COMP300024-Project-A Report

Written by ThePinkCoder

The Analysis of the Searching Problem

The game is formulated as a search problem by having initial state created by provided test cases, actions including slide and swing, goal test which aims to eliminate all lower tokens and the path cost would be counted by turns.

Specifically, a single state would record the situation on the board which contains all upper, lower and block tokens in a single round, and the initial state is the game board based on the test case when the game starts. Through game playing, two actions (slide and swing) would be applied to change the state in each turn. Eventually, the final goal would be non-active lower token on the board, and this is how the goal test detects whether the task has been completed or not. Path cost in this game is calculated by turn but not distance. Therefore, when we tried to get an optimal result, we were generally checking turns (which are also related to execution time and explored states).

The Algorithm

A* search is used in this program to reach the goal. Since the game is for single player and all token moves can be traced and recorded meaning that the information of future path cost is possible to be obtained. In this case, informed searching like Best-first search and A* search would be ideal. However, A* search is more preferred because it would give an optimal solution for this type of problem. In this game, heuristic function can be formulated through the board condition and the distance between upper and lower tokens.

Basically, the strategy for formulating heuristic function is that an upper token would prefer to move toward the closest lower token which it can attack. Yet, the algorithm would not only consider the next move, but also the move after. Specifically, the heuristic function cost is calculated by the total cost from every upper token to eliminate all lower tokens. To ensure the searching is on the right track, the algorithm would choose the state with minimum costs for each upper to eliminate every lower token which can be defeated in priority. For example, when there are two upper scissor tokens and four lower paper tokens, the program would try different combinations of future path to decide the shortest way to make two scissor tokens eliminate four paper tokens at the end. To some extent, the algorithm reduces the number of exploring state which the upper tokens are wasting time on unnecessary detour and it would also designate each upper token to charge their aiming lower tokens in a more efficient way.

Beside the heuristic function, we also improve the algorithm for expanding new states. For instance, after an upper paper token beats all lower rock tokens, its expanded outcome would decrease from six to one (in a general situation). Therefore, it would stop exploring redundant states, since the next step of this token is not important anymore.

In general, the heuristic function is not guaranteed to be admissible because the cost generated by heuristic function “h(n)” would be always greater than the true cost. Since the distance between two positions is calculated by Euclidean distance, the slide and swing actions would always run in a longer path cost. On the other hand, in our A* search algorithm, optimality is not guaranteed duo to the overestimated heuristic function cost. However, the program would have a high performance for finding a solution in a short period time because the heuristic function cost would never be underestimated. For the performance of its completeness, since a duplicated-state check list is used in the program to prevent infinite searching loop, it would always figure out a proper solution unless the initial board is not possible to be resolved.

The Influence of Starting Configuration

The starting configuration including the position and number of tokens would deeply affect the program’s time and space requirement. When more tokens are distributed evenly and widely, the searching would expand more possible state and go through deeper search tree duo to the increase of possible move. Furthermore, if the number of upper token increase, the branching factor would significantly raise because there are more child states generated by current state. On the other hand, if there are less tokens on the initial board and they are centralized at the centre of the board, the number of possible expanded state would be smaller, since the gap distance is shorter. Therefore, the time and space complexity would also become lesser.