AI Project 1: Shortest Path Problem

Teng-Sung Yu, Jaehee Jeon, Minseo Choi

September 28, 2022

1 Introduction

This document contains the method of solving the shortest path problem through the A* (A star) algorithm. We implement a weighted A* algorithm with the graph search. A* algorithm is the most informed search algorithm due to its completeness, optimality, and optimal efficiency. In A* search, a priority queue is used to select a node from the frontier, whereas the DFS and BFS algorithms use FIFO and LIFO strategies.
First, in the Method part, there are the explanations of the overall program operation process and each functions. Next, there are the outputs of the program in the Results part.

2 Method

2.1 Process of the Program

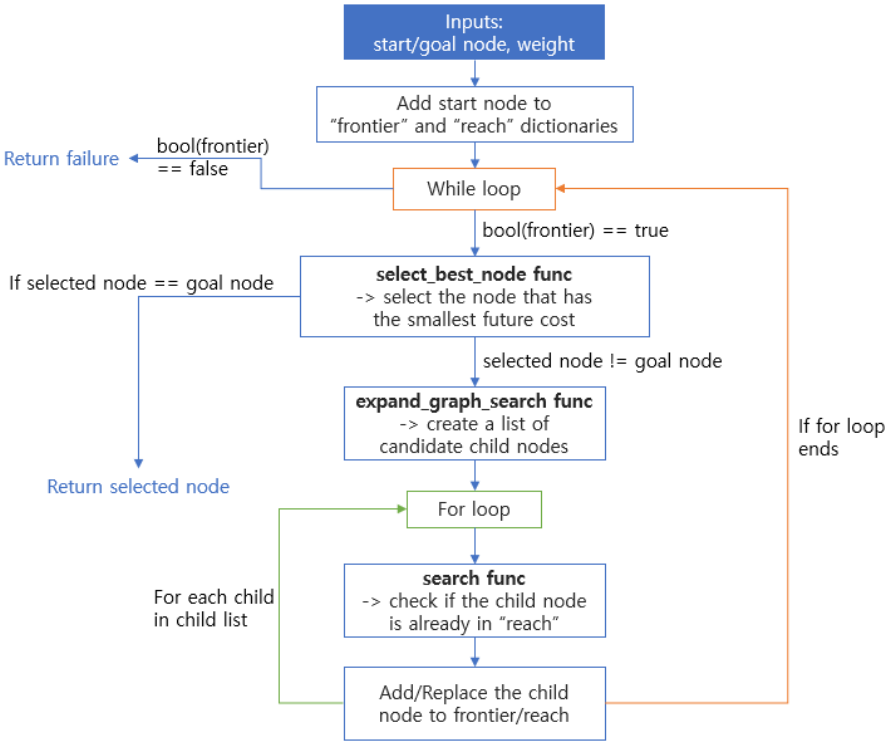

Figure 1: The Flowchart of the Program.

2.2 About Functions

2.2.1. Select Best Node Function
The method to calculate the estimated cost is as follows: $cost = w * g(n) + (1 - w) * h(n)$.
$g(n)$ is the path cost from the initial state to node n. $h(n)$ is the estimated cost of the shortest path from n to a goal state, and it is called heuristic cost. In this program, the heuristic cost is calculated by the following:

$h(n) = |x_n - x_{goalnode}| + |y_n - y_{goalnode}|$. w is the given weight associated with the weighted A* search.
The function select the smallest cost by the for loop, and save the node with the smallest cost. Then, it removes the selected node from the frontier, and return that node.

2.2.2. Expand Graph Search Function
This function returns a $child - list$ containing all nodes that are currently expandable in the node. Expandable refers to all child lists of current nods except for nodes that have been visited once.
The cost of each child(s') is calculated by the following: $C' = the\ cost\ of\ reaching\ state\ s + the\ transition\ cost\ from\ state\ s'\ to\ state\ s$.

3 Results

Here are the results of our program:

Starting node	Ending node	Weight parameter	No. nodes generated	Cost of the path	Sequence of the nodes of the path
0	19	0.5	41	95	[0, 10, 11, 12, 13, 14, 24, 25, 15, 16, 17, 18, 19]
0	19	0	24	96	[0, 1, 2, 12, 13, 14, 24, 25, 15, 16, 17, 18, 19]
11	97	0.5	82	105	[11, 21, 31, 32, 33, 34, 44, 45, 46, 56, 66, 76, 77, 87, 97]
11	97	0.25	35	105	[11, 21, 31, 32, 33, 34, 44, 45, 46, 56, 66, 76, 77, 87, 97]
40	49	0.5	88	112	[40, 41, 51, 52, 42, 43, 44, 45, 46, 47, 48, 38, 28, 29, 39, 49]
49	40	0.5	59	101	[49, 59, 58, 48, 38, 37, 36, 35, 34, 33, 32, 31, 30, 40]
0	99	0.5			no path found
99	0	0.5	94	130	[99, 98, 97, 87, 77, 76, 75, 65, 64, 54, 44, 34, 33, 23, 22, 21, 20, 10, 0]
99	0	0.25	37	130	[99, 98, 97, 87, 77, 76, 75, 65, 64, 54, 44, 34, 33, 23, 22, 21, 20, 10, 0]

4 Discussion

In the case of weight=0 in the weighted A star algorithm, it is the same as the greedy algorithm. Greedy search is a method to reach the final solution by chasing only the locally optimal situation at every moment of selection, but there is no guarantee that it is globally optimal. However, it is efficient that it doesn't take long time, because the number of nodes to search is small. On the other hand, when weight=1, the dijkstra algorithm is followed. This is a dynamic search algorithm that the shortest path can always be obtained. There is a disadvantage that it takes a long time because the number of nodes to be searched is large.

In weighted A star search, it can be seen that the two search methods are combined. It proceeds with a search aimed at finding an approximation of the shortest path, not finding the perfect shortest path. Therefore, the search is faster than Dijkstra's algorithm, and the accuracy is higher than that of greedy search. If w is less than 0.5, a search that is close to greedy search is performed, and if it is greater than 0.5, a search is close to dijkstra search. Therefore, if different paths come out when different weights are used, in the Results part, this is because the priority factors in each search are different.

5 Reference

Kamran Forghani, "Heuristic search.pdf" (Artificial Intelligence, joint section HT22)