Lauren Gee
21495335

The Resistance:

AI Tournament

Literature Review

The Resistance is a hidden role game. Some players are spies, who have complete information, and need to deceive the resistance players. The remaining players are members of the resistance, who don't know who the spies are, so have incomplete information about the game state. The resistance players need to work out who the spies are, so that they can make the right decisions to win the game. Without complete information, the game state could be any one of many possible worlds. As more information is gathered, epistemic logic can be used to eliminate worlds that are impossible. To do this, there must be a set of rules that can be used to distinguish between the possible worlds (van Ditmarsch & Kooi, 2015).

Deductive logic, which uses values that are either true or false, only works when those values have no uncertainty. Using only deductive reasoning limits the amount of useful information we can gather in scenarios with a high degree of uncertainty. Probability theory is a better framework for reasoning in the case of *The Resistance*, since this game involves a lot of uncertainty. Inductive logic is a generalisation of deductive logic, which applies Bayesian probability to deal with uncertainty. If a statement has a probability of 1, then that is the same as something being true in deductive logic. Similarly, if a statement has a probability of 0, then it is false. Otherwise, statements will have a probability value between 0 and 1 (Oaksford & Chater, 2009).

Gill (2011) highlights the difference between actions and probabilities, using the example of the Monty Hall problem. If we only use Bayesian probability to model a scenario, this will predict the most likely behaviours for random opponents. However, to play optimally in a multi-agent scenario, each agent is going to use some strategy rather than random chance. The Monty Hall problem asks you for an action, not a probability. By using probabilities to model the problem, we are showing the likelihood of our opponent making particular decisions. The accuracy of our model is only as good as how accurately we model our opponent's behaviour.

Bayes' theorem describes the relationship between two conditional probabilities, $P(A \mid B)$ and $P(B \mid A)$:

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

Cadwallader Olsker (2011) uses the example of a test for the presence of a disease. The accuracy of this test is more complex than a single probability value, such as 95%. Instead we need to consider two cases: the accuracy of the test when the disease is present, and the accuracy of the test when the disease is not present. In this example, the probability of the disease being present $P(disease)$ is 1%. If the disease is present, then the probability of a positive test result $P(positive \mid disease)$ is 80%. If the disease is not present, then the probability of a positive test result $P(positive \mid no\ disease)$ is 9.6%. To calculate $P(positive)$, we need to add up the probabilities of the test giving a positive result in each case, weighted by the probability of each case occurring.

$$P(positive) = P(positive \mid disease) \times P(disease) + P(positive \mid no\ disease) \times P(no\ disease)$$
$$P(positive) = (0.8 \times 0.01) + (0.096 \times 0.99) = 0.10304$$

Now, let's use Bayes' theorem:

$$P(disease \mid positive) = P(positive \mid disease) \times P(disease) \div P(positive)$$
$$P(disease \mid positive) = 0.8 \times 0.01 \div 0.10304 = 0.07764$$

This shows that when there's a positive test result, there is only a 7.8% chance of the person actually having the disease. The logic of Bayes' theorem can be confusing and counter-intuitive. Cadwallader Olsker (2011) explains the importance of interpreting a problem correctly, so that the correct solution can be reached.

The genetic algorithm can be used to optimise the parameters of a strategy. To do this, the strategy must be parameterised. Each variation of a parameter will have an effect on how a strategy behaves. Lin and Ting (2011) use a representation of chromosomes, encoding the parameters as genes. To develop their strategy for tactical formations in the game *StarCraft*, they used many different variations of the same strategy, each with its own unique chromosome. The success of each chromosome determines whether or not its genes will be used to produce the next generation of chromosomes. The best combinations of genes will be kept, and less successful genes will be eliminated. Over the course of many generations, the strategies will be improved towards the optimum.

Selected Technique

Using epistemic logic, we could create a set of rules for how to deduce knowledge about the game state. However, since *The Resistance* has a high degree of uncertainty, it is very difficult to reach any definitive true or false answers. Most in-game actions, such as nominations, voting, and accusations, could happen if the player is a member of the resistance or if they are a spy. Only a mission failure can give solid information, since a resistance player can't fail a mission. Until there is some solid information, a resistance player using epistemic logic won't be able to distinguish between good and bad decisions. Within the course of a five round game, it is unlikely that a resistance player would be able to identify all of the spies before the spies achieve three failed missions.

Since *The Resistance* involves a lot of uncertainty, it would be more appropriate for our strategy to work with probabilities rather than true or false values. Using probabilities won't

give us any solid information, but it will allow us to identify which decisions are more likely to be good decisions. Because of this, we can gather some useful information even when there are uncertainties involved. Bayes' theorem is a really powerful tool for resistance players, since it allows them to convert between different conditional probabilities. For instance, if they know the probability of each world being the true game state, and if they can estimate the probability of a team failing in each of those worlds, then they could use Bayes' theorem to work out the overall probability of a team being successful. This would allow them to compare between different teams, which would help them to make decisions about nominations or voting.

Using probabilities and Bayes' theorem would be less useful for spy players, since the spies don't have any uncertainty. On the other hand, it is a useful strategy for spies to pretend to be members of the resistance. If the spies maintain the probabilities as if they are a resistance player, they can use this as a guide for how a resistance player would behave. If they act like a resistance player whenever possible, this will limit the amount of information that the resistance players get, since the resistance players will be looking for suspicious spy-like behaviour. Of course, the spies will also have to fail as many missions as possible whilst they are deceiving the resistance players, so they will have to find the right balance between good and evil behaviour in order to win.

The genetic algorithm could help to optimise a strategy. However, we must consider the costs and benefits of implementing the genetic algorithm. Since we're already using Bayes' theorem to solve the problem, we would only need to optimise the thresholds for how probabilities are used to make decisions. It would take some time to implement a genetic algorithm, so perhaps the time would be better spent manually optimising the strategy.

Implementation

Firstly, we need a set of all the possible worlds. Each of these worlds is represented by a string containing the names of the spies in that world. Next, we need to map each world to a probability value, which gives the probability of each world existing, $P(w)$. Initially all the worlds will have equal probabilities, given by:

$$P(w) = \frac{1}{number\ of\ worlds}$$

This probability value will be modified by in-game actions. Whenever the probability of a world reaches 0, that world has become impossible, so it can be removed from the set of all worlds.

When dealing with an action a, we need to estimate the probability of that action occurring in each of the possible worlds, which gives $P(a \mid w)$. To get the overall probability of the action occurring, $P(a)$, we need to add up all the probabilities of the action occurring in each world, weighting each by the probability of its world existing:

$$P(a) = \sum_i P(a \mid w_i)\, P(w_i)$$

When an action a occurs, we need to update all the world probabilities to include this new information. To do this, we need to calculate $P(w \mid a)$, the probability of each world existing given that action. We can calculate this using Bayes' theorem:

$$P(w \mid a) = \frac{P(a \mid w)\, P(w)}{P(a)}$$

Since we know that the action a has occurred, we can update all of the $P(w)$ values with the value of $P(w \mid a)$ for that world. We can do this update step with any type of in-game action, including nominations, votes, mission results, and accusations.

The accuracy of our probability values depends on how we calculate $P(a \mid w)$ for each type of action. It's relatively straightforward to estimate $P(a \mid w)$ for mission results, because we only need to consider the spies on the team, and spies have perfect knowledge. It's more difficult to estimate $P(a \mid w)$ for nominations, votes, and mission results, since we're considering resistance players and spies at the same time. In these cases, I decided to use the assumption that all players have perfect knowledge. This is not a comprehensive model of behaviour, but it is simple to calculate and it doesn't contradict any knowledge that a player would have in that world. A spy will make a decision using perfect knowledge. Any actions that against the optimum spy strategy will be identified as more resistance-like. As such, it allows us to contrast between spy-like behaviour and resistance-like behaviour.

When making decisions, we need to consider each possible action we can make, and work out the probability of that action being a good idea. For nominations, we need to generate all the possible teams, and work out $P(a)$ for the action of that team succeeding, considering all of the possible worlds. This is weighted by the probability of each world existing, so the most likely worlds will have more of an effect on the value of $P(a)$. Using this value, a resistance player can rank each possible team, and nominate the team that is most likely to succeed.

A spy player needs to consider two different probability values. As with a resistance player, they need to consider the probability of a mission succeeding from a resistance player's point of view. They also need to consider the probability of a mission failing given their knowledge of who the spies are. To minimise the amount of information that the resistance players get from the mission result, they should aim for a perfect number of fails. That is, if the mission requires two fails, they should maximise the chance of exactly two spies playing fail cards. By multiplying these two probability values together, the team with the highest ranking will be a team that looks good to resistance players and has a high chance of failing.

We can make voting decisions by comparing the probability of success for the nominated team with the average probability of success for all the possible teams. If the ratio is above a certain threshold, a resistance player should approve that team. If the ratio is below a certain threshold, a resistance player should reject that team. If a spy is being cautious, they should also vote in this way. In between these two thresholds, a spy player should vote in a way

that is best for achieving a spy victory; if a mission has enough spies to fail, then that team should be approved, otherwise the team should be rejected.

In between these two thresholds, it is uncertain whether a resistance player should vote to reject or approve that team. In this case, the player should consider how trustworthy the current leader is, and compare that with future leaders. This trustworthiness measure can be calculated by adding up all the probabilities of worlds, $P(w)$, where a given player is a member of the resistance. If there is a future leader who is more trustworthy than the current leader, then a resistance player should reject past the current leader.

Accusations can be used as a measure of how suspicious the other players think you are. To decide whether to make an accusation, find the probability of each player being a spy. If a player's probability is beyond a certain threshold, then accusing that player is a good idea. Both resistance players and spies should use the same strategy in this case. Unlike voting and nominations, accusations don't affect other in-game actions, so this is a good opportunity for spies to pretend to be resistance players.

When there are multiple spies on a mission team, and it is a good idea to fail that mission, each spy needs to decide whether or not to play the fail card. It is best if there are just enough fail cards to fail the mission; if there aren't enough fail cards, then the mission won't fail and the resistance players will be closer to victory; if there are too many fail cards, the resistance players get more information about who the spies are. If all spies use a random number generator decide whether or not to play a fail, we can maximise the chance of the correct number of fails by optimising the threshold between playing a success or failure. I found the optimum threshold to be:

$$threshold = \frac{number\ of\ fails}{number\ of\ spies}$$

A spy will generate a random number between 0 and 1. If the number is less than the threshold, then they should play the fail card. Otherwise, they should play the success card.

Validation

Because the agent code uses so many probability values, one of the biggest risks is floating point precision. To minimise this risk, the code uses the double datatype to hold probability values. Doubles can hold more data than floats, which allows for more floating point precision. Another alternative is to store probability values as fractions, with long ints holding the top and bottom parts of the fraction. This would ensure that the probabilities are stored with complete precision. However, testing showed that the numbers at the top and bottom parts of the fraction would frequently overflow the upper limit of the long datatype. While using doubles is not completely precise, there wasn't any noticeable reduction in performance, so the double datatype seems adequate for this scenario.

To test the performance of my agent code, I ran a tournament between the RandomAgent, my agent code (BayesFTW), and a version of my agent code with some of the features removed (BayesMinimal). BayesMinimal is the same as BayesFTW, except the world

probabilities are only updated by mission results, and not by nominations, votes, or accusations. This tournament included 100 000 games of *The Resistance*.

Name	Spy Wins	Spy Plays	Res Wins	Res Plays	Win Rate	Spy Win Rate	Res Win Rate
RandomAgent	51786	94322	35368	156088	0.3480452059	0.5490341596	0.2265901286
BayesMinimal	72033	94487	46816	155589	0.4752515235	0.7623588430	0.3008953075
BayesFTW	72637	94514	62843	155257	0.5424168538	0.7685316461	0.4047675789

Both of the Bayesian agents performed significantly better than the RandomAgent, for both spies and resistance players. This shows that using strategy does improve the chances of an agent's success. For spies, BayesFTW performed slightly better than BayesMinimal, but the difference was not significant. For resistance players, BayesFTW performed significantly better than BayesMinimal. Overall, BayesFTW was clearly the best performing agent. This shows that the extra features of BayesFTW made a significant improvement to its performance, especially for resistance players.

References

Cadwallader Olsker, T. D. (2011). when 95% accurate isn't: Exploring Bayes's Theorem. *The Mathematics Teacher, 104*(6), 426-431. Retrieved from http://www.jstor.org.ezproxy.library.uwa.edu.au/stable/20876909

Gill, R. D. (2011). The Monty Hall problem is not a probability puzzle. *Statistica Neerlandica, 65*(1), 55-71. http://dx.doi.org/10.1111/j.1467-9574.2010.00474.x

Lin, C., & Ting, C. (2011). *Emergent Tactical Formation Using Genetic Algorithm in Real-Time Strategy Games.* Paper presented at the 2011 International Conference on Technologies and Applications of Artificial Intelligence. http://dx.doi.org/10.1109/TAAI.2011.63

Oaksford, M, & Chater, N. (2009). The uncertain reasoner: Bayes, logic, and rationality. *Behavioral and Brain Sciences, 32*(1). 105-120. Retrieved from https://doi-org.ezproxy.library.uwa.edu.au/10.1017/S0140525X0900051X

van Ditmarsch, H., & Kooi, B. (2015). *One Hundred Prisoners and a Light Bulb*. Switzerland: Springer International Publishing.