Solving Sokoban with Reinforcement Learning

Omar Tounsi, Joseph Lin, Jihyo Park

October 9, 2024

Objective

The objective of this project is to apply reinforcement learning (RL) algorithms to the Sokoban puzzle game and assess their performance. Furthermore, by comparing RL approaches to traditional search-based algorithms such as depth-first search (DFS) and breadth-first search (BFS), the study aims to explore how these algorithms differ in terms of efficiency and performance.

Motivation

Sokoban is a classic puzzle game where a player must push boxes into designated cells on a grid, with solid walls acting as boundaries and obstacles. Sokoban is a challenging game because the player must carefully plan each move to push boxes to their designated locations without getting stuck during the process. The player can only push boxes, not pull them. This means that if a box is pushed into a corner or against a wall without room to maneuver, it may become stuck, making the puzzle unsolvable. The computational complexity of Sokoban is known to be PSPACE-complete, which means solving Sokoban is as hard as the most difficult problems that can be solved using a polynomial amount of memory (PSPACE). Sokoban presents a challenging environment for testing reinforcement learning due to its complex spatial reasoning, the long-term effect of actions, and the potential for deadlocks. This project provides a meaningful opportunity to assess the strengths and limitations of RL in game-playing, while also benchmarking its performance against traditional search-based algorithms.

Tentative Plan

The game will be represented it as a Markov Decision Process (MDP). The state space will consist of the grid, and the action space will include the agent's possible movements (up, down, left, right). A pre-existing script may be used to generate Sokoban environments based on specific puzzle characteristics, such as box locations, target positions, and the agent's starting point. We will select a set of puzzles with varying difficulty (e.g., grid size, number of boxes, and obstacles) to test the models. The transition function will be deterministic, and the reward function will encourage the agent to place boxes on target spots while penalizing moves that result in trapped boxes. The objective for each puzzle is for the agent to place all boxes in the correct positions while minimizing the number of actions.

We will experiment with both model-based and model-free algorithms. Specifically, we will focus on finite horizon, episodic settings, applying methods such as Monte Carlo and Temporal Difference learning. Each episode will end when the agent either wins the game by placing all the boxes correctly or encounters a dead-end where a box is stuck and cannot be moved. We will need to clearly define the conditions that determine when a box is irreversibly trapped. Additionally, we plan to explore other algorithms beyond Monte Carlo and Temporal Difference as the course progresses.

We intend to measure several key metrics to assess the performance of each algorithm. A tentative list is: percentage of boxes place correctly, number of steps taken by the agent, time and space complexity, and convergence speed (how fast the algorithm learns an effective policy).