Assignment Part A

Rohan Hitchcock and Patrick Randell

1 Introduction

Contained in this report are the details of our approach to Project Part A: Searching, the first assignment in the course COMP30024: Artificial Intelligence at the University of Melbourne 2020 Semester 1.

2 Definitions

Before reading the following, it is worth defining some terms that will be used throughout this report.

- "Components" - In every board configuration there exists black stacks that need to be removed. If a black stack is within the 3x3 radius of another black stack, then both would explode if either one did. All black stacks which are recursively within the 3x3 radius of another are considered to be "components"
- "Goal group" - A goal group is a set of positions for which a white stack may situate itself, and explode, in order to remove one or more components. It is generally, but not always, the collective radius of a component, as we will discuss in the section "Intersections" at the end of the report.

3 Problem Formulation

After trialling many different methods for formulating the problem, we found that the best way for a search algorithm to solve all possible scenarios was to have knowledge of the entire board at each state. Thus, the problem was formulated as follows:

3.1 States

A state was defined as a valid board configuration, including the positions of all black and white stacks on the board, and their heights. Included in a state are the goal groups (see definition): A list of sets of board positions, that white stacks need to be in, in order to "BOOM" and remove all black stacks, winning the game. These goals are generated from the black components in that state. It was important to have these goals included as an attribute of a State, as in order to solve certain scenarios, a goal group needed to be removed so others can be reached.

3.2 Actions

An action is defined as any valid move of a white stack in the current state. This can include moving an entire stack, a subset of a stack, exploding a stack or stacking stacks.

3.3 Path Costs

The cost of a path to a solution is 1 for each action made until the goal test is passed.

3.4 Goal Test

The goal test for this problem was to check whether all black components were removed. More simply, the goal test checks whether all black stacks are removed from the board, and is passed if they are.

4 Search Algorithm

We decided to choose the A* search algorithm for this problem. We chose A* because it is both complete and optimal, and allowed us to carefully design a heuristic that would optimise the search for our needs and requirements. Due to time being the main constraint for this assignment, we were less concerned about the space required by A* to store all nodes.

4.1 Heuristic

The heuristic we used for a state was the minimum moves for any white stack to any goal position plus the number of goal groups in that state. This was calculated by finding the minimum cost from each white stack to any position in the set of coordinate goals, with the possibility of moving h squares at a time (where h is the height of the stack). We then chose the minimum, after calculating this value for each white stack.

Adding the number of goal groups meant that the algorithm was rewarded for removing goal groups. We also included a condition, that if the sum of the heights of all white stacks was less than the number of remaining goal groups (considering intersections of goal groups as a single group), the cost would be infinite, thus pruning branches that would not find a solution early.
A visualisation of this is below.
This heuristic is not admissible, as in certain states, the true cost to a solution would include stacking then moving. However, our heuristic only considers moving the existing stacks on the board, not creating new ones. It is important to note though, that when generating next states, the state including stacking would be given the lowest cost, making it still optimal.

5 Complications and Problem Features

There were several ”classes” of problems that needed to be considered when solving this searching problem.

5.1 Intersections

In some cases, two or more black components would exist that were themselves disjoint, however their explosion radius had overlapping coordinates. Given less white stacks than number of components, these white stacks would have to position themselves in the intersection of the radii of these components in order to explode and arrive at the solution. This situation was solved by calculating the sum of the heights of white stacks on the board, and testing if this sum was less than the number of black components. If it was, we attempted to find any intersections between the radius of black stack components, and if they existed, making these intersections a goal group in place of the black stack components to which they belonged, reducing the number of goal groups (which would otherwise be the collective radius of a component) and potentially making a solution available to our algorithm.

5.2 Stacking

In some cases, the only way to find a solution would be for two white stacks to combine, and move over 1 or more black stacks to reach parts of the board which were otherwise inaccessible. By having each state an entire board configuration, when generating new states, merging stacks was included and thus this problem was solved by design

5.3 Pre-exploding

In some cases, a component had to be removed in order for other white stacks to be able to reach the other goal groups. These cases also ended up being solved by design, as other options would be exhausted and A* would eventually explore this option.

5.4 Combinations

In a case where pre-exploding was required, but the remaining component (after pre-exploding) had an intersection with the first component (the one that was pre-exploded), these cases would not be solved if our algorithm found the intersections of goal groups before testing whether it was required (sum(heights) ¡ num(components)), as it would set the goal position beyond what was accessible. (probs insert picture?)