Simulation-based Artificial Intelligence Framework

Z.Liu

1 Problem description

The problem considers a simple task. A robot on a 8×8 grid wishes to push a bomb into a river surrounding the grid. The state of the problem is a two-element tuple, where the first element represents the location of the robot and the second element represents the location of the bomb. At each step, the robot can take the action of moving towards one of the adjacent locations with a cost of 1. By successfully pushing the bomb into the river, the robot will be rewarded by 100. However, the bomb has a timer. If the robot fails the task within 200 steps, the bomb explodes with a cost of 200, resulting in the maximal cost of 400.

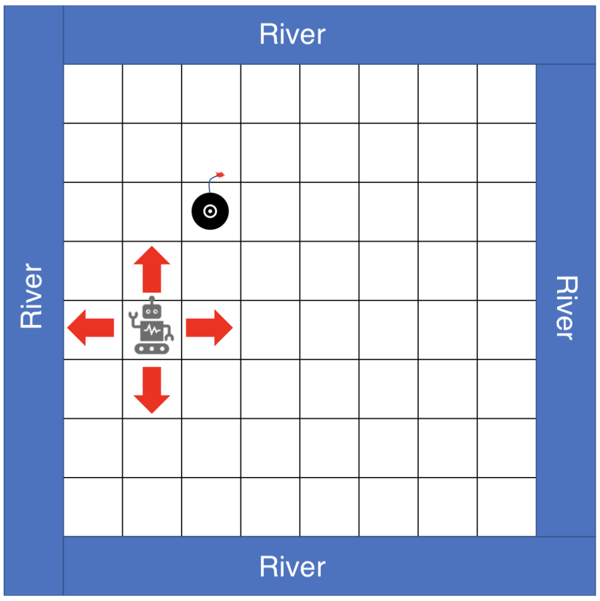

Figure 1: An illustration of the task. A robot wishes to push a bomb into the river.

2 Framework

To solve this problem, we build a simulation-based artificial intelligence (AI) framework, consisting of three parts: an AnyLogic simulation model, a Java Maven project and a Python reinforcement learning (RL) algorithm. The AnyLogic model is integrated into a Java project, where a Python interface is developed in Java to transfer outputs to the Python program. In the Python program, we use a Monte Carlo reinforcement learning algorithm to learn the optimal actions from the simulation output in an iterative fashion.

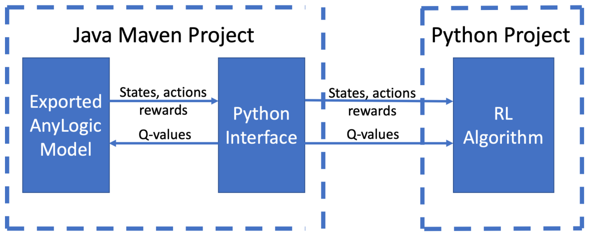

Figure 2: The simulation-based AI framework.

The AnyLogic model simulates the aforementioned task for one episode. It ends when either the robot pushes the bomb into the river, or the bomb explodes after 200 steps. After simulation, the model outputs all the states, actions and rewards that occurred during the episode to the Java project. Then, the outputs are transferred to the Python program through the interface.

Algorithms are run in the Python program to learn better actions for the robot. The learning task is to determine a policy, i.e., a function that maps a state to an action. The policy is calculated from Q-values, which is the expected future reward at a certain state with a specific action. In the Python program, Q-values are updated according to the simulated rewards. The updated Q-values are transferred back to the Java program and used as AnyLogic model inputs. At the next iteration, the AnyLogic model is called, where agents act according to a policy derived from the updated Q-values.