**Design Defense: Pirate Intelligent Agent in a Treasure Hunt Game**

Human vs. Machine Problem-Solving Approaches

If a human were to solve the treasure hunt maze, they would likely approach the problem using logical reasoning and trial-and-error. They might visually scan the maze, identify open paths, mentally simulate routes, and backtrack when they hit dead ends. This approach is highly flexible and benefits from intuition and pattern recognition built through experience. However, it is also prone to inconsistency and fatigue.

In contrast, the intelligent agent I developed follows a structured process using deep Q-learning. The agent does not "see" the maze as a whole but interacts with the environment through states and rewards. It begins with little knowledge and gradually learns optimal actions through repeated episodes and feedback from the environment. While the human can adapt quickly, the agent needs many iterations but excels at consistency and scalability once trained.

The similarity between both approaches lies in the concept of learning from experience. The difference is that the machine learns purely from numeric feedback, whereas humans use abstract reasoning and foresight. The machine relies on an epsilon-greedy policy to balance exploration and exploitation, while a human might explore instinctively.

Purpose of the Intelligent Agent in Pathfinding

The pirate intelligent agent's purpose is to find the treasure before the human player does. This requires it to learn how to navigate the maze efficiently. One of the most important dynamics in reinforcement learning is balancing exploration and exploitation. Exploration involves trying new paths to discover better solutions, while exploitation uses known successful strategies to reach the goal. In this project, we set the exploration factor (epsilon) to 0.1, which means 10% of the time, the agent chooses a random action to explore.

In a pathfinding scenario like this one, an ideal balance might start with high exploration and decay to lower exploration as the model becomes more confident in its learned strategies. The agent in this project adapted well using this logic, and over time, its win rate improved as it relied more on exploitation.

Reinforcement learning helps the pirate agent determine the path to the goal by rewarding it for reaching the treasure and penalizing it for hitting walls or failing. Through thousands of episodes stored in experience replay, the model generalizes which actions are most likely to lead to success (Welcome to Spinning Up in Deep RL! - Spinning Up Documentation, n.d.).

Evaluating the Use of Deep Q-Learning

To implement deep Q-learning, I used a neural network with three layers. The input layer received the environment state, and the output layer returned Q-values for each possible action (up, down, left, right). The agent used these Q-values to make decisions during training.

The training process involved storing past experiences (state, action, reward, next state, done) and sampling batches from memory to train the model using mean squared error loss. The model trained over thousands of epochs, adjusting weights using the Adam optimizer. As training progressed, the win rate increased and the agent demonstrated effective navigation strategies.

While deep Q-learning requires more time to train than rule-based systems or traditional algorithms like A\*, it provides flexibility and adaptability. Once trained, the model could generalize to different starting points within the maze, making it a powerful solution for pathfinding tasks where the environment may vary (GeeksforGeeks, 2025b).

**References**

*Welcome to spinning up in deep rl!¶* Welcome to Spinning Up in Deep RL! - Spinning Up documentation. (n.d.). https://spinningup.openai.com/en/latest/

GeeksforGeeks. (2025b, March 19). *Deep Q-learning in reinforcement learning*. https://www.geeksforgeeks.org/deep-q-learning/