Comp 6721 - Artificial Intelligence - Project 1 project report

Federico O'Reilly Regueiro - 40012304

October 9th, 2016

1 Intro

The present project touches on informed and uninformed search implementation and heuristics. The implementation was done in Python 3 consisting of 4 source files. A script implementing the particulars of the 8-puzzle and three general search files; a problem and problem-node file, a search-algorithms file and a generic search file. The code is included as part of the deliverable as well as the output of the instance in a text file.

2 Uninformed search

2.a forced left-branch

For uninformed searches, which can take a very long time, an initial state s_0 =`[2, 3, 4, 1, 8, 5, 7, 6, 'B']` was implemented such that the goal state could be found taking the first left branch at every step (or by applying the first available state-transition). For this initial state, as can be expected, both bfs and dfs performed equally, finding a minimal path to the goal in roughly 0.00008 seconds for DFS and 0.0001 for BFS. We have timed, with Python's `timeit` module, the run-time for both algorithms during three iterations in order to minimally weed-out system-related discrepancies. We can see in table 1

algorithm	iter 1 run-time	iter 2 run-time	iter 3 run-time	steps
DFS	0.00088113 s	0.000081716 s	0.000079201 s	6
BFS	0.00614800 s	0.001547771 s	0.001599457 s	6

Table 1: Times for solving uninformed searches of the forced-left branch to solution from s_0 =`[2, 3, 4, 1, 8, 5, 7, 6, 'B']` along with the number of steps to the solution

algorithm	run-time in secods	steps to solution
DFS	0.23411961	550
BFS	0.000525982	2

Table 2: Times for solving an uninformed search for the solution to s_0 =`[2, 3, 4, 1, 8, 5, 7, 6, 'B']` along with the number of steps of the found path to the solution

2.b random start state

Secondly, a random initial state was implemented by applying 27 random state-transitions to the goal state. This was done both to ensure relative proximity of the start and goal states in the state-space and to ensure that the initial state could transition to the goal state, since the state-space is partitioned in two and half of it is inaccessible from the goal state.

A single run of uninformed searches with a random start-state was performed. [1] For the particular instance of the program, s_0 = `[1, 2, 3, 8, 6, 4, 7, 5, 'B']`. The results were conclusive regarding both the search time and the optimality of bfs as can be seen from 2.

3 Heuristic search

For heuristic search, the three given heuristics: $h_{md}(s)$ = `manhattan_distance(s)`, $h_{op}(s)$ = `out_of_place(s)` and h_{min} = `min(h_md(s), h_op(s))`, were implemented. Additionally, both an inadmissible heuristic $h_i(s)$ and two admissible heuristics, $h_{pt}(s)$ = `push_tiles(s)` and $h_b = max(h_{pt}(s), h_{md}(s))$ were implemented

$h_{pt}(s)$ consists of computing, for each out_of_place tile, t_{op_i}, the sum of manhattan distances from the blank position to each one of the tiles that lie between t_{op_i} and its goal position. once all displacement sums have been calculated, the maximum total sum for a given t_{op_j} is chosen and all other $t_{op_n}, s.t. n \neq j$ add one per tile to the heuristic function of the given state.

The rationale behind $h_{pt}(s)$ is that the tiles that lie between any t_{op_i} and its goal position will need to move out of the way for t_{op_i} to advance towards its goal. Only one such sum of movements is chosen since in some cases, these movements might reduce the movements needed to take other t_{op_n} to their respective goal positions *eg.*`[8, 1, 2, 'B', 4, 3, 7, 6, 5]`[2].

Performance-wise, $h_{pt}(s)$ performs at least as well as $h_{op}(s)$ since they are equivalent in some limit cases such as, again, `[8, 1, 2, 'B', 4, 3, 7, 6, 5]` position but generally $h_{pt}(s)$ will yield higher values since it will grow at

[1] A single run was performed since for some states dfs can take well over an hour on the i5 available to the author.

[2] From such a position one needs only to move al pieces counterclockwise on the top 2 rows to reach the goal state

```
...

inadmissible  f(n) =  24 , where h =  16 , and cost =  8
- - - - - - - - - -
   8    1    2
   7    4    6
   B    5    3
inadmissible  f(n) =  25 , where h =  16 , and cost =  9
- - - - - - - - - -
   8    1    2
   B    4    6
   7    5    3
inadmissible  f(n) =  24 , where h =  14 , and cost =  10
- - - - - - - - - -

   B    1    2
   8    4    6
   7    5    3
inadmissible  f(n) =  23 , where h =  12 , and cost =  11
- - - - - - - - - -

...
```

Figure 1: Non-monotonic progression of $f(s)$ along the solution path of $s_0 =$ `[7, 8, 1, 5, 4, 2, 3, 'B', 6]` given by A^* with the inadmissible heuristic $h_i(s)$

least 1 for everyt_{op_i} plus the sum of displacements for a single tile to get to the goal state.

As an inadmissible heuristic, $h_i(s) = h_{op}(s) + h_{pt}(s)$ was chosen, yielding relatively good results but we can observe that for the first iteration, with $s_0 =$ `[7, 8, 1, 5, 4, 2, 3, 'B', 6]`, $f(s)$ is not characterized by monotonic growth. Namely, during the 10^{th} and 11^{th} steps of the solution path, $f(s)$ decreases from the prior step as can be seen in 1.

4 presents the results for the configurations presented in 3. It is clear that there is a trade-off regarding the speed of the search and the optimality of it, where Best First tends to find a sub-optimal solution fairly quickly compared to A*. We can also see that the choice of h_{min} performs very poorly both in he time it takes to find a solution and in the number of steps (see h_{min} BF for $s_{0,1}$) whereas h_b performed significantly better in most instances.

$s_{0,1}$			$s_{0,2}$			$s_{0,3}$		
7	8	1	3	5	1	1	6	4
5	4	2	6	4	2	8	2	3
3	B	6	8	7	B	7	5	B

Table 3: Initial board configurations obtained pseudo-randomly for the heuristic searches during the instance of the program used for this report.

algo - h(s)	$s_{0,1}$ rt / steps	$s_{0,2}$ rt / steps	$s_{0,3}$ rt / steps
BF - h_{op}	0.6570 s / 959	0.3705 s / 700	0.0369 s / 166
BF - h_{md}	0.2913 s / 591	0.1307 s / 364	0.0276 s / 118
BF - h_{min}	6.4226 s / 2801	0.1142 s / 318	0.0329 s / 118
BF - h_{pt}	0.7811 s / 959	0.4481 s / 700	0.0553 s / 166
BF - h_b	0.1351 s /295	0.7373 s / 928	0.4864 s / 698
BF - h_i	0.7999 s / 959	0.4651 s / 700	0.0595 s /166
A* - h_{op}	13.5955 s / 21	308.8241 s / 24	0.0222 s / 12
A* -h_{md}	27.8330 s / 21	367.9768 s / 24	0.0860 s / 12
A* - h_{min}	45.5584 s / 21	692.7713 s / 24	0.0920 s / 12
A* - h_{pt}	14.7079 s / 21	296.3946 s / 24	0.0269 s /12
A* - h_b	5.6599 s / 21	157.2643 s /24	0.0322 s /12
A* - h_i	1.1148 s / 21	20.7655 s / 24	0.1925 s / 12

Table 4: Times for solving informed searches for $s_{0,1}$, $s_{0,2}$ and $s_{0,3}$ shown on 3 using different heuristic functions.