SHIV NADAR UNIVERSITY

Project Writeup

Group Name: Eagle Nebula	**COURSE: Artificial Intelligence (CSD311)**
Group Number: 24	**Group Members:** **1. Alok Pandey(2010110071)** **2. Harshit Agarwal(2010110824)**

Sokoban

Sokoban is a puzzle game which Hiroyuki Imabayashi devised in 1982. The layout of our Sokoban solver mimics the search strategy adopted by Rolling Stone, JSoko and others, i.e. A* (pronounced "A-star").

Due to its completeness, optimality, and efficiency, the graph traversal and path search algorithm A* suited for our solver. Since it saves all created nodes in memory, it has a $O(b^d)$ space complexity, which is a significant practical limitation. A* is still the best option in many situations, but in practical travel-routing systems, it is typically outpaced by algorithms that can pre-process the graph to achieve higher speed as well as memory-bounded techniques.

Searching in Game Space: We employ A* together with a straightforward Manhattan distance heuristic to locate paths within the game space. Floyd-Warshall or other comparable precomputations are inefficient since the game space changes each time a push is conducted.

Searching in State Space: We will first loosen our definition of a state in Sokoban to condense the search space. A state is the same if the box locations are the same in both states and the player can go from one state to the other without moving any boxes.

The push required to get from one successor to the next is indicated on an edge joining a vertex to its successors.

This definition changes the maximum degree of a vertex from four, which corresponds to the four directions the player could go, to the number of moves that are possible in the associated state.

Finding a route from the starting state to the ending state is the key to solving a Sokoban level. By gathering the pushes made along this path and adding the other required moves in between, a solution may be created. The latter is accomplished by looking for a way through

the game space from the final player position to the location necessary to carry out the subsequent push.

Heuristic: To determine the smallest number of pushes required to complete a Sokoban level from a given state, we employ a heuristic. Each box is given a goal, and the distance between each box and its goal is then added up.

Distance Metric: To calculate how far a box must be moved to move from square A to square B, various metrics can be used. We are only ever concerned with the distance between a square and a goal in our situation.

Manhattan: Two squares (Ax, Ay) and (Bx, By) are separated by the Manhattan distance, denoted as d:

$$d1((Ax, Ay),(Bx, By)) = |Ax - Bx| + |Ay - By|$$

The Manhattan distance is acceptable because a box can only be moved in the four cardinal directions and not diagonally.

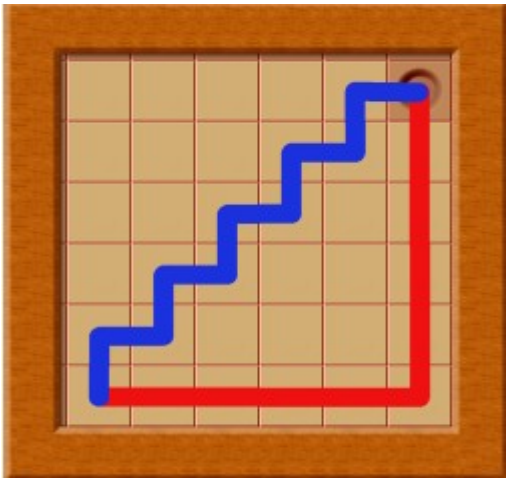

Figure 4.1: The Manhattan distance heuristic. The length along the blue path is the same as the length along the red path.

Pythagorean: In a similar way, the Pythagorean distance is defined:

$$d2((Ax, Ay),(Bx, By)) = \sqrt{((Ax - Bx)^2 + (Ay - By)^{2)}}$$

If two squares are diagonal to one another, they are closer together in relation to the Manhattan distance.

Deadlock: We only take into account straightforward deadlocks, when a box cannot be moved from its current position to a goal at any time, regardless of where the other boxes are situated on the board. The state becomes impossible to solve as soon as we place a box in a square that is a deadlock.

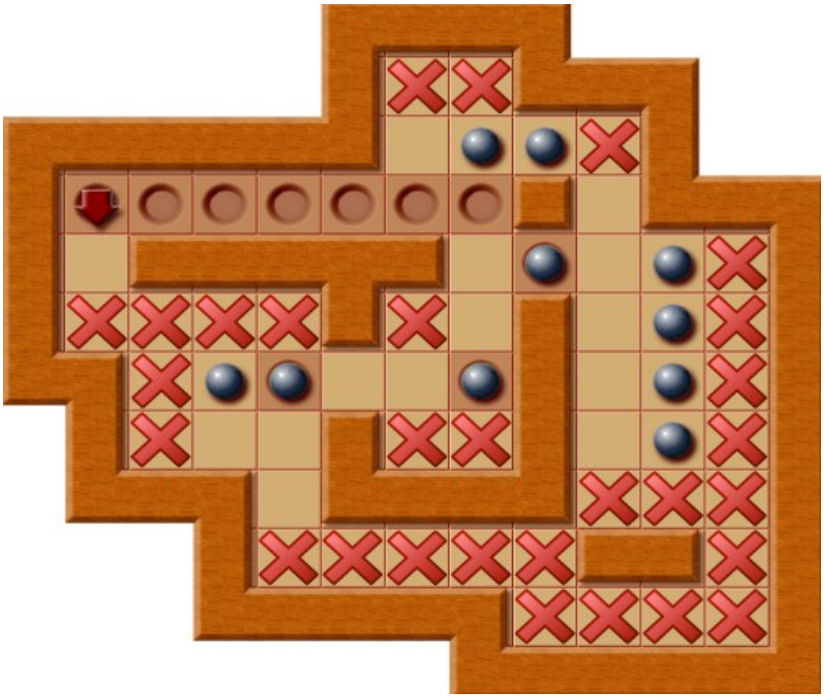

Figure 2: Level 1694 of the collection Sven. All dead squares are marked with a red X.