Report for Sokoban game solution with Path-finding Algorithms (pt. 2)

Tan Ngoc Pham
Faculty of Computer Science, University of Information Technology
19520925@gm.uit.edu.vn

March 2021

1 Greedy Search

1.1 Naive greedy search

I have constructed my "naive" greedy search algorithm by putting a discounting factor γ into the cost function as:

$$CostFunction \cdot \gamma$$

The discount factor can be recognized as the discount factor in Q-learning, which is used to prioritize shorter paths.

And experimentally, with the use of $\gamma = 0.7$, it take off encouraged results in comparison with the uniform cost search. In details, despite the fact that two algorithms itself might give the same answer, the greedy search finds an answer with a faster time.

1.2 Euclidean greedy search

In substitution of using naive greedy search, I use the formula to calculate the euclidean distance from the box to the goal as the heuristic. The euclidean distance is given as below:

$$d(box, goal) = \sqrt{(goal.x - box.x)^2 + (goal.y - box.y)^2}$$

s.t. (x, y) is the *x-axis* and *y-axis* for the respective coordinate of the mentioned object.

This search algorithm gives out more encouraging result in comparison with the naive greedy search in terms of speed. Most of the answers are given under 0.5 seconds with easy maps and under 2 seconds with hard maps.

1.3 Mahattan greedy search

Here I use the Mahattan distance to calculate the distance from the box to the goal, the formula is shown as below:

$$d(box, goal) = |goal.x - box.x| + |goal.y - box.y|$$

For the first thought, I have expected this scale of distance measuring would be better than the euclidean. However, its performance is the same as the one of euclidean or sometimes worse.

1.4 All-in-one greedy search

In the final trial, I have thought of combining all the three different greedy methods I have used so far through this section into one algorithm. The distance from the box to the goal is defined as follow:

$$d(box, goal) = \left(\sqrt{(goal.x - box.x)^2 + (goal.y - box.y)^2} + |goal.x - box.x| + |goal.y - box.y|\right) \cdot \gamma$$

s.t. (x, y) is the *x-axis* and *y-axis* for the respective coordinate of the mentioned object and $\gamma = 0$.

This result is taken off relatively when the time finishing both easy and hard maps is all under 1 seconds, however, there are still unsolvable maps like map 5.

2 A-star ($A*$) Search

The A* search algorithm itself consists of two different heuristic functions for only one algorithm. The first one is the heuristic to calculate the cost for Sokoban in each iteration. And the other one is the heuristic to estimate the distance from the boxes to the goals.

The first heuristic is illustrated as follow:

$$CostFunction = \sum cost_{iteration}$$

And the second one is defined as:

$$d(box, goal) = \sqrt{(goal.x - box.x)^2 + (goal.y - box.y)^2}$$

s.t. (x, y) is the *x-axis* and *y-axis* for the respective coordinate of the mentioned object.

The algorithm itself is considered to be far better in comparison to other path-finding algorithms, and it is. A-star surpassed others with its speed and solution. Almost all the solutions are given under one second, and all the solutions it produces are among the best throughout the beginning.

As far as we can see, the A* algorithm is by far the best path-finding algorithm in both speed and accuracy fields.