Experimental report for the 2021 COM1005 Assignment: The Rambler's Problem*

Your Name

May 23, 2021

1 Description of my branch-and-bound implementation

I create a new java class 'RamblersState', which represents the state of current node. It contains the current coordinate and local cost. The method 'getSuccessors()' in this class will return states in four directions (up, down, left, right) of the current state, and the duplicated successor states in open list and closed list will be removed. The method 'sameState()' will return true if the coordinate of a state is equal to that of the current state. The method 'goalPredicate()' is responsible to decide whether the program can stop, and the criterition is that the current state represents end point and there is no other node in the open list whose global cost is smaller than the end state.

2 Description of my A* implementation

The implementation of the A* is very similar to that of the branch-and-bound. The difference bewteen them is that the implementation of A* should indicate the distance method at the beginning to calculate the heuristics.

*Please, add here your github url

3 Assessing efficiency

3.1 Assessing the efficiency of my branch-and-bound search algorithm

I use the giving map 'tmc', and I try diffent start and end points. The result is shown in the Table 1.

Start Point (x, y)	End Point (x, y)	Efficiency	Iteration time
(0, 0)	(15, 15)	0.144	215
(5, 8)	(15, 15)	0.083	214
(4, 6)	(11, 13)	0.113	238
(5, 3)	(14, 10)	0.134	129
(2, 2)	(10, 8)	0.062	236

Table 1: efficiency of branch and bound algorithm

3.2 Assessing the efficiency of my A* search algorithm

I use the giving map 'tmc', and I try diffent start and end points. The result is shown in the Table 2.

Start Point (x, y)	End Point (x, y)	Efficiency	Iteration time	distance method
(0, 0)	(15, 15)	0.159	195	Manhattan Distance
(0, 0)	(15, 15)	0.158	196	Euclidean distance
(0, 0)	(15, 15)	0.143	217	height difference
(2, 3)	(10, 13)	0.139	194	Manhattan Distance
(2, 3)	(10, 13)	0.139	194	Euclidean distance
(2, 3)	(10, 13)	0.196	138	height difference
(1, 1)	(5, 8)	0.052	237	Manhattan Distance
(1, 1)	(5, 8)	0.054	238	Euclidean distance
(1, 1)	(5, 8)	0.054	230	height difference

Table 2: efficiency of A* algorithm

3.3 Comparing the two search strategies

I use the giving map 'tmc', and I try diffent start and end points. I also use Manhattan Distance as heuristic because the efficiency of Manhattan Distance and Euclidean distance is very similar. The result is shown in the Table 3.

Start Point (x, y)	End Point (x, y)	algorithm	Efficiency	Iteration time
(0, 0)	(15, 15)	BranchAndBround	0.144	215
		Astar	0.159	195
(5, 8)	(15, 15)	BranchAndBround	0.084	214
		Astar	0.092	200
(4, 6)	(11, 13)	BranchAndBround	0.122	230
		Astar	0.124	221
(3, 0)	(13, 11)	BranchAndBround	0.212	101
		Astar	0.438	51
(5, 3)	(14, 9)	BranchAndBround	0.132	122
		Astar	0.249	67

Table 3: comparison of two search strategies

4 Conclusions

The effiency of using different heuristic in A* are almost the same, but in some cases the efficiency of height difference is better than other heuristic methods.

In general, the efficiency of A* is better than branch-and-bound. The effiency of A* algorithm is much better than that of branch-and-bound in some cases, but sometimes A* performs a little better than branch-and-bound.