Marios Pafitis (mpafit02)

EPL341- A* Sokoban Solver, Heuristics Comparison

Heuristics Description

For my A* Sokoban Solver I have used two different heuristics. The heuristic 1 performed much better most of the times in terms of time performance and space allocation than the heuristic 2.

Heuristic 1

For the first heuristic I have used the sum of all Manhattan distances between a misplaced box and an empty target slot plus the steps that they player had already made.

Example:

```
# # # # #
#     .  @ #
#     $     #
#           #
# $     .   #
# # # # #
```

The first '.' (target) is on the position [1, 2]
The second '.' (target) is on the position [4, 3]
The first '$' (box) is on the position [2, 2]
The second '$' (box) is on the position [1, 3]

The Manhattan distance is |2 - 1| + | 2 - 2| = 1 for the first box and the first target.
For the second box and second target the Manhattan distance is |4 - 1| + |3 - 3| = 3
So, the total result for the Manhattan distances would be f = 3 + 1 = 4

If the player had covered g steps then the heuristic function (h1 = g + f) would: h1 = g + 4

Heuristic 2

For the second heuristic I have used the number of misplaced boxes plus the steps that they player had already made.

Example:

```
# # # # #
#     .  @ #
#     $     #
#           #
# $     .   #
# # # # #
```

The number of misplaced boxes '#' is 2. So, f would be equal to 2.

If the player had covered g steps then the heuristic function (h2 = g + f) would: h2 = g + 2

Heuristics Comparison

From my 15 tests with different difficulty I have decided that the heuristic 1 performs better in all the cases. We have significant improvement in the time (color red) in comparison with the heuristic 2. The values that the heuristic 1 generates are closer to the final result (movements) in order to find the solution. The heuristic 1 produces less nodes than the heuristic 2. Especially in the puzzle SOK_MED1.txt the heuristic 1 produced 85% less nodes than the heuristic 2 and the time to solve the puzzle was 95% less. Even in the case of an unsolvable puzzle, the heuristic 1 was able to abort 50% faster than the heuristic 2. The average time for heuristic 1 was 0.172 sec and 0.504 sec for heuristic 2. Almost, 3 times faster the heuristic 1 on average than the heuristic 2. Also, the average number of nodes created by heuristic 1 was 1308 and 2301 for heuristic 2. Approximately 55% less nodes created in heuristic 1 than heuristic 2 on

average. I can clearly see that the performance and tha space needed of an A* algorithm for solving a puzzle problem is fully related on how good my heuristic is, and how close it is to the actual number of steps for reaching the goal. I was expecting those results since the beginning because if a heuristic value is closer to the expected result is going to find a solution faster. The heuristic 1 created all the nodes as heuristic 2 did, but heuristic 2 created more unnecessary nodes which the heuristic 1 did skip.

Note: Currently the code is using the heuristic 1. If you would like to use heuristic 2 you should **uncomment** the line `this.dist = getDist2(puzzle);` and **comment** the line `this.dist = getDist1(puzzle);` which are in the constructor of the **State.java** class.

NODES CREATED & EXPLORED		
PUZZLE/ HEURISTIC	HEURISTIC 1	HEURISTIC 2
SOK_EASY1.txt	25	36
SOK_EASY2.txt	65	75
SOK_EASY3.txt	55	215
SOK_EASY4.txt	46	209
SOK_EASY5.txt	450	994
SOK_MED1.txt	638	4273
SOK_MED2.txt	3354	4019
SOK_MED3.txt	1150	2398
SOK_MED4.txt	2230	5609
SOK_HARD1.txt	1073	1379
SOK_HARD2.txt	1237	1908
SOK_HARD3.txt	3087	4183
SOK_HARD4.txt	3231	5204
SOK_HARD5.txt	2623	3411
SOK_UNSOLVABLE1.txt	353	608
AVERAGE	1307.8	2301.4

TIME TO SOLVE PUZZLE		
PUZZLE/ HEURISTIC	HEURISTIC 1	HEURISTIC 2
SOK_EASY1.txt	0.002	0.003
SOK_EASY2.txt	0.005	0.003
SOK_EASY3.txt	0.003	0.006
SOK_EASY4.txt	0.001	0.012
SOK_EASY5.txt	0.014	0.085
SOK_MED1.txt	0.061	1.195
SOK_MED2.txt	0.591	0.722
SOK_MED3.txt	0.071	0.227
SOK_MED4.txt	0.384	1.733
SOK_HARD1.txt	0.048	0.087
SOK_HARD2.txt	0.100	0.199
SOK_HARD3.txt	0.453	0.787
SOK_HARD4.txt	0.495	1.867
SOK_HARD5.txt	0.335	0.593
SOK_UNSOLVABLE1.txt	0.011	0.048
AVERAGE	0.172	0.504