Heuristic Analysis

Tianpei Xie

Dec 7th., 2017

1 Optimal plan

- For Problem 1, the shortest path length plan is

$$\begin{aligned}&Load(C2, P2, JFK)\\&\to Load(C1, P1, SFO)\\&\to Fly(P2, JFK, SFO)\\&\to Unload(C2, P2, SFO)\\&\to Fly(P1, SFO, JFK)\\&\to Unload(C1, P1, JFK)\end{aligned}$$

- For Problem 2, the shortest path length plan is

$$\begin{aligned}&Load(C2, P2, JFK)\\&\to Load(C1, P1, SFO)\\&\to Load(C3, P3, ATL)\\&\to Fly(P2, JFK, SFO)\\&\to Unload(C2, P2, SFO)\\&\to Fly(P1, SFO, JFK)\\&\to Unload(C1, P1, JFK)\\&\to Fly(P3, ATL, SFO)\\&\to Unload(C3, P3, SFO)\end{aligned}$$

- For Problem 3, the shortest path length plan is

$$\begin{aligned}&Load(C2, P2, JFK)\\&\to Load(C1, P1, SFO)\\&\to Fly(P2, JFK, ORD)\\&\to Load(C4, P2, ORD)\\&\to Fly(P1, SFO, ATL)\\&\to Load(C3, P1, ATL)\\&\to Fly(P1, ATL, JFK)\end{aligned}$$

$$\rightarrow Unload(C1, P1, JFK)$$
$$\rightarrow Unload(C3, P1, JFK)$$
$$\rightarrow Fly(P2, ORD, SFO)$$
$$\rightarrow Unload(C2, P2, SFO)$$
$$\rightarrow Unload(C4, P2, SFO)$$

2 Comparison of non-heuristic search

Searching strategy	optimality	time elapsed	number of node expansions	new nodes	plan length
breadth-first-search	min-plan-length	0.0201 sec.	43	180	**6**
breadth-first-tree-search	min-plan-length	0.6608 sec.	1458	5960	**6**
depth-first-graph-search	min-time/fastest	**0.0050** sec.	**12**	48	12
depth-limited-search		0.0641 sec.	101	414	50
uniform-cost-search	min-plan-length	0.0255 sec.	55	224	6

Table 1: Comparison of performance for problem 1

Searching strategy	optimality	time elapsed	number of node expansions	new nodes	plan length
breadth-first-search	min-plan-length and fastest	**5.8467** sec.	3343	30509	**9**
breadth-first-tree-search		not finished (> 10 min)	*	*	*
depth-first-graph-search		8.6659 sec.	**1669**	14863	1444
depth-limited-search		603.6768 sec.	222719	2054119	50
uniform-cost-search	min-plan-length	8.0126 sec.	4853	44041	9

Table 2: Comparison of performance for problem 2

We summarize results as followings:

- As shown in Table 4, Table 5 and Table 6, the **BFS** always find the *shortest path* within a relatively limited amount of time.
- On the other hand, the **DFS** has the least node expansions. For Problem 2 and Problem 3, the DFS find the solution with *minimal time*, i.e. DFS is fastest in these problems. The

Searching strategy	optimality	time elapsed	number of node expansions	new nodes	plan length
breadth-first-search	min-plan-length	28.3921 sec.	14663	129631	**12**
breadth-first-tree-search		not finished (> 10 min)	*	*	*
depth-first-graph-search	fastest	**1.9802** sec.	592	4927	571
depth-limited-search		not finished (> 10 min)	*	*	*
uniform-cost-search	min-plan-length	35.9198 sec.	18223	159618	12

Table 3: Comparison of performance for problem 3

drawback is that the returned path length is always is largest.

- Both DFS and BFS (and uniform-cost-search) can find a solution within 10 mins. But the depth-limited-search may need more time.
- Uniform-Cost-Search has similar performance as BFS, which finds the shortest path.
- The BFS-tree-search may not find a solution when the graph is not tree.
- Depth-limited-search set upper bounds on the length of returned path 50, but requires a great amount of node expansion and is very slow. (> 10 min for Problem 2 and Problem 3)

3 Comparison of A* with different heuristic

Searching strategy	optimality	time elapsed	number of node expansions	new nodes	plan length
A^* with ignore precondition	min-plan-length and fastest	**0.0271** sec.	41	170	**6**
A^* with level sum	min-plan-length	8.04801 sec.	**11**	50	**6**

Table 4: Comparison of A^* performance for problem 1

Searching strategy	optimality	time elapsed	number of node expansions	new nodes	plan length
A^* with ignore precondition	min-plan-length and fastest	3.0463 sec.	1450	13303	**9**
A^* with level sum		4449.5539 sec.	**86**	841	9

Table 5: Comparison of A^* performance for problem 2

We summarize results as followings:

Searching strategy	optimality	time elapsed	number of node expansions	new nodes	plan length
A^* with ignore precondition	min-plan-length	11.9714 sec.	5040	44944	**12**
A^* with level sum		> 20 min.	$*$	$*$	$*$

Table 6: Comparison of A^* performance for problem 3

- A^* algorithm with both heuristics can find the optimal solution with shortest path.
- Using level-sum heuristic, the A^* has least node expansion but is significant slower than the algorithm with ignore condition heuristic. Using ignore condition heuristic, the algorithm is faster.

4 Best heuristic and non-heuristic search

- In terms of *speed*, the best heuristic search is A^* with *ignore precondition heuristic*. In terms of *node expansion*, A^* with *level-sum heuristic* has fewer node expansion.
- The optimality of A^* search is given in book [1] chapter 3.6.1, with a simplied proof seen in footnote[1]. In particular, both ignore preconditions and level-sum heuristics satiesfies the admissable and monotonic conditions.
- A^* search algorithm with both heuristics can find the shortest path plan, *same* as the BFS. This is because both heuristic is admissible and optimistic. With level-sum heuristic, it can have significantly *fewer* node expansion than BFS.
- Compare to DFS, A^* search algorithm with both heuristics can find the shorter path plan. With level-sum heuristic, it can have similar amount of node expansion as DFS.
- The reason ignore precondition heuristic is faster than the level-sum heuristic is because the latter requires to run a loop of evaluation until the goal which is approximately in $O(h)$ where h is the level size of longest path. For a complicated problem such as Problem 2 or Problem 3, the longest path is larger than 1000, which is very slow for each step of evaluation.
- The reason A^* returns the shortest path is because the path length is used as the cost.
- With proper choice of heuristic, A^* algorithm can be seen as having merits of both BFS and DFS.

[1] `https://stackoverflow.com/questions/10195780/proof-of-a-algorithms-optimality-when-heuristics-always-underesti`

References

[1] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.