Written Analysis

Optimal plans

Problem 1 contains only 2 cargos, 2 planes and 2 airports. It's very easy to build optimal plan:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Only 6 steps required to solves this problem.

Problem 2 operates with 3 cargos, 3 planes and 3 airports. Solution is to load each cargo to the nearest plane, deliver and unload cargo:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
3 steps per cargo. 9 steps in total.

There are 4 cargos, 4 airports and only 2 planes in Problem 3. Optimal strategy here is for each plane grab cargo in the nearest airport and then fly to pick up next cargo, that needs to be delivered the same airport as first one:

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)
Unload(C3, P1, JFK)
Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)
Unload(C4, P2, SFO)
It requires 12 steps to achieve this goal.

Search result comparison. Non-heuristic search.

First problem was very easy. I run 5 different searches: uniform cost search, depth-first graph search, breadth-first tree search, breadth-first search and depth-limited search. There is already an comparison of breadth-first search, depth-first search and uniform cost search in Udacity lectures[1], that shows that depth-first search is not optimal. But here is results for this problem:

Name	Expansions	Goal Tests	New Nodes	Length	Time
Uniform_cost_search (5)	55	57	224	6	0.026273461000528187
depth_first_graph_search(3)	21	22	84	20	0.01024372800020501
breadth_first_tree_search(2)	1458	1459	5960	6	0.6718043170112651
breadth_first_search(1)	43	56	180	6	0.025069947994779795
depth_limited_search(4)	101	271	414	50	0.07855980400927365

Uniform cost search, breadth-first tree search and breadth-first search perform very well and return optimal plan. Breadth-first search was the fastest between them and has the fewer number of expansions. Depth-first graph search on the other hand has fewer expansions and new nodes in total but returns very poor plan with 20 actions. Depth-limited search returns worst plan that stuck on loading/unloading cargo in one airport. Breadth-first tree search was the slowest because it has too many expansions.

Second problem was more complicated. Breadth-first tree search and depth-limited search did not return any plan in 15 minutes, so they were excluded from comparison. It is expected, because they shown poor performance on trivial Problem 1.

Name	Expansions	Goal Tests	New Nodes	Length	Time
Uniform_cost_search (5)	4852	4854	44030	9	8.586110803007614
depth_first_graph_search(3)	624	625	5602	619	3.053510173020186
breadth_first_tree_search(2)	-	-	-	-	-

[1] Udacity. Artificial Intelligence Nanodegree. Lesson 8. Video 22 - https://www.youtube.com/watch?v=RMt_NiyY4nU

breadth_first_search(1)	3343	4609	30509	9	11.641944442002568
depth_limited_search(4)	-	-	-	-	-

Optimal amount of steps - 9. Only 2 searches lead to optimal plan: uniform cost search and breadth-first search. As in Problem 1, depth-first graph search return the fastest answer, but completely wrong. Uniform costs search had more expansions than breadth first search, but it was 3 seconds faster.

The hardest problem is Problem 3. As in Problem 2 breadth-first tree search and depth-limited search did not return any plan in 15 minutes.

Name	Expansions	Goal Tests	New Nodes	Length	Time
Uniform_cost_search (5)	18149	18151	159020	12	36.3898939260107
depth_first_graph_search(3)	408	409	3364	392	1.4615364930068608
breadth_first_tree_search(2)	-	-	-	-	-
breadth_first_search(1)	14663	18098	129631	12	97.57466169900727
depth_limited_search(4)	-	-	-	-	-

Breadth-first search was significantly faster than other 2 problems and had less expansions, than in Problem 2. But still it's plan unacceptable - 392 steps instead of 12! Uniform cost search and breadth-first search return optimal plans with only 12 steps. But in this problem difference between performance of this two searches became huge - breadth-first search was almost 3 times slower, than uniform cost search.

Search result comparison. Heuristic search using A*.

I run 3 A* searches: A* with the "ignore preconditions" , "level-sum" and "h-1". A* with the "ignore preconditions" is heuristic described in *"Artificial Intelligence: A Modern Approach"*[2] book as "removing all preconditions". A* with the "level-sum" is heuristic that uses planning graph and were described in same book[3]. Heuristic "h-1" is not a real heuristic, it just return always 1.

[2] "*Artificial Intelligence: A Modern Approach"*.Stuart Russel and Peter Norvig. 2nd edition. Chapter 11. Part 2

[3] "*Artificial Intelligence: A Modern Approach"*.Stuart Russel and Peter Norvig. 2nd edition. Chapter 11. Part 4

Here is a results for Problem 1:

Name	Expansions	Goal Tests	New Nodes	Length	Time
astar_search with h_1	55	57	224	6	0.027992707007797435
astar_search with h_ignore_preconditions	41	43	170	6	0.023066354000548017
astar_search with h_pg_levelsum	11	13	50	6	0.6679104220238514

Optimal plan consists of 6 steps and each search return optimal plan. A* with "level-sum" has fewer expansions, but was the slowest. The fastest search was the A* with "ignore preconditions".

Here is the result for more complicated Problem 2:

Name	Expansions	Goal Tests	New Nodes	Length	Time
astar_search with h_1	4852	4854	44030	9	8.696127097005956
astar_search with h_ignore_preconditions	1450	1452	13303	9	3.1014471309999863
astar_search with h_pg_levelsum	86	88	841	9	51.6102150779916

Results is similar to Problem 1 results. All three searches returns optimal plan. A* with "ignore preconditions" was the fastest. Now, with more complex problem, it is visible, that fewer number of expansions leads to the fastest result (comparing to "h-1"). A* with "level-sum" still has significantly smaller amount of expansions.

The last one is the Problem 3:

Name	Expansions	Goal Tests	New Nodes	Length	Time
astar_search with h_1	18149	18151	159020	12	37.62553846600349
astar_search with h_ignore_preconditions	5038	5040	44924	12	11.548922842999673
astar_search with h_pg_levelsum	316	318	2914	12	257.5983053109958

Still, all 3 searches return optimal plans. It looks like A* with "ignore preconditions" still is the fastest and A* with "level-sum" has fewer number of expansions.

Best approach

According to the observed result, the best heuristic was the "ignore preconditions". It is reasonable, because it uses all advantages of A* search, but expand nodes, that are closer to the goal. That is why it perform faster, than fake heuristic "h-1", that always return 1. Let's compare all the A* searches with best non-heuristic approach - uniform cost search.

Problem 1

Name	Expansions	Goal Tests	New Nodes	Length	Time
Uniform_cost_search (5)	55	57	224	6	0.026273461000528187
astar_search with h_1	55	57	224	6	0.027992707007797435
astar_search with h_ignore_preconditions	41	43	170	6	0.023066354000548017
astar_search with h_pg_levelsum	11	13	50	6	0.6679104220238514

Problem 2

Name	Expansions	Goal Tests	New Nodes	Length	Time
Uniform_cost_search (5)	4852	4854	44030	9	8.586110803007614
astar_search with h_1	4852	4854	44030	9	8.696127097005956
astar_search with h_ignore_preconditions	1450	1452	13303	9	3.1014471309999863
astar_search with h_pg_levelsum	86	88	841	9	51.6102150779916

Problem 3

Name	Expansions	Goal Tests	New Nodes	Length	Time
Uniform_cost_search (5)	18149	18151	159020	12	36.3898939260107
astar_search with h_1	18149	18151	159020	12	37.62553846600349
astar_search with h_ignore_preconditions	5038	5040	44924	12	11.548922842999673
astar_search with h_pg_levelsum	316	318	2914	12	257.5983053109958

So, you can see comparison of all best searches, both heuristic and non-heuristic. As you can see, in all problems A* search with "ignore preconditions" always wins. Uniform cost search is slower that A* search with "ignore preconditions", but faster than other A* implementations. It shows, that A* search could be improved, using good heuristic. Together with "removing negative effects" "ignore preconditions" (or "removing all preconditions") were suggested as valuable and admissible heuristics in book *"Artificial Intelligence: A Modern Approach"*: "Ignore preconditions" heuristic relax search planning problem and ignore cases when one action could lead to effect, that negated other action or when one action achieve multiple goals[4]. It is possible because of "subgoals independance" assumption, that claim the cost as sum of independent cost of each subgoal. So, A* search with "ignore preconditions" is best search approach for all 3 problems in this project.

[4] "*Artificial Intelligence: A Modern Approach"*.Stuart Russel and Peter Norvig. 2nd edition. Chapter 11. Part 2