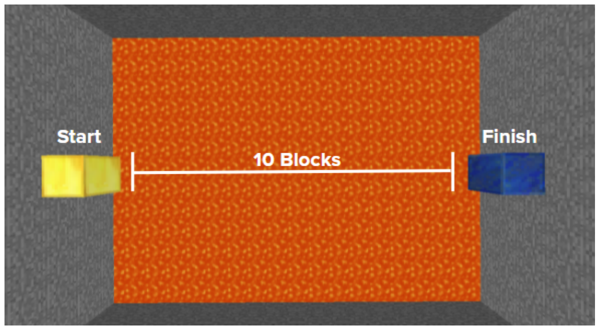
Continuous Reinforcement Learning through Discretization: Bridging in Minecraft with AI

Denis Callinan and Zach Yahn

Problem Summary

Minecraft is a rich open world environment that lends itself well to artificial intelligence problems. In order to narrow down the broad possibilities present in the base video game, this project focused on a smaller subproblem: bridging across a gap without falling into a pit of lava. The gap is ten "blocks" (in-game units) wide, with a single start and end block at either side. See **Figure 1** below for an image of the setup.



Bridging is a necessary skill in Minecraft, as it is the only way to cross certain geographical features like canyons and ravines.
Not to mention, bridging uses all of the basic actions of Minecraft: placing blocks, crouching, walking, and looking around. There is also more than one way to bridge: while the player can guarantee their safety by continuously crouching, this comes at the cost of time. In Minecraft, crouching yields a much slower movement speed than walking, so crouching the entire time results in significantly slower bridging. The fastest bridging techniques involve a rapid sequence of crouching and uncrouching, with the difference being made by how precisely the player can sequence these moves. Most players cannot keep this up for more than a few blocks, and being able to "speed bridge" across a gap of 10 blocks is considered an impressive achievement.

The goal of this project was to train an agent that could perform at human or near-human bridging abilities. It takes a basic human player less than a few hours to learn how to "safe bridge" by crouching, but only the best players learn how to speed bridge at all - and then only after many hours of practice. Training an agent to speed bridge would be a significant accomplishment, and thus is the stretch goal for this project.

Approach

Although Minecraft is represented as a three-dimensional grid world to the user, the functional representation of the environment is actually continuous; movement and viewport adjustment commands are all performed under the context of a continuous state-action space. To connect Minecraft with AI implementations, the Microsoft-developed Malmo API was utilized. This API provides both a discrete and a continuous action set for agents to use. However, the discrete action set is unable to capture the precise behavior needed to solve the bridging problem. The discrete action set utilizes Minecraft teleportation mechanics, which counteracts the use of the crouch action that prevents the agent from falling from the bridge being constructed. Thus, solving this problem would require the use of this continuous action set, which utilized acceleration parameters ranging on the interval [-1,1] and exhibit noise (not 100% accurate). Given the nature of the problem, reinforcement learning was considered to be the best approach due to the need for the agent to learn how to navigate the space. However, the unique problem related to the action set resulted in the discretization of the Minecraft state-action space with the creation of an action handler to translate the continuous commands into a discrete space.

Action Handler

Much of the initial work to address this problem involved the discretization of the state-action space with the development of the aforementioned action handler. The objective of the handler was to intake actions that the reinforcement learning mechanism could interpret as discrete while performing such a discrete action in the continuous state-action space that Minecraft provides. This will result in the reduction of a complex continuous problem into a discrete problem solvable using the chosen RL technique. A protocol was developed, similar to the action encoding provided for the continuous action space with the added addition of a relative target parameter. This relative target parameter allows the agent to perform an action from the start state to a state the specified distance away, thereby mimicking a discrete action set. See **Figure 2**, below, for the full translation of the action spaces.

Continuous Action	Move backward	Crouch/uncrouch	Use	Turn	Pitch
Discretized Version	Move backward *X* units	Crouch/uncrouch	Use once	Turn *X* degrees	Pitch *X* degrees

Note: X units could be specified in the code to get more precise movements

This simple reduction approach, however, failed to consider the error attributed to the continuous Malmo actions. With the initial handler, the action would overshoot its target due to the logic within the handler. To minimize this error, a regression model for each action type (move, pitch, turn) was developed that correlated the action's speed to the difference in actual and expected distance. This model was developed under the assumption that the time for actions at any speed to recognize that the target bound was exceeded was constant. Combined with slowing the speed of the agent within a certain range of the expected target, the subtraction of this model from the effective target in the logic resulted in precise actions off by $\leq \pm 0.2$ units in most cases.

Q-Learning

For all of the varieties of reinforcement learning mechanisms, Q-Learning was utilized due to its model-free approach that prevents the need to know transition probabilities ahead of time. Given the flavors of Q-Learning, a tabular approach was favored due to the feasible size of the state-action space for smaller iterations of the problem. A feature-based approach may be better suited for larger state-action spaces once translated with the handler, but this was considered a stretch goal due to the uncertainty of the noise from the handler and the lack of prior utilization of this approach within Malmo.

Given tabular Q-Learning, a reward structure was formulated that incentivized moving closer to the end as fast as possible without falling into the lava. Large positive and negative rewards were granted for reaching the goal or falling into the lava, respectively, and the reward for living was balanced with the reward for nearing the end so that the agent was never incentivized to sit still and passively collect rewards.

Results and Improvements

There were varied results for the different subproblems that were attempted in this project. To some degree everything was accomplished, though there is much room for improvement in some of the stretch goals.

Basic Results

First and foremost, the goal was to build an agent that could complete the bridging problem. With some constraints on the action space the problem became tractable with 150 trials of training. At that point, the agent was able to bridge across all ten blocks consistently without falling into the lava. The agent's speed was comparable to a basic human player. Most human players do not need to achieve great speeds when bridging, so they use a much safer technique that does not involve precise timing. This is what the agent learned in our basic trials when speed was not a major factor: safety was worth being slower. These results on their own are not particularly impressive, because the limited action space made it fairly easy to accomplish using tabular q-learning. The real merit of the results is in how well the stretch goals were accomplished.

Stretch Goal Results

While the agent was able to successfully bridge across the lava 100% of the time after training for 150 rounds, it used a technique that was very slow: stay crouching, walk backwards, and place a block after enough time had passed. This technique would not work for the second goal, which was to build an agent that could bridge across as fast as possible. Once time was added to the reward system, the agent was incentivized to take riskier moves. This led to a faster technique, but also one that was much more prone to falling into the lava. Since most of the reward was not earned until the agent reached the goal, its learning was slower until it was capable of reaching that distance. As a result, it had to rely on rewards for dying, living, and decreasing the distance to the goal. This meant the agent learned slowly, and took many trials before making any progress. Unfortunately, the Minecraft world had to be reloaded after every trial to reset the blocks that were placed, severely slowing down training as well. While this was not a problem for the 150 trials of the basic agent, it became prohibitive for the more complex task. As a result, the agent was never able to successfully "speed bridge" across all ten blocks to reach the goal. However, by the end of 150 trials of training, it was able to bridge for 5 blocks consistently. Notably, it also traversed this distance faster than the average skilled human player (and significantly faster than either of us). While this is not a perfect success for this problem, it demonstrates that given more time and computational resources our agent would have surely been able to complete the problem.

Conclusion

Given the results of the project, success was found in applying reinforcement learning to a continuous state-action space utilizing a discretization technique, particularly where the nature of the environment is largely inadequate for a feature-based on model-driven approach. Although more complicated iterations of the problem were not fully completed in this project, much of the difficulty in addressing this problem came with the creation of the action handler, which allowed for the Q-Learning algorithm to implicitly categorize an infinite number of continuous states into a finite number of discrete states. This reduction in complexity is particularly well-suited for problems where a large number of continuous, adjacent states will execute the same policy upon convergence. It is worth mentioning that a substantial amount of time was spent learning how to work with Project Malmo and implementing Q-Learning in it from scratch without a library.

In future projects, a Deep-Q approach without an action handler could potentially be more optimal, particularly for iterations with a larger number of states or translated actions due to its ability to better approximate large numbers of Q-values. This represents the logical next step for investigating the use of AI to solve the Minecraft bridging problem.