A human would likely utilize a trial-and-error methodology when solving a maze. In doing so, they would initially, and at each step, consider what available options they could take, such as moving straight, turning, or moving backwards. This approach allows a human to explore the environment, which is described as “subjects tended towards breadth planning, which suggests that they may be more likely to adopt a strategy that allows them to explore multiple options before committing to a specific solution” (Kadner et al., 2023). These options would then be evaluated against the progress towards the goal, such as understanding that moving backwards may be inefficient when other, unexplored options are available. This allows a human to evaluate how beneficial each move is in the scheme of finishing the maze. During this period of exploration, it is likely that humans will encounter dead ends where they will have to backtrack to a position with new options. Subsequently, they can iteratively recognize and remember where turns with poor end states exist to avoid them in future attempts. After perhaps many runs, the human will refine the strategy of sequencing moves based on learned experience, allowing them to complete the maze with increasing efficiency.

Alternatively, the intelligent agent implemented takes similar steps and uses a similar methodology for solving this pathfinding problem. Initially, the agent has no knowledge of the environment. Therefore, it must, at least initially, explore available actions in the given state at random (Gulli & Pal, 2017). While many of these actions are guaranteed to be poor choices, this allows the agent to understand poor actions based on the penalties. With each move, the agent will build a repository in memory that contains the action taken at a specific state and what the resulting reward or penalty was. A fail-safe mechanism can be implemented through a minimal penalty on each move to prevent the agent from repeating moves in an infinite loop. After many random-choice actions, the agent will eventually finish the maze. In future iterations, the agent will continue exploring an increasing number of actions in any given state to build and refine this matrix in memory. However, the agent will continuously rely less on random exploration and begin exploiting the matrix more by choosing moves that have been proven to be the most beneficial at each state (Gulli & Pal, 2017). Over time, the agent will be able to make precise decisions based on previous experience, allowing it to complete the maze with increasing efficiency.

These two approaches share commonalities and differences in the methodology utilized by the participants. First, both the human and agent will initially have no experience in the maze environment and will subsequently have to rely on random exploration to make progress. However, the human will already understand certain limitations that the agent may not have, such as not being able to move through walls. As each participant continues to move through the maze, they will accumulate knowledge that can be used in future iterations to increase efficiency. However, exactly how this knowledge is stored is different. For a human, this knowledge is strictly memory-based by remembering a sequence or pattern, while the agent utilizes reward or penalty assignments for each action in any given state. Once sufficient knowledge of the environment is obtained, the participants can begin to fulfill the task more efficiently by leveraging past experiences gained from exploration. However, the human will likely rely on following a memorized pattern of moves, while the agent will perform the action with the greatest rewards.

As previously mentioned, the intelligent agent will utilize exploration and exploitation to solve the pathfinding problem. Exploration is the process of randomly performing actions based on the current state. This process is highly important to explore the state space, allowing the agent to refine the matrix with rewards and penalties for each move (Gulli & Pal, 2017). Alternatively, exploitation is the process of utilizing stored knowledge to make an informed decision regarding the best possible action. The key benefit of leveraging both tactics is described as “the first possible path that might be utilized to reach a solution is not guaranteed to be the best path. With this being stated, it is unlikely that it will always be the case that if we keep searching, we will find a better solution than the current one, and therefore we abstain from solving the problem” (Beysolow, 2019). For the agent’s purposes, the epsilon is the parameter that controls whether the agent will perform exploration or exploitation for any action. In this pathfinding problem, the epsilon is set at 0.1, which means the agent will exploit previously gained knowledge nine out of ten times and will explore one out of ten times. This allows the agent to primarily focus on utilizing previous game knowledge and experience to accomplish the goal while occasionally attempting to gain new knowledge, which may lead to better pathing routes.

To accomplish the end goal, the agent utilizes reinforcement learning to determine the optimal path to take. Reinforcement learning encompasses how an agent learns to behave in an environment by performing actions and observing associated rewards (Lamba, 2018). Every action should have an accompanying reward or penalty, which will teach the agent how to discern between a beneficial and detrimental action. This information is then stored in a matrix for future use, which is described as “as we start to explore the environment, the Q-function gives us better and better approximations by continuously updating the Q-values in the table” (Lamba, 2018). As the Q-values become more precise, the agent can rely less on random exploration and instead perform the action with the highest reward based on future discounted reward predictions. Once enough training has been done to refine the matrix Q-values, the agent can determine the optimal path to the goal by strictly following the policy.

With this information, the complex pathfinder problem was solved with deep Q-learning using a neural network. Q-learning is described as “a model-free reinforcement learning algorithm. The goal of Q-learning is to learn a policy that tells an agent what action to take under what circumstances” (Düğmeci, 2021). First, it was necessary to outline the rewards and penalties system to allow the agent to continually obtain the optimal policy. Each move had a minimal penalty to reinforce the need to complete the maze efficiently. Attempting to move into an occupied square or exit the game boundary resulted in a large penalty to enforce game boundaries. However, the agent received a large reward for reaching the treasure at the end of the maze, which served as the primary motivator. Next, the epsilon parameter had to be implemented for the epsilon-greedy principle to force the agent to either explore the environment to possibly identify alternative routes or exploit previously gained knowledge by performing the action with the highest expected reward. Empirical data for each move was also stored in replay memory, which was utilized in a random order to train the model by making a prediction for the best possible action. Finally, it was necessary to implement a win condition when the agent had been adequately trained to perpetually accomplish the goal. This win condition was also instrumental in displaying the agent’s progress throughout training.

Once a game iteration had begun, the agent would first observe the valid actions available. The agent would perform an action, either by exploration or exploitation, depending on the epsilon, and store the state, action, and rewards for future reference. On each move, the program would verify the game status to determine if the goal had been achieved or not.

Furthermore, the empirical data was then utilized to train the neural network by adjusting the Q-values for each action-state pair. Therefore, future exploitation moves would be more accurate relating to the highest future discounted rewards. Once the agent accomplished the goal, the program would check the win rate to determine if the agent had been sufficiently trained. If not, the next game iteration would commence, and the process would repeat. Subsequently, the Q-values were modified to be increasingly precise, while the agent was allowed further opportunities to explore the environment for potentially more beneficial routes. Therefore, by the end of training, the agent was able to perform actions that had the highest associated reward, allowing the game to be completed effectively and efficiently.

References

Beysolow, I. T. (2019). Applied reinforcement learning with python : With openai gym, tensorflow, and keras. Apress L. P.

Düğmeci, D. (2021, November 21). Finding Shortest Path using Q-Learning Algorithm. Medium. https://towardsdatascience.com/finding-shortest-path-using-q-learning-algorithm-1c1f39e89505

Gulli, A., & Pal, S. (2017). Deep learning with keras : Get to grips with the basics of keras to implement fast and efficient deep-learning models. Packt Publishing, Limited.

Kadner, F., Willkomm, H., Ibs, I., & Rothkopf, C. (2023). Finding your Way Out: Planning Strategies in Human Maze-Solving Behavior. Proceedings of the Annual Meeting of the Cognitive Science Society, 45, 1–7. https://escholarship.org/uc/item/94t1t8kw

Lamba, A. (2018, August 27). A brief introduction to reinforcement learning. Medium. https://medium.com/free-code-camp/a-brief-introduction-to-reinforcement-learning-7799af5840db