Reinforcement Learning Algorithms with Changing State Spaces in BurlapCraft

Yoshiaki Kuida

Reinforcement learning algorithms are commonly used in static or turn-based environments where the algorithm and environment both update at the same pace. However, there are many scenarios where the environment is not static and contains many agents moving independently. BurlapCraft is a suitable platform for testing reinforcement learning algorithms in time-sensitive environments because of its discretizable state space and its multi-agent functionality.

BurlapCraft is a Minecraft module developed by the Humans To Robots Laboratory that uses Burlap, the Brown-UMBC Reinforcement Learning and Planning Library, and Minecraft Forge to control an agent that performs learning and planning tasks. My project modifies the BurlapCraft source code such that the Minecraft agent learns to fight different mobs using different reinforcement learning algorithms. I tested three algorithms (Gradient-descent Sarsa(λ) with Tile Coding value approximation, and Potential Shaped RMax, and Q-learning) on three types of mobs (zombies, witches, and skeletons).

My results indicate that gradient-descent Sarsa(λ) is quick to learn, but since it does not consider previous iterations as much as the other algorithms, it works best for fighting zombies, since the optimal fighting policy is a very simple sequence of actions. Gradient-descent Sarsa(λ) performs the worst for learning to fight witches and skeletons, since their respective optimal fighting policies are more complicated sequences of actions that necessitate multiple iterations to learn. The success of the Potential Shaped RMax learning algorithm was heavily influenced by the fit of the potential function to the mob. Although the use of a good potential function encouraged faster learning, a bad potential function resulted in essentially no learning at all. Lastly, while Q-learning has a distinct positive learning curve for witches, it has a much flatter curve for zombies and only observes a slight improvement. This is because Q-learning tends to perform better with smaller state spaces. Against zombies, Q-learning performs slightly better than the random policy algorithm but would most likely need more than 100 iterations per trial to learn a better policy. The fact that zombies move much more than witches exacerbates this issue. Value approximation may be a beneficial addition to the Q-learning algorithm. This would generalize the information gained through iterations, leading to faster learning.