# **“The Resistance”: AI Project Report**

By Jake Lyell (22704832) and Yi Ming Liu (22730325)

*Does the consideration of Expert Rules improve the performance of a game playing agent in “The Resistance”?*

“The Resistance” is a multi-player, social deduction game, in which players are randomly selected to be government spies, or resistance fighters, with the goal of thwarting or succeeding missions respectively. A third of the players in the game are selected as government spies, their goal is to deceive the resistance and fail their missions. The rest of the players are members of the resistance, who are unaware of who the spies are and are trying to succeed their missions. This means that “The Resistance” is an imperfect information game, as not all information about the game state is known to all players at each turn. Players without comprehensive information about the game state will have to use deduction to narrow down which possible state the game is currently in. As the game proceeds, more information can be discerned, which can allow for more accurate deductions about the game state. “The Resistance” is of particular interest to the development of game playing AI, due to its apparent simplicity, lack of randomness and its focus on logical and social deduction. This report provides an overview of past research on game playing agents for “The Resistance”, as well as suitable techniques for playing the game, and a comparison of the performance of 2 agents, one of which, utilizing “Expert Rules”.

Gameplay in “The Resistance” centres heavily around probabilities as there is a large amount of uncertainty within the game. Each Resistance member has no idea which players are spies and which are resistance members at the start of the game. Resistance members must use deductive reasoning to narrow down who the spies might be, and to not let them sabotage missions. Due to players not having complete information about the game state, the conventional techniques used to solve perfect information games like the minimax and alpha-beta pruning algorithms cannot be used easily (Blair, Mutchler, & Liu, 1993; Koller & Pfeffer, 1995). Koller and Pfeffer (1995) noted that game theory can be applied to help solve these imperfect information games. One commonly used method in game theory is the application of Bayes’ Rule. Other methods that could be considered includes Monte Carlo Tree Search and the addition of Expert Rules.

Monte Carlo Tree Search (MCTS) is an artificial intelligence methodology that has been successfully applied to agents in the playing of imperfect information games (Lisý, 2014). Highly advanced games such as Go have been optimized using a MCTS algorithm (Whitehouse, 2014). The most famous example of which is DeepMind’s AlphaGo, which was the first AI to beat a professional human Go player. Monte Carlo Tree Search is similar to a depth first search of a tree, except that it also includes other steps which make it significantly faster in finding good game states. An MCTS algorithm performs a series of operations to update the nodes in the tree in order to find the best solution. These operations are Selecting, Expanding, Simulating and Updating (Wang, 2021). First you select the node with the highest possibility of winning, out of currently visible nodes. You then expand upon the selected node, generating child nodes from that node. We then simulate games using reinforcement learning, making randomized decisions further down from the child nodes. The decisions that performed the best will be used to score the path of the child nodes. After simulation we go back and update the score of each nodes parent node, all the way back up the tree. The final result is the best possible move from the current game state (Wang, 2021). MCTS has been used to successfully play many games like Poker and Magic: The Gathering (Whitehouse, 2014), both imperfect information games like The Resistance.

Bayes’ Rule is a simple formula used for calculating conditional probabilities based on prior knowledge of factors that are related to the event (Horgan, 2016; Joyce, 2003). This method can be used to update your current world view. Bayes’ Rule is a very good method of deducing facts from imperfect knowledge, for example, given the prior chance that a single person out of the population has cancer is 10%, and they receive a positive test result, we can calculate that person’s chance of having cancer based on the sensitivity and specificity of the test. Bayes’ Rule can be difficult to understand without a concrete example, we go further into detail about Bayes’ Rule below. Bayes’ Rule is a great method for playing a Bayesian Game, which is defined as a game in which players have incomplete information about the other players (Java T Point, 2021). The Resistance can certainly be categorized as a Bayesian Game, and so we decided to implement Bayes’ Rule as the main method for deciding on actions within the game.

Rule-based systems are a basic form of artificial intelligence that uses human-like expert knowledge to solve a specific problem (Grosan & Abraham, 2011). Expert Rules are characterized by performing an action based on a set input, regardless of the other factors in play. For example, an agent may choose to vote no for a mission proposal if they are not featured on a team, regardless of the chance that the mission may contain a spy. This is an Expert Rule and will take precedent over the probabilities normally in charge of calculating the best course of action. It is infeasible to create an agent who acts purely on Expert Rules alone, however Expert rules have shown to be effective when used in tandem with another method for game playing, as detailed below.

In a paper analysing the performance of game playing agents for “The Resistance”, Taylor (2014) noted that the best performing agents that all used some “Expert Rules”. Taylor (2014) identified the following as common Expert Rules:

* As a leader, agents should ensure they themselves are a part of the proposed mission. As a resistance member, including yourself in the team minimizes the chance of the team having a spy as your own identity is known.
* Agents should always vote to pass the fifth mission proposal. This is because the game states that after 5 failed votes, the mission is failed automatically, thus there is no advantage for resistance members to fail the 5th proposal even if they believe it will result in a failed mission.
* Spies should avoid missions with more than the required number of spies to fail the mission. This is because a mission failing with more than the required betrayals will reveal extra unnecessary information to other players.
* Spies should always sabotage a mission after 2 failed missions have already occurred, as the third sabotage will result in the game ending and a win for the spies.

(Taylor, 2014) also writes that there is no reason to reject the first mission proposal as no information is yet known about player roles, however, this does conflict with the first expert rule, where players should generally try to vote for teams that include themselves. In a Board Game Geek forum, a user (McKenzie, 2010)(McKenzie, 2010) agrees with this sentiment, contending that even on the first mission proposal, a player should vote no if they are not on the team. He argues that players that vote yes to a mission they are not on in the first round means that they are more likely to be a spy trying to approve a mission with another spy. In the same forum, Rockwell (2010) proposed the following strategies:

* In a round that requires the same number of agents to go on a mission as there are resistance members, any resistance member that is not on the proposed mission should always vote no, as there is guaranteed to be a spy on that mission
* As a spy, it can be advantageous to vote to pass a mission with few players, and later sabotage a mission with more players, as this will create more uncertainty in who the spies are.

Implementing only these expert rules was found to be somewhat effective against ‘beginner’ opponents (Taylor, 2014), but a method of further evaluating the available information is required for a more competent agent.

## Design Description and Method Rationale:

In “The Resistance”, players can model the game’s possible states as a set of worlds that could be true. We define a world to be a unique combination of players who are the spies within a game. For example, the world (2,4) refers to the circumstance in which players 2 and 4 are the 2 spies present in the game. We can then rewrite Bayes’ Rule to have more meaningful variable names. Below is Bayes’ Rule:

In our example:

* is the probability of the world being true given that a failed mission has occurred.
* is the probability of a mission failing given the world is true.
* is the probability of the world being true.
* or is the overall probability of a mission failing, which is equal to the product of the probability of a world being true and the probability of the mission failing in that world.

This formula can be used to update the probabilities of each world being true after a mission fails or succeeds. A mission failing provides the agents with two pieces of information:

* The set of players who were on the mission which failed.
* The number of betrayal votes which caused the mission failure.

This allows the agent to discern the values of and in turn calculate the value of .

Take the following scenario of a game with 6 players for example:

There are always 2 spies in a 6-player game, and so there are 6C2 = 15 possible combinations of spies. Let each player be denoted by a number from 1 to 6, and let number 1 be a resistance member

Number 1 can choose to update the probability table of the initial worlds, knowing that worlds that contain themselves as a spy cannot be possible, and such will have a probability of zero:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| World | P(W) | World | P(W) | World | P(W) | World | P(W) | World | P(W) |
| 1, 2 | 0 | 2, 3 | 0.1 | 3, 4 | 0.1 | 4, 5 | 0.1 | 5, 6 | 0.1 |
| 1, 3 | 0 | 2, 4 | 0.1 | 3, 5 | 0.1 | 4, 6 | 0.1 |  |  |
| 1, 4 | 0 | 2, 5 | 0.1 | 3, 6 | 0.1 |  |  |  |  |
| 1, 5 | 0 | 2, 6 | 0.1 |  |  |  |  |  |  |
| 1, 6 | 0 |  |  |  |  |  |  |  |  |

Suppose agents 2 and 3 go on a mission, and the mission fails with 1 fail vote. For each world, is the probability of a mission with 2 and 3 failing if we assume the world is real and can be calculated using the following method:

* Obtain the number of agents that exist in both the world set and the mission set (the overlap). E.g. if the world is (2, 4), then the common agent is agent 2, and the overlap is 1.
* Obtain the total amount of outcomes the mission could have had (total). E.g. for a mission with 2 agents, the possible outcomes are: [(Success, Success), (Fail, Success), (Success, Fail), (Fail, Fail)], where each tuple corresponds to the same tuple of agents.
* is then the overlap divided by the total. E.g. if the world (2, 4) is real, then the only way a mission with 2 and 3 fails is if agent 2 fails. Hence, giving a value of .

Applying this for all worlds results in the following:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| World | P(F|W) | World | P(F|W) | World | P(F|W) | World | P(F|W) | World | P(F|W) |
| 1, 2 | 0 | 2, 3 | 0.5 | 3, 4 | 0.25 | 4, 5 | 0 | 5, 6 | 0 |
| 1, 3 | 0 | 2, 4 | 0.25 | 3, 5 | 0.25 | 4, 6 | 0 |  |  |
| 1, 4 | 0 | 2, 5 | 0.25 | 3, 6 | 0.25 |  |  |  |  |
| 1, 5 | 0 | 2, 6 | 0.25 |  |  |  |  |  |  |
| 1, 6 | 0 |  |  |  |  |  |  |  |  |

And so,

Now to find :

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| World | P(W|F) | World | P(W|F) | World | P(W|F) | World | P(W|F) | World | P(W|F) |
| 1, 2 | 0 | 2, 3 | 0.25 | 3, 4 | 0.125 | 4, 5 | 0 | 5, 6 | 0 |
| 1, 3 | 0 | 2, 4 | 0.125 | 3, 5 | 0.125 | 4, 6 | 0 |  |  |
| 1, 4 | 0 | 2, 5 | 0.125 | 3, 6 | 0.125 |  |  |  |  |
| 1, 5 | 0 | 2, 6 | 0.125 |  |  |  |  |  |  |
| 1, 6 | 0 |  |  |  |  |  |  |  |  |

Since the condition of is known (that is, the mission consisting of agents 2 and 3 has failed), is then the same .

## Design description:

2 Agents were implemented for this Project, named BasicBayes and BayesJond. These agents are both implemented using Bayes’ Rule, however, BayesJond also utilizes “Expert Rules”. Bayes’ Rule is used to calculate the probability of each world being the real world, and this information is then used to assign a ‘suspiciousness’ value to each player. This suspiciousness value is defined as the sum of the probabilities of the worlds containing a certain player divided by the total number of worlds where the player is a spy. That is:

These suspiciousness values are used to inform agents’ mission proposals and voting behaviour. The following page contains a table that explains and compares the behaviour of each agent when playing as each of the two roles.

|  |  |  |
| --- | --- | --- |
|  | **BasicBayes** | **BayesJond (007)** |
| **Resistance behaviour** | Initialises all world probabilities to be equal, but removes worlds containing itself as spy. | Initialises all worlds probabilities to be equal and keeps worlds where self is spy. |
| Calculates suspiciousness of other players using world probabilities. | Calculates suspiciousness of other players using world probabilities. |
| Proposes missions by selecting players with the lowest suspicion values  Since the probabilities of the worlds containing itself are 0, proposed missions will likely always contain itself. | Proposes missions by selecting itself and then the players with the lowest suspicion values. |
| Votes for mission proposals by ranking all players by their suspiciousness. The agent will not approve a team proposed by or consisting of players in the *n* most suspicious players. Here, *n* is the number of spies | Votes for mission proposals by checking the suspiciousness of each player and compares this to the average suspiciousness of all players. If a player is above the average, missions they are a part of, or missions that they propose will not be approved. The following expert rules take precedent over this methodology:   * The agent will always approve the 5th mission proposal * The agent will approve of mission proposals by other players that have a suspicion of 0 * The agent will always fail missions they are not on if the mission size is equal to the number of resistance members. |
| **Spy behaviour** | Initialises all worlds probabilities to be equal to track resistance perception of self and other spies | Initialises all worlds probabilities to be equal to track resistance perception of self and other spies |
| Calculates suspiciousness of other players using world probabilities | Calculates suspiciousness of other players using world probabilities |
| Proposes missions by always selecting the required number of spies for a mission to fail. The chosen spies are the least suspicious spies. | Proposes missions by always selecting the required number of spies for a mission to fail. The chosen spies are the least suspicious spies but will always include itself. |
| Vote as if it were a resistance member, but will not approve missions containing all spies | Votes as if it were a resistance member but will not approve missions containing all spies. The only expert rule that applies is that the agent will always approve the 5th mission proposal |

## Results:

We tested both the BasicBayes agent, and the BayesJond agent against each set of opponents 20 000 times for each game size from 5 to 10 players, for a total of 960 000 games.

We tested both the simple Bayes Agent (BasicBayes) and the Bayes Agent with Expert Rules (BayesJond) against multiple sets of opponents. We tested against teams consisting of; all random agents, all Basic Bayes agents, all BayesJond agents, and a randomized combination of the 3 types of agents. These tests aim to achieve the following:

* Compare the performance of the 2 agents against different types of opponents
* Compare how the agents differ when working together with itself

The tests were also run with the base random agent given to us, as a basis for comparison, and to verify that the implemented agents perform better than a random one. A total of 1 440 000 games were simulated, and the win rates for each agent are shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Other Players in the game** | | **BasicBayes** | **BayesJond** | **Random** |
| **All Random Agents** | **Resistance Win Rate** | 83.395 % | 82.982 % | 78.698 % |
| **Spy Win Rate** | 56.303 % | 60.935 % | 21.295 % |
| **Overall Win Rate** | 73.184 % | 74.666 % | 57.060 % |
| **All BasicBayes Agents** | **Resistance Win Rate** | 18.856 % | 35.669 % | 17.154 % |
| **Spy Win Rate** | 81.358 % | 81.593 % | 51.672 % |
| **Overall Win Rate** | 42.513 % | 53.069 % | 30.219 % |
| **All BayesJond Agents** | **Resistance Win Rate** | 31.551 % | 36.349 % | 30.972 % |
| **Spy Win Rate** | 55.056 % | 64.526 % | 41.369 % |
| **Overall Win Rate** | 40.457 % | 47.042 % | 35.014 % |
| **Random combination of all 3 agents** | **Resistance Win Rate** | 44.235 % | 52.217 % | 42.440 % |
| **Spy Win Rate** | 56.136 % | 61.091 % | 35.378 % |
| **Overall Win Rate** | 48.746 % | 55.569 % | 39.772 % |

The results show that the BayesJond agent had the best overall performance, with the highest overall win rates for each test, and only losing to BasicBayes for Resistance win rate in the “all random agents” test. BayesJond also had the most positive overall win rates, recording only a negative win rate when paired with itself. As expected, both agents performed significantly better than the random agent for all tests.

The BasicBayes agent appeared to work poorly with itself when playing as resistance, achieving only an 18% win rate. BayesJond on the other hand cooperated much better with both itself and other BasicBayes agents, achieving a resistance win rate of around 36% for both. It appears that the BayesJond agent makes for a better resistance teammate, with both BasicBayes and Random agents recording a higher resistance win rate when playing with all BayesJond agents.

Overall, the 2 agents had stronger performances as spies than as resistance, except for when against random agents. This is likely due to both agents utilize a spy behaviour which attempts to play as similar to a resistance member as possible, making it much harder for the resistance members to identify the spies. BayesJond also displayed stronger spy competency, matching the spy win rates of BasicBayes against other BasicBayes, and outperforming BasicBayes by a large margin in the other tests.

## Conclusion:

The implementation of “Expert Rules” in BayesJond seemed to only be effective against competent opponents, with BasicBayes and BayesJond having similar win rates when playing against random agents. However, against stronger opponents, it can be concluded that the consideration and utilisation of “Expert Rules” does provide a significant advantage to the agent, with BayesJond clearly being the strongest agent of the 3.

## Future Research Considerations:

We noted the following items that could be of interest for future research into agents for “The Resistance”

* While the BasicBayes agent did not have any “Expert Rules” explicitly implemented, the BasicBayes agent would always include itself when proposing missions as a resistance member. This is due to the agent setting the probability of worlds including itself as a spy to 0, which means it will always have itself as the least suspicious player when selecting players to go on a mission. It could be interesting to what impact actively excluding this “Expert Rule” from BasicBayes would have on its performance.
* The tests conducted in this paper shed some light onto how well the agents work together when testing them with themselves. This test could be further extended to test the performance of a resistance or spy team consisting of the same agents against different agents to obtain a more definitive conclusion on how well it cooperates with itself.

# References:

Blair, J. R. S., Mutchler, D., & Liu, C. (1993). Games with Imperfect Information. *AAAI Technical Report, FS-93-02*.

Grosan, C., & Abraham, A. (2011). Rule-Based Expert Systems. In *Intelligent Systems: A Modern Approach* (pp. 149-185). Berlin, Heidelberg: Springer Berlin Heidelberg.

Horgan, J. (2016). Bayes's Theorem: What's the Big Deal? *Scientific American*.

Java T Point. (2021). Bayes theorem in Artificial Intelligence - Javatpoint.

Joyce, J. (2003). Bayes’ Theorem. In *Stanford Encyclopedia of Philosophy Archive (Spring 2019 Edition)*.

Koller, D., & Pfeffer, A. (1995). *Generating and solving imperfect information games*. Paper presented at the Proceedings of the 14th international joint conference on Artificial intelligence - Volume 2, Montreal, Quebec, Canada.

Lisý, V. (2014). *Monte Carlo Tree Search in Imperfect-Information Games*.

McKenzie, C. (2010). The Resistance Strategy thoughts | The Resistance. In. Board Game Geek.

Rockwell, A. (2010). The Resistance Strategy thoughts | The Resistance. In. Board Game Geek.

Taylor, D. P. (2014). Investigating approaches to ai for trust-based, multi-agent board games with imperfect information; with don eskridge’s “the resistance”. *Discovery, Invention & Application*(1).

Wang, B. (2021). Monte Carlo Tree Search: An Introduction. Retrieved from <https://towardsdatascience.com/monte-carlo-tree-search-an-introduction-503d8c04e168>

Whitehouse, D. (2014). *Monte Carlo Tree Search for games with Hidden Information and Uncertainty.* (PhD), University of York, Retrieved from <https://etheses.whiterose.ac.uk/8117/>