different approaches to creating artificial intelligent agents to *play The Resistance* game as well as discuss the agents that were created. The literature review will address the issues of complete and incomplete information developed by the game and the consequences it carries. The created agents include the `BayesRuleAgent` (Agent 1) and the `BayesBehaviourAgent` (Agent 2). Both the agents were constructed using the Bayesian theorem but were implemented differently and produces different results. A rationale for the agents will be demonstrated along with their validation based on results obtained from simulated games. Other referenced agents that include `ExpertAgent` and `StatAgent` were adapted from the 2012 Vienna Game/AI Conference, The Resistance Competition, to validate our agents through game simulations [1] [2].

Literature Review

The Bayesian theorem and the Monte Carlo Tree Search are techniques that will be discussed in this section which are helpful tools for artificial intelligent agents to decide on the moves or actions they should make during the game to win.

Bayesian theorem

Bayesian theorem is a useful technique to update strategies and beliefs of game playing agents for better decision making (ref). The Bayesian rule is defined as

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

where A and B are events. P(A|B) is the conditional probability of event A given that B is true where A and B are different events [3]. With the implementation of Bayesian theorem, the resistance players will be able to better identify players with the spy role and make decisions around them. The initial probability/rating of players was set as a constant for all players in the beginning of the game which was calculated by

$\frac{1}{Numberofplayers-1}$. From there, a built-in function updates the spy probability/rating according as the game progresses.

Perfect Bayesian Equilibrium refers to a set of strategies and beliefs which are updated via Bayes rule [3], this is applicable to Agent 2 because it needs both strategies and beliefs based on observed behaviour from other players. *The Resistance* is sequentially rational as data is collected as the game progress and probabilities gets updated. Based on the rule of *The Resistance*, both the game types of screening game and signalling games are relevant. Screening games refer to uninformed players making the first move, while signalling games refer to informed players making the first move [3]. When a resistance is the first leader of the game, it is a screening game because resistance players are uninformed by not having any information at the start of the game, while signalling games applies when spies are the first leader as they know everyone's roles.

Separating equilibrium and pooling equilibrium are also applicable for *The Resistance*. Separating equilibrium is defined by uninformed players learning everything about its opponent after observing an equilibrium move, whereas pooling equilibrium is defined by uninformed players being unable to update its beliefs about its opponent after observing an equilibrium move [3]. Separating equilibrium applies when the agent tries to learn about its opponent based on the behaviour which was implemented by Agent 2. Despite that, pooling equilibrium may apply when the resistance player is unable to update its beliefs which is the spy probability after observing moves or behaviours because spies are allowed to mimic resistance actions to conceal their role and cause confusion. These actions are the consequences of uncertainty which applies for human players as well.

Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) algorithm works similar to the minimax algorithm as they both branching off from one node to multiple nodes and each node represent a state of the game [4]. But this is where their similarities stop, as instead of doing depth-first search for each and every one of the nodes, the MCTS will just perform random sampling on the node and try to reach as far as it could before it reaches the terminal node. This is also where its name come from. Constraints are applied to enable algorithm to avoid reaching the terminal state too early. For example, avoiding the ghosts in pac-man or try to eat as much snacks as possible to collect more points.

4 steps are performed for each iteration of MCTS [5]. The first step is selection where is selects the child node that maximises the Upper Confidence Bound 1 applied to trees formula, the selected nodes usually have zero visits which makes their value of UCT infinite. This formula balances the exploration-exploitation trade-off. $\frac{w_i}{n_i} + c\sqrt{\frac{\ln N_i}{n_i}}$ The first component of the formula corresponds to exploitation which is proportional to the average

win ratio whereas the second component of the formula corresponds to exploration which is inversely proportional to the number of nodes being sampled in the simulation. After selecting a node, the second step is expansion where it creates many children nodes but did not have the confidence to select it. This brings us to the third step which is simulating or roll-out. Simulations are performed on each of the child nodes. Constraints are applied to avoid enemies or approach rewards. The further down the simulation goes, the better for the child node. After the simulations, the values calculated backpropagated from the child node all the way to the root node and the number of visits for each of the nodes would increase by one for every visit. The cycle repeats until the agent wins or loses [6] [7].

One way to improve the Monte-Carlo simulation is by incorporating a Model-Opponent technique [8]. The accuracy of the simulations would increase with the Bayes' Rule as it learns the behaviours of its opponents.

Rationale

With high levels of overall uncertainty for *The Resistance* game, it is important that the agent is able to make decisions when data from number of sabotages is insufficient. Techniques such as logic programming will not be sufficient to enable to agent to assess the environment efficiently due to the level of uncertainty present. The Bayesian theorem allows the agent to assess the environment and update the spy probabilities of players when no sabotages occurred in the last mission by calculating the probabilities of a spy given different combination of players. In addition, the implementation of taking behaviours into account allows the agent to better predict the other player's roles and make more accurate decisions.

Bayes' Inference is a good approach to get a starting agent. It is better than the genetic algorithm as it does not require immense resources to learn about the game. It can even be integrated into Monte Carlo Tree Search algorithm to improve the accuracy of the Monte-Carlo simulation [5] [8]. The downside of implementing a MCTS algorithm is that it uses a huge amount of memory after a few iterations of it and the path it chose might not even be the most optimal one due to randomness of it [9].

Implementation

Agent 1: BayesRuleAgent

`BayesRuleAgent` uses the Bayes' Rule to try to find the spies in the game and try to actively prevent them from participating in the missions. The most important methods of the `BayesRuleAgent` are the `bayes_rule`, `get_probabilities` and `update_sus_table`. Some expert rules are used for the spies as they do not utilise the Bayes' Rule to make decisions based on the situation of the game. At the start of the game, the probabilities of being the spies is evenly distributed across all players except the agent that is calculating the probabilities as it knows that it is not a spy. If it is the leader in the first round, then it will randomly select players to form a team. If it is not the leader, then it will vote for the team

as the first round does not provide any information to the resistance players. Once the round is completed, the `update_sus_table` will be called to update the probabilities of each player.

This method is the main function that uses the Bayes' Rule to calculate the probabilities [5]. It does this by getting all the combinations of the spies in a team and then calculating the probabilities of the mission outcome given the combination of the team. As each combination of the team are independent from each other, the calculated probabilities are then summed to get the probabilities of the mission outcome, pB. The probability of the mission outcome given the spies are actually spies, pBa, are calculated in the same way but the combinations of the team are based on the actual number of betrayals that occurred in the mission. There might be combinations where no players are spies as there are no betrayals.

The probabilities of the player being a spy given the mission outcome, pAb, for each player are updated by using the Bayes' Rules. It is calculated by using the probabilities of the player being a spy multiply by the probability of the mission outcome given the player being a spy and then divide by the probability of the mission outcome. These actions are performed for each of the players for every round in the game [5].

Agent 2: `BayesBehaviourAgent`

This agent used a similar technique as Agent 1 where calculations for the spy ratings were done through the Bayes theorem, although a different structure was used. The most prominent difference between this agent and Agent 1 is that the `BayesBehaviourAgent` takes the behavioural actions of other players into consideration. Other player's spy ratings are updated alongside the variables `assistedSpies`, `spyBehaviour` and `resistanceBehaviour`. Spy ratings represent the likelihood of a player having a spy role which is calculated with the Bayesian theorem by looking at the player's probability of being a spy given lists of combinations for players on and off the mission. In addition, there are logical checks that will put player's id to `confirm_spies` or `confirm_resistance` lists which indicate that the following player's roles have been found and actions will be decided accordingly. This is because some scenarios such as when there are only 2 spies in the game and 2 sabotages have been observed from the mission indicates both players are the spies. Details for each behaviour are as follows:

- `assistedSpies`: This variable tracks each player's actions that has assisted the spy during the voting phase. The weight of this behaviour was set to be 0.10 because it indicates the relationship between players; the higher the actions observed, the stronger the relationship between players indicating cohesion which is spy like. Each time a voting phase is complete, if the player has voted for a team that is likely to fail based on the total spy rating of the team or the player has voted against a team likely to succeed, the player has assisted the spies. At the end of the round, if this

player has assisted the spies, their spy rating would be increased as an external behavioural influence to calculated Bayes spy rating.

- `spyBehaviour`: This variable tracks each player's actions that are likely to be spies. The weight of this behaviour was set to be 0.20 which is the highest of the 3 behaviours as these actions are more distinctive for spies. Each time a player votes for a team they are not on, they are likely a spy because a resistance would not vote for a team that is likely to fail. Another scenario would be rejecting the first mission of the game where the mission only consists of resistance because resistance players would have high uncertainty while spies know roles of the players in the mission and do not want the first mission to be successful. At the end of the round, if this player was seen with spy like behaviours, their spy rating would be increased as an external behavioural influence to calculated Bayes spy rating.
- `resistanceBehaviour`: This variable keeps tracks of each player's actions that may be associated to resistance like behaviours. Because this is a highly uncertain behaviour as spies can mimic resistance behaviours intentionally to cause confusion, the effects of this influence are low with a 0.05 change per action observed. At the end of the round, if this player was observed to have resistance behaviours, their spy rating would be decreased as an external behavioural influence to calculated Bayes spy rating.

Each of these behaviours contribute to the final spy rating of any given round by multiplying a calculated weight value and updates to the `spy_rating` list. With a Bayes theorem application alongside behavioural implications, the agent is now able to make more accurate decisions based on the spy ratings. The spy rating is used as a decision-making mechanism for resistance players to propose missions and decide if they should vote for or against missions. It was also implemented as a resistance perspective for spies to choose less suspicious players on missions they propose to increase the chances of the mission going through after they have picked enough spies for the mission to be sabotaged.

The limitation for this specific implementation due to the internal `_choose_team` function is the role placement for players in a given game. If by chance, all spies are located in the front indexes, the resistance team will have 0% to 30+% chance of winning dependent on the number of players due to the sequence of gameplay. The spy players will be able to play their moves effective and are able to dictate the game as they know player roles which represents low uncertainty compared to the resistance players who have high uncertainty due role ambiguity. Furthermore, if all spies are placed to the front, the resistance players do not have the authority to propose missions and are unlikely to oppose the mission proposals early in the game and some assisting spies' behaviours can be observed from resistance players due to wrong actions made in the beginning of the game with little to no knowledge, this negatively impacts the resistance perspective for other resistance players as well.

Agent 3: ExpertAgent [2]

The ExpertAgent is a simplification of the Invalidator. It maintains scores of each combination of the players given the number of spies in the game and keeps track of the outcomes of the mission proposal, the mission voting, and the mission itself [8].

Agent 4: StatAgent [1]

The StatAgent is another agent that uses Bayes' Theorem to score the players based on their mission proposals, mission voting and their performance in the mission. It does not utilise any expert rules to perform an action, instead it is uses suspicion scores for each of the players to make an action [8].

Validation

To validate our agents, we have decided not to play against RandomAgents because they are inappropriate validators due to random the random actions, they make which are poor performance indicators for the created agents. The following simulations were conducted with fixed resistance and spy player indexes where player index 0 is a resistance, followed by spies incrementally and lastly all other resistance players. The following simulations were conducted with fixed resistance and spy player indexes where player index 0 is a resistance, followed by spies incrementally and lastly all other resistance players.

Performance:

- 10,000 simulations for 5 players took ±10 seconds
- 10,000 simulations for 10 players took ±32 seconds.

Game simulation of agents against themselves

These simulation results show the win rate for the respective agents with resistance roles for 10,000 games with the range of 5 to 10 players. This will indicate which agent has better decision making as resistance players.

BRAgent against itself (Agent 1)	Res Winrate
5	14.68%
6	24.23%
7	15.06%
8	30.48%
9	44.40%
10	52.82%

BBAgent against itself (Agent 2)	Res Winrate
5	5.04%
6	20.12%
7	27.41%
8	35.86%
9	48.59%
10	56.41%

Based on the results shown on the 2 tables, it can be seen that Agent 1 has a better resistance win rate compared to Agent 2 when they are 5 or 6 player games. Agent 2

performs better at 7 to 10 player games. This indicates that the behavioural implementation becomes is more effective with higher number of players, jumping from ~5% to ~20% with an increase of 1 player and subsequently higher win rates as player number increases. A factor to keep in consideration is the different implementation and conditionals both agents have despite using a similar framework for the Base calculation. Some results may be unidentified outliers due to the different rule sets applied for the agents.

Game simulation of mixed agents

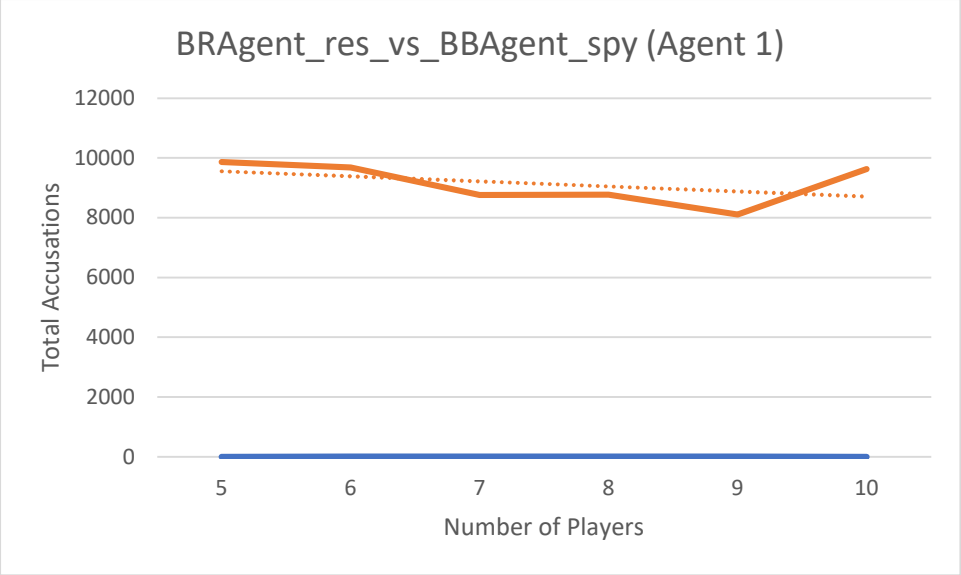
The following table shows the win rate of the resistance team from a mixed agent simulation where Agent 1 and 2 switch roles to show their performance level for 5 player games. The first 3 agents are fixed with resistance role and the last 2 agents are spies. They played 10,000 games with the same roles to produce the shown result. This simulation was restricted to 5 player games due to the limitations of the `expertAgent`.

Agent Players	Win rate
ExpertAgent, BBAgent, StatAgent as res against ExpertAgent, BRAgent as spy	1.10%
ExpertAgent, BRAgent, StatAgent as res against ExpertAgent, BBAgent as spy	0.13%

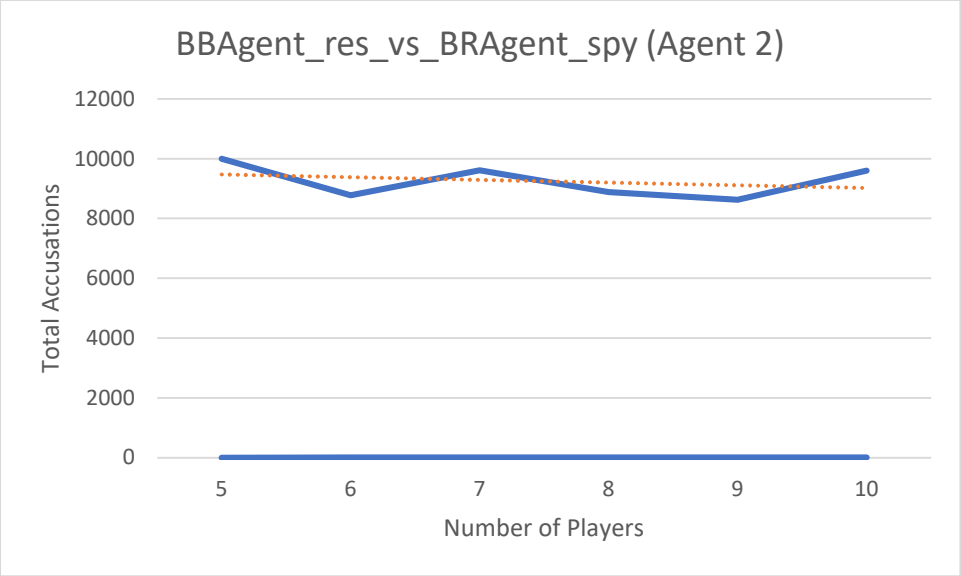
From the results produced, it can be seen that given the same three opponents, Agent 2 performed better by almost 10 times compared to Agent 1. The reason for this difference in result is the consideration of behavioural observation and implementation. Agent 2 was better enabled to identify spies given the group of players in the simulations. Other than that, a factor that may be impacting the results may be the agent's ability to conceal itself as a spy during the games.

Game simulation of Agent 1 vs Agent 2

These simulation results demonstrate the accuracy of a resistance player accusing a spy of being a spy correctly for 10,000 games with the range of 5 to 10 players. This will demonstrate the accuracy of a resistance player accusing a spy of being a spy correctly for 10,000 games. Accusation number can range from 0 to 10,000, a higher number represents a better accuracy of an agent's resistance ability to identify a spy.

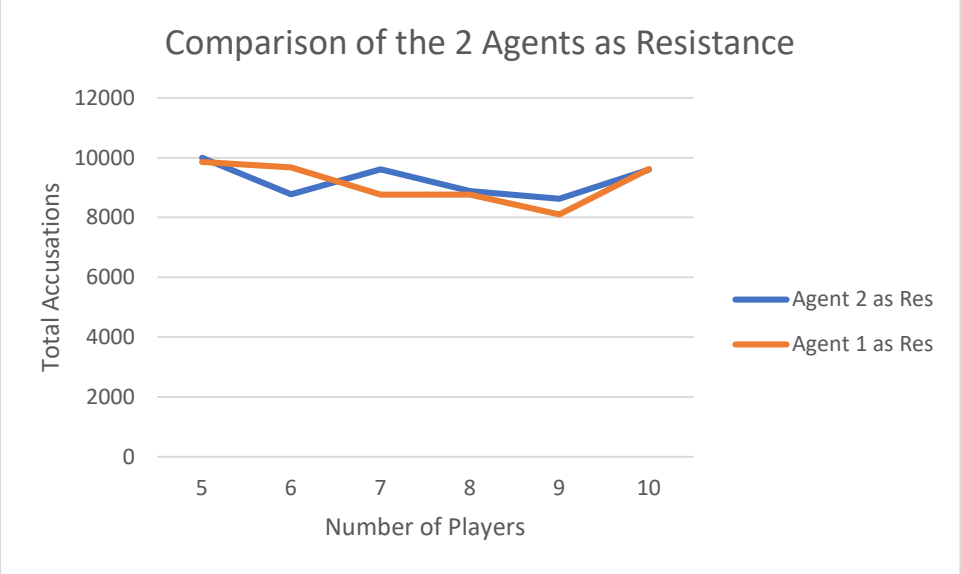


From the results obtained, Agent 1 was able to accuse a spy correctly with ~98% accuracy for the 5-player game and subsequently decreased in accuracy as number of players increased to a lower ~81%. This trend represents a relatively weak negative relationship between accuracy and number of players. Although, it can be observed that when the ratio of spies increases when compared to the previous game, the accuracy increases slightly. This is due to the increased probability for the resistance player to guess a spy correctly within the game.



From the results obtained, the resistance agent was able to accuse the spy agent correctly 10,000 times which indicates a 100% accuracy in the given scenario. Based on the line graph, a trend can be seen where accuracy falls as the number of ratios of resistance player increases compared to the previous simulation. Whereas accuracy increases when the ratio of spy player increases when compared to the previous simulation. These results are due to

the change in level of uncertainty in the game. With more resistance members, role ambiguity increases for the resistance team as they are required to find the spies sequentially and that spies are able to mimic resistance behaviour to create confusion leading to higher inaccuracy. In contrast, a higher ratio of spies in the game allows the resistance members to guess more accurately due to higher chances of spies showing spy like behaviours to increase suspicion levels from a resistance perspective. Other than that, the trendline of Agent 2 can be seen to have a very weak negative slope which shows that the agent is less affected by the increase in number of players when compared to Agent 1.



As seen on the graph, Agent 2 more accurately identifies a spy within the simulation on average with the exception of a 6-player game. The given result was caused by the implementation of behavioural observations which influences the spy rating of players respectively. This shows that by adding behavioural considerations to the spy rating allows the agent to better identify spies and therefore will be allowed to make better decisions.

Conclusion

Based on our findings, we can conclude that Bayes theorem allows agents to find a good move or action, but it may not be an optimal move. More complex opponent modelling implications are required to allow agents to find better or the optimal move given a scenario and environment. In consistent results can also be seen when agents of different levels are put in the same game. This is due to knowledge discrepancy between agents as they learn or model opponents at different levels. Different agents with similar levels of opponent modelling would work best to produce expected or good results. Monte Carlo Tree Search could be used to expand the search state, thus improving the win rate of our agents.

References

[1] AiGameDev, *StatAgent,* THE RESISTANCE Competition, Vienna Game/AI Conference 2012, 2012.

[2] A. J. Champandard, *ExpertAgent,* THE RESISTANCE Competition, Vienna Game/AI Conference 2012., 2012.

[3] W. Spaniel, "Perfect Bayesian Equilibrium," Game Theory 101, 18 July 2018. [Online]. Available: http://gametheory101.com/courses/game-theory-101/. [Accessed 6 October 2021].

[4] G. Strong, "The minimax algorithm," 2011. [Online]. Available: https://www.cs.tcd.ie/Glenn.Strong/3d5/.

[5] D. Whitehouse, "Monte Carlo Tree Search for Games with Hidden Information and Uncertainty," 2014.

[6] C. Isbell, "Monte Carlo Tree Search," Udacity, 7 June 2016. [Online]. Available: https://www.youtube.com/watch?v=onBYsen2_eA. [Accessed 5 October 2021].

[7] J. Levine, "Monte Carlo Tree Search," 6 March 2017. [Online]. Available: https://www.youtube.com/watch?v=UXW2yZndl7U&t=630. [Accessed 5 October 2021].

[8] D. P. Taylor, "Investigating Approaches to AI for Trust-Based, Multi-Agent Board Games with Imperfect Information With Don Eskridge's "The Resistance"," 2013-2014.

[9] R. Roy, "Monte Carlo Tree Search," GeeksforGeeks, 14 January 2019. [Online]. Available: https://www.geeksforgeeks.org/ml-monte-carlo-tree-search-mcts/.