Sokoban Search Algorithm*

An Khanh Vo
19520007@gm.uit.edu.vn

March 2021

Abstract

Sokoban is a simple transport puzzle game. In the game, the agent is reponsible for pushing boxes to target points. The rule are quite simple, but it is very difficult for humans when the computations become larger. In this report, we try to find the solution for this game using various informed search algorithm such as Greedy and A*. We have implemented these three algorithms and compare their performance.

1 Heuristic Design

We design the heuristic function as follows: Calculate the distance of each box to each corresponding target. We randomly assign a box with a corresponding target and calculate the distance between them. We randomly assign a box with a corresponding target and calculate the distance between them (in the code we assign based on the index of the 2 arrays). This heuristic function shows the total distance remaining to the boxs could be pushed to the target. We designed two heuristic functions, one for Euclidean distance and one for Manhattan distance.

2 Result Analysis

In order to illustrate the differences between various search algorithms, we are going to show the result of the same map and compare the results with respect to the runtime.

	Greedy(Euclidean)-Runtime(s)	Greedy(Manhattan) - Runtime(s)	A*(Euclidean) Runtime(s)	A*(Manhattan)- Runtime(s)
Level 1	0.36	0.35	0.20	0.21
Level 2	0.03	0.03	0.02	0.02
Level 3	0.50	0.50	0.24	0.24
Level 4	0.03	0.03	0.01	0.01
Level 5	Too long (more than 5 min)	Too long (more than 5 min)	142.59	146.14
Level 6	0.06	0.06	0.05	0.05
Level 7	3.26	3.26	1.94	1.93
Level 8	0.67	0.67	0.65	0.65
Level 9	0.03	0.03	0.02	0.02
Level 10	0.06	0.06	0.06	0.06
Level 11	0.07	0.07	0.07	0.07
Level 12	0.35	0.34	0.32	0.32
Level 13	0.58	0.58	0.64	0.64
Level 14	9.65	9.44	9.04	8.96
Level 15	0.86	0.85	0.87	0.87
Level 16	52.14	51.75	32.02	32.08
Level 17	No solution	No solution	No solution	No solution
Level 18	No solution	No solution	No solution	No solution

Figure 1: Performance comparison between A* and Greedy Search

Clearly, we can see that the speed of the A * algorithm is significantly faster than that of Greedy. Herustic function using Manhattan distance seems slightly faster than the Heuristic function using Euclidean distance. However, the opposite also happens in a small number of cases.

In Greedy Search, as we mentioned, we can see that the performance of the two heuristic functions is heavily dependent on the input. For some simple levels, Greedy Search may render it optimal in a very short time. But in difficult levels, Greedy can still return very bad results.

Like Greedy Search, the performance of A * also depends on the quality of the Heuristic function.

*This is my report for CS106 - Artificial Intelligence (Spring 2021) at University of Information Technology - Vietnam National University HCMC

3 Summary

Personally, I think it is very difficult to design a Heuristic function that is suitable for this problem. Therefore, I think A * and Greedy are not suitable for solving this problem compared with DFS, BFS and UCS.