The Resistance Agent Implementation Analysis (Mahit Gupta - 23690265)

This report outlines the techniques researched/implemented and failed for the Intelligent Agents Project.

Techniques Investigated

1. Simple Heuristic Probability Analysis

This approach involves using basic probability theory and heuristics to make decisions based on observed game events. This was my first attempt for the agent. Although basic, this is definitely a working solution and something I based my final agent on. This agent was capable to beat the `basic` and `random` agent almost certainly but would only beat the `satisfactory` agent about 50% of the time

Here is the implementation of my update_spy_probs which tracks all the players and possibility of them being a spy depending on the mission result(s):

https://github.com/MahitGtg/The-Resistance-Agent/blob/main/Simple Heuristic Probability Analysis

This worked for quick decisions and was easy to code. For example, if a mission failed, I could quickly increase the spy probability for those team members. However, this method had some some problems. It struggled with complex game situations, like when a spy deliberately passed a mission to seem innocent. Our agent also had trouble planning for the whole game, often focusing too much on the current round. This made it vulnerable to long-term strategies by other players. For instance, if a spy consistently voted against spy-heavy teams to appear trustworthy, our agent might wrongly trust them.

2. Monte Carlo Tree Search (MCTS) with Depth Cutoff

Monte Carlo Tree Search is a heuristic search algorithm that builds a decision tree through random sampling. I attempted a simplified version for mission proposals and betrayal decisions, but full implementation proved challenging within the project constraints.

https://github.com/MahitGtg/The-Resistance-Agent/blob/main/mcts_code

Potential Benefits and Limitations

- **Benefits**: Adaptability to game states, balanced exploration/exploitation, handling uncertainty.
- **Limitations**: Time constraints, simulation accuracy challenges, difficult state evaluation.

I was initially planned to have (MCTS) for my Resistance agent. It seemed perfect for handling the game's hidden information and balancing different strategies. Holver, implementing it was tough. The 1-second time limit and the complexity of simulating social deduction lre big hurdles. My basic MCTS version sometimes lost even to the `basic` agent. While research suggests MCTS could be great for games like The Resistance, I couldn't make it work within our constraints.

3. Bayesian Networks

Bayesian networks are probabilistic graphical models that represent a set of variables and their conditional dependencies. In the context of The Resistance, a Bayesian network could model the relationships betlen player actions, game events, and hidden roles.

Potential Applications:

1. **Role Inference**: Updating beliefs about player roles based on observed actions and mission outcomes.
2. **Action Prediction**: Predicting likely actions of other players based on their past behavior and inferred roles.
3. **Decision Making**: Using the network to inform mission proposals and voting decisions.

While I did not fully implement a Bayesian network, I considered the following structure:

```
Players (Roles) -> Mission Proposals -> Voting Behavior -> Mission Outcomes
       |                  ^                  ^                 |
       |                  |                  |                 |
       +------------------+------------------+-----------------+
```

Reasons for Not Pursuing

1. **Complexity**: Designing an accurate model that captures all relevant dependencies is extremely complex.
2. **Computational Intensity**: Updating and querying the network within the 1-second time limit would be challenging.
3. **Data Sparsity**: Limited observations in a single game make it difficult to train an accurate model.

This was a very interesting approach, I didn't spend a lot of time researching about this but Bayesian theory is quite vast and seemed much complex for a project like this, although intriguing; this is something I would research about in my own time and try it's basic implementations, for this project I couldn't write a working code for this and hence it was a failed attempt

Our Chosen Approach: Enhanced Probabilistic Reasoning with Q-Learning(Reinforcement Learning)

After exploring various methods, we decided to combine enhanced probabilistic reasoning with Q-Learning for our final agent implementation. This approach leverages both probabilistic modeling and reinforcement learning, resulting in an agent that is adaptive and efficient.

Key Components of Our Implementation

1. **Dynamic Spy Probability Updates**: We continuously update our beliefs about each player's likelihood of being a spy. This mechanism considers mission outcomes and voting patterns, refining our understanding of player roles as the game progresses.
2. **Trust Score Calculation**: We developed a trust score system for each player that evolves based on their actions and mission outcomes. This adds nuance to our decision-making process : https://github.com/MahitGtg/The-Resistance-Agent/blob/main/trust_score_q_learning
3. **Q-Learning for Action Selection**: We incorporated Q-Learning to guide our decision-making process, particularly for proposing missions and deciding whether to betray (when playing as a spy). This allows our agent to learn and improve its strategy over time.

   Q-Learning Explained

   - **Q-Values**: Numerical values representing the expected utility of taking certain actions in specific states.
   - **Epsilon-Greedy Strategy**: Balances exploration and exploitation by choosing random actions with probability ε and the best-known actions otherwise.
   - **Learning from Rewards**: After each action, the agent updates its Q-value based on the received reward and anticipated future rewards, using the formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$$

Where:

- s is the current state.
- a is the action taken.
- r is the reward received after taking action a in state s.
- s' is the new state after action.
- a' is any possible action from the new state s'.
- α is the learning rate ($0 < \alpha \leq 1$).
- γ is the discount factor ($0 \leq \gamma < 1$).

4. **Incorporation of Expert Player Strategies**: To make our agent's decisions more human-like, we integrated strategies commonly used by experienced players, such as:
   - Scrutinizing leader behavior, especially when they don't include themselves in missions.
   - Analyzing voting patterns to identify potential spy collaborations.
   - Balancing information gathering with mission success when proposing teams.
   - Coordinating spy actions to avoid multiple betrayals on a single mission.

Advantages of Our Approach

1. **Balance of Sophistication and Efficiency**: Our method enables complex decision-making while staying within the 1-second time limit, which was a challenge with methods like MCTS.
2. **Adaptability**: By combining probability updates with Q-Learning, our agent adapts to various game states and opponent behaviors—a crucial feature in a game where strategies vary widely.
3. **Handling Uncertainty**: The probabilistic foundation effectively manages the hidden information aspect of the game, addressing a key challenge in The Resistance.
4. **Long-Term Strategy**: Q-Learning encourages our agent to consider the long-term implications of its actions, leading to more sophisticated game strategies.

Performance Results

In sample tests, our agent achieved the following win rates: **Against RandomAgent**: 78%, **Against BasicAgent**: 65% and **Against SatisfactoryAgent**: 53%

Comparison with Alternative Approaches

1. **Simple Heuristic Probability Analysis**: While this method was the foundation of our approach, our enhanced version with Q-Learning:
   - Adapts its strategy over time through reinforcement learning.
   - Considers long-term consequences of actions.
   - Handles complex game scenarios more effectively.
2. **Monte Carlo Tree Search (MCTS)**: Although promising in theory, our approach proved more practical:
   - Operates efficiently within the 1-second time limit.
   - Requires less complex simulation of game states.
   - Adapts better to the partially observable nature of The Resistance.
3. **Bayesian Networks**: Our method offers similar benefits with less complexity:
   - More computationally efficient within game constraints.
   - Easier to implement and tune for the game's dynamics.
   - Combines probabilistic reasoning with learning capabilities, which pure Bayesian approaches lack.

Our enhanced probabilistic reasoning with Q-Learning effectively balances the strengths of these approaches while mitigating their weaknesses. It provides the adaptability of MCTS, the uncertainty handling of Bayesian networks, and builds upon the simplicity of heuristic probability analysis. The result is an agent that performs well across various game scenarios and opponent types, as evidenced by its superior performance against reference agents

References

1. https://www.reddit.com/r/boardgames/comments/1p8601/what_are_your_strategies_for_playing_the/ (Popular game strategies for the resistance)
2. https://boardgames.stackexchange.com/questions/4719/optimal-or-just-effective-strategy-for-the-resistance (Popular game strategies for the resistance)
3. https://www.youtube.com/watch?v=nIglv4IfJ6s (Reinforcement Learning - Exploitation vs Exploration)
4. https://www.simplilearn.com/tutorials/machine-learning-tutorial/what-is-q-learning - (Q-Learning/Image)
5. https://teaching.csse.uwa.edu.au/units/CITS3001/project/2017/paper1.pdf- (Research for MCTS, Ch - 2.4)