Assignment Part A

Rohan Hitchcock and Patrick Randell

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

Definition 1. A *component* is a subset of the black stacks on a board such that if one stack in the component explodes then all other stacks in the component will necessarily explode.

Definition 2. A *goal set* (or *goal*) is a subset of board positions in which a white token can explode in order to remove one or more components. For example, the set of positions surrounding a component could be a goal set.

1 Problem Formulation

We approached this problem as an informed search problem on the space of valid board configurations, the goal of which is to arrive at a board configuration does not contain any black tokens via a sequence of valid white actions. Prior to beginning the search, we formulated the goal as follows. We first computed the set of components (Definition 1) of the starting board configuration, and from this a set of goal sets (Definition 2) such that exploding a white token in all of the goal sets will remove all components – and hence black tokens – from the board. In the simplest case the goal sets are the set positions surrounding a component, however in some problems (such as in the second test case) goal sets must be formulated as intersection between positions surrounding two or more components. This decision was made based on the number of white tokens available.

We defined a state as a valid board configuration, which comprises of the position and height of all white and black stacks. Also included in the state are the remaining goal sets. Although it is possible to calculate these from the black stack data it is more efficient to transfer this data between states. A state is a goal state if and and only if its set of goal sets is empty, which is equivalent to a goal state containing no black stacks.

The transitions from a state are given by the valid actions of white stacks. A valid white action is either moving some number of tokens from a stack to another position, or exploding a stack. In order to reduce the branching factor, we chose to only allow a white stack to do an explode action if it was in a goal state.

We defined the true cost of reaching a goal state from a state S as the minimum number of white actions required to reach a goal state starting from S. This means each transition has cost one.

2 Search Algorithm

We decided to choose the A* search algorithm for this problem since it is complete and optimal (given an admissible heuristic). Experimentation showed that the memory use for A* was not prohibitive: the hardest problems required around 2000 nodes, which when considering the size of a node is in the order of 1Mb. Most problems do not require more than 100 nodes.

2.1 The heuristic

For some state, let W be the set of white stacks and G the set of goal sets. In a board with no other stacks, the minimum number of moves of a white stack $w \in W$ of height n at position (a, b) must take to reach position (x, y) is

$$c(w, (x, y)) \coloneqq \left\lceil \frac{|x - a|}{n} \right\rceil + \left\lceil \frac{|y - b|}{n} \right\rceil .$$

This is the L^1-norm dilated according to the movement speed of the stack. Since each goal set $g \in G$ is a set of board positions we define the distance of a goal to a white token to be

$$c(w, g) \coloneqq \min_{(x,y)\in g} c(w, (x, y))$$

We defined the heuristic as the sum of the distance of each goal to its closest white token, plus the cost of exploding at each goal.

$$h(W, G) \coloneqq \sum_{g\in G} \min_{w\in W} c(w, g) + |G|$$

Note that the same white stack may be the closest stack to two or more different goals. While $h(W, G)$ would be a better approximation if each goal had a unique white token in the sum, computing this minimum proved to be more computationally expensive than the gains made by the improvement in the heuristic.

A slight improvement can be made by observing that if $|W| < |G|$ then it is impossible to reach a goal state. This was incorporated into the heuristic like so:

$$h'(W, C) \coloneqq \begin{cases} \infty & |W| < |G| \\ h(W, C) & \text{otherwise} \end{cases}.$$

2.2 Admissibility of the heuristic

This heuristic is not admissible since it does not consider the potential advantage of stacking. For example in Figure 1 the true cost is 3 but our heuristic estimates 4. This means that our implementation is not

Figure 1

optimal. However it is worth noting that our implementation will still find the optimal solution in the situation of Figure 1. Despite the heuristic not being admissible, we believe that starting configurations for which our implementation finds a suboptimal solution are rare. The only examples of such configurations we have managed to find require a 10×10 board.

3 Problem Features

The time and space complexity of A* is $\mathcal{O}(b^{d\cdot E(h)})$, where b is the branching factor, $E(h) = \frac{|h-h^*|}{h*}$ is the error in the heuristic, and d is the average depth of the solution.Let n be the average stack size, m the number of stacks and g the number of goal sets.

The branching factor is $\mathcal{O}(m \cdot n!)$ since any number of tokens can be moved from any of the m stacks. For most states $n \leq 3$, so taking $b = \mathcal{O}(m)$ is a reasonable estimate in the average case.

In a simple board configurations the depth of the solution d is in the order of the length of the board and $E(h)$ is close to zero, however it is easy to come up with configurations where the depth and error is much is much higher. In these situations $E(h)$ roughly of the order of the length of the board.

One might think that increasing the number of goals can increase both d and $E(h)$. While this is true to a point, when g becomes too large both d and $E(h)$ tend to drop because the board is dense with goal sets. Therefore g does not play much of a roll except in pathological configurations.

This gives the complexity of our implementation to be $\mathcal{O}(m^k)$ where k is somewhere between 0 and 64. The large range of k suggests that the running time of our algorithm is far more dependent on the specific board configuration rather than the values of n, m and g. This is supported by experimentation, where configurations with the same values of n, m and g can be designed to have very different running times by exploiting the strengths and weaknesses of the heuristic h.