Aarya Bhorra 23277271

Project Report

Problem Definition

The primary challenge in the game is identifying who the spies are and who the loyal players are.

State: The state of the game can be described by mission history, the players in the games and beliefs of the agent.

Actions: Agent would be able to do three actions propose a mission, vote on a mission and betray a mission if they're a spy

Outcomes: Based on the agent's actions a mission team can be rejected or accepted, and mission can pass or fail.

Goals: For an agent who is part of the resistance the goal is to maximize the number of passed mission and minimize the number of failed mission, and if agent is spy the goal is to do the opposite.

Methodologies

MiniMax Agent

While MiniMax Algorithm is typically used for Zero Sum games, it can be employed here to generate a game state as a decision tree, where nodes reflect mission proposals, voting outcomes, and mission success or failure. Minimax can be used to choose actions that maximize its team's chances of success, whether it is resistance or spies. For resistance, it minimizes the chance of spies infiltrating missions, while for spies, it maximizes the likelihood of sabotaging missions without revealing their identity. However since this is not a zero-sum game with only two players, this method would result in very high space and time complexity.

Reinforcement Q-Learning Agent

A Reinforcement Learning (RL) agent could be applied to play The Resistance. The agent would track the rewards of its actions after each round.

For the action of voting "Yes" for a proposed team, the Q-values would be calculated as follows: If the agent votes for a team that fails, it decreases the Q-value for voting "Yes" on future teams containing the same players. The Q-value for proposing a team with those players also decreases. Conversely, if the mission is successful, the Q-values increase.

If Agent was a spy would also keep track of when mission fails. It would have an internal state of the team size, number of betrayals that caused the mission fail and number of spies in team. It would then reward mission with a certain ratio of spies to team member to vote to betray.

```
Q-Values:
------------------------------------------------
Player: 1, Action: True, Q-Value: -0.0018910000000000177
Player: 0, Action: True, Q-Value: -0.1009
Player: 2, Action: True, Q-Value: -0.09009999999999999
Player: 4, Action: True, Q-Value: -0.1
Player: 3, Action: True, Q-Value: -0.19
------------------------------------------------
```

Hidden Markov Model Agent

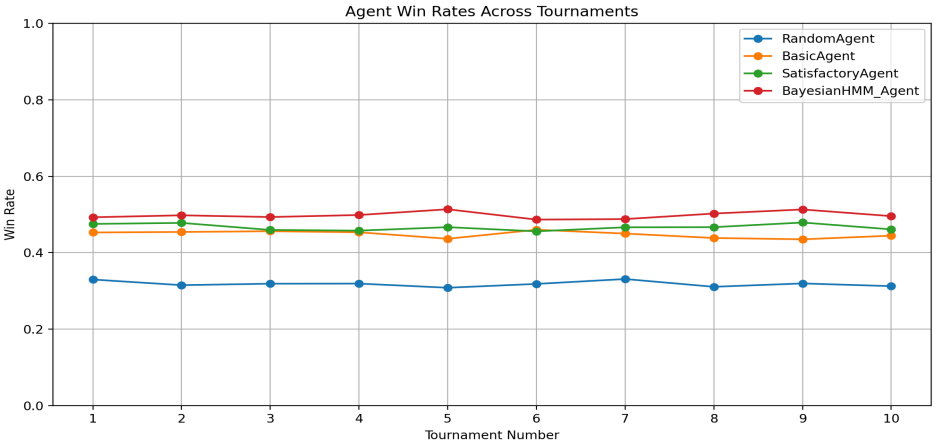
A Hidden Markov Model (HMM) Agent is well-suited for identifying spies in the game, leveraging probabilistic reasoning to make informed decisions based on player behaviour. The model operates by estimating the likelihood that a player is a spy, utilizing observations from the game's dynamics.

Hidden Variable: The core hidden variable in this model is whether a player is a spy or not.

Observations: The agent considers three key observations to update its beliefs about players: the player is part of a failed team, the player voted for a failed team, the player proposed a failed team.

Initial Belief: The agent begins with an initial belief that any player is a spy, set to 1/3
Belief Updating: The agent updates its beliefs based on mission outcomes and voting behaviours: Trust increases for players who are part of successful missions and Trust decreases for players involved in failed missions or who voted for a failing team.
Voting: The agent votes for or against proposed teams based on its trust levels for the players involved.
Team Proposals: When proposing a team, the agent prioritizes players with the highest trust levels.If the agent is a spy, it will only betray when the number of spies on the team matches the required number of betrayals to ensure the mission's failure.
Performance
This model beats the reference agents reliably across tournaments. HMM agent has the highest win-rate across 10 tournaments

```
Resistance Wins: 213, Spy Wins: 787, Resistance Win Rate: 0.2130
 1: BayesianHMM_Agent | win_rate=0.4912 res_win_rate=0.2513 spy_win_rate=0.8804
es=873 spy_wins=633 spy_losses=86
 2: SatisfactoryAgent | win_rate=0.4507 res_win_rate=0.2183 spy_win_rate=0.8268
ses=906 spy_wins=592 spy_losses=124
 3: BasicAgent        | win_rate=0.4402 res_win_rate=0.2085 spy_win_rate=0.8268
ses=915 spy_wins=573 spy_losses=120
 4: RandomAgent       | win_rate=0.3253 res_win_rate=0.1486 spy_win_rate=0.6317
ses=1014 spy_wins=434 spy_losses=253
```



Comparison of Different Techniques

Merits Reinforcement Learning Agent

The agent could perform well in generalized situations since it considers only a few key variables, keeping the model simple. Random variables such as the type of agent it's up against or the number of players in the game would not affect the agent.

Merits HMM Agent

This approach allows the HMM agent to create a reliable prediction model for identifying spies over a span of five rounds, considering essential variables and adapting its strategies based on observed behaviour.

Comparison

With only 5 rounds, assigning appropriate Q-values to actions becomes challenging, as it is difficult gain meaningful data in such a brief time. The exploration rate of the agent must be set to a very low value.
Additionally, since it considers only one factor when setting the team (if voted for a member part of a failed team), some resistance players might repeatedly participate in failed missions by misfortune, leading to erroneously low Q-values. Reinforcement Learning also does not account for the voting history of other players, which is crucial for determining who the spies

are and deciding on actions. Since, the agent's memory does not persist across games in a tournament; new agents are created for each game, preventing learning from previous experiences. As a result, any form of learning in this context is highly limited.
As shown in the Screenshot below it does not perform better than BasicAgent.

```
Resistance Wins: 368, Spy Wins: 632, Resistance Win Rate: 0.3680
 1: SatisfactoryAgent | win_rate=0.5500 res_win_rate=0.4083 spy_win_rate=0.7762
 2: BasicAgent        | win_rate=0.5219 res_win_rate=0.3928 spy_win_rate=0.7284
 3: QLearningAgent    | win_rate=0.4159 res_win_rate=0.3407 spy_win_rate=0.5511
 4: RandomAgent       | win_rate=0.3820 res_win_rate=0.3043 spy_win_rate=0.5092
```

Bayesian Agent outperforms reinforcement agent by a large margin because it is able to update it's internal state mapping players to the probability of being spy much better in 5 rounds than the RL agent.

```
LEADERBOARD AFTER 1000 GAMES
Resistance Wins: 258, Spy Wins: 742, Resistance Win Rate: 0.2580
 1: BayesianHMM_Agent | win_rate=0.5209 res_win_rate=0.3184 spy_win_rate=0.8523 |
 2: QLearningAgent    | win_rate=0.3620 res_win_rate=0.1908 spy_win_rate=0.6465 |
```

Since the Bayesian agent considers most key factors in determining who is a spy or not, it is able maintain a fairly accurate internal mapping of players to probability of being a spy.
For most games the agent can at least accurately guess one spy.

```
Spy Probabilities:
------------------------------------------------
Player 0: 0.30
Player 1: 0.21
Player 2: 0.30
Player 3: 0.42
Player 4 (You): 0.00
------------------------------------------------
```

The spy win rates for the Bayesian Agent are also fairly stable as it only betrays when the number of betrayals to fail the mission can be met and if the team size is not equal to the number of betrayals to avoid detection.
RL learning agent is using a decaying exploration rate to vote for betrayals. Given the current way of updating, rewarding ratio of number of spies in team to team size to betray, 5 rounds is not enough to come up with a successfully policy.
The Bayesian models' win rates are stable across the number of players, however the Reinforcement agent's win rates falter as the model does not cope with complexity well. With 10 players the RL agent does not maintain Q Values would lead to a meaningful strategy.

Conclusion

The Bayesian model is ideal for The Resistance because it efficiently updates beliefs based on mission outcomes and player behaviour, even in limited rounds. It accounts for key factors like voting patterns and team membership, allowing it to accurately estimate spy probabilities. This adaptability makes it superior to reinforcement learning and minimax strategey, which struggles with limited data and exploration in short games. The Bayesian model's ability to handle uncertainty and adjust trust in real-time gives it a strategic edge, leading to better performance in identifying spies.