Heuristic Analysis

Abstract

This documents include my experiment of heuristic and non-heuristic planning searches. The algorithms are on the list below;

- Uninformed algorithms
  - breadth_first_search(BFS)
  - breadth_first_tree_search(BFTS)
  - depth_first_graph_search(BFGS)
  - depth_limited_search(DLS)
  - uniform_cost_search(UCS)
- Informed algorithms
  - recursive_best_first_search with h_1
  - greedy_best_first_graph_search with h_1
  - astar_search with h_1
  - astar_search with h_ignore_preconditions
  - astar_search with h_pg_levelsum

There are two types of algorithms, one is Uninformed, and the other is Informed. Uninformed algorithms do not take the knowledge of this problem, i.g. h_1, h_pg_levelsum and so on. The Uninformed algorithms do not contract with the problem so that we can use it any problem field of search, but generally, it needs the delay of elapsed time to find the solution. In spite of Uninformed algorithm, Informed algorithms use the knowledge of the problem using evaluate the function of h_1 or and so on. It is faster than Uninformed one if you choose a good evaluation function.

My result

Hands on with my pen

Air Cargo Problem 1

1. Load(C2, P2, JFK)
2. Load(C1, P1, SFO)
3. Fly(P2, JFK, SFO)
4. Unload(C2, P2, SFO)
5. Fly(P1, SFO, JFK)
6. Unload(C1, P1, JFK)

Air Cargo Problem 2

1. Load(C3, P3, ATL)
2. Fly(P3, ATL, SFO)
3. Unload(C3, P3, SFO)
4. Load(C2, P2, JFK)
5. Fly(P2, JFK, SFO)
6. Unload(C2, P2, SFO)
7. Load(C1, P1, SFO)
8. Fly(P1, SFO, JFK)
9. Unload(C1, P1, JFK)

Air Cargo Problem 3

1. Load(C2, P2, JFK)
2. Fly(P2, JFK, ORD)
3. Load(C4, P2, ORD)
4. Fly(P2, ORD, SFO)
5. Unload(C4, P2, SFO)
6. Load(C1, P1, SFO)
7. Fly(P1, SFO, ATL)
8. Load(C3, P1, ATL)
9. Fly(P1, ATL, JFK)
10. Unload(C3, P1, JFK)
11. Unload(C1, P1, JFK)
12. Unload(C2, P2, SFO)

Computed Result

Air Cargo Problem 1

Type of Algorithm	Name	Optimally	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed[sec]
Uninformed	breadth_first_search	Yes	43	56	180	6	0.053
	breadth_first_tree_search	Yes	1458	1459	5960	6	1.630
	depth_first_graph_search	No	12	13	48	12	0.014
	depth_limited_search	No	101	271	414	50	0.160
	uniform_cost_search	Yes	55	57	224	6	0.116
Informed	recursive_best_first_search with h_1	Yes	4229	4230	17029	6	4.843
	greedy_best_first_graph_search with h_1	Yes	7	9	28	6	0.008
	astar_search with h_1	Yes	55	57	224	6	0.062
	astar_search with h_ignore_preconditions	Yes	41	43	170	6	0.052
	astar_search with h_pg_levelsum	Yes	11	13	50	6	1.341

Comments

- "depth_limited_search" and "depth_first_graph_search" are not optimally, because it could not find plan of length 6.
- "recursive_best_first_search" takes time to get solution, it might be this problem is not only based on the cost of deistance, so if we improve this, we should take the other cost value.
- The other search method looks good to me especially "breadth_first_search" and "astar_search" looks optimal.

Air Cargo Problem 2

Type of Algorithm	Name	Optimally	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed[sec]
Uninformed	breadth_first_search	Yes	3346	4612	30534	9	22.121
	breadth_first_tree_search	No	-	-	-	-	inf
	depth_first_graph_search	No	1124	1125	10017	1085	12.248
	depth_limited_search	No	-	-	-	-	inf
	uniform_cost_search	Yes	4853	4855	44041	9	20.165
Informed	recursive_best_first_search with h_1	No	-	-	-	-	inf
	greedy_best_first_graph_search with h_1	No	998	1000	8982	21	4.048
	astar_search with h_1	Yes	4853	4855	44041	9	19.783
	astar_search with h_ignore_preconditions	Yes	1450	1452	13303	9	6.305
	astar_search with h_pg_levelsum	No	86	88	841	9	99.403

Comments

- "breadth_first_tree_search", "depth_limited_search" and "recursive_best_first_search" could not reach to the solution for calculation time.
- "depth_first_graph_search" and "greedy_best_first_graph_search" could not take the optimal solution which is 9 length, because the problem space is more bigger than Problem 1.
- "astar_search with h_pg_levelsum" could reach the optimal solution, however the elapsed time to get there is delayed.
- The other search method looks nice to me especially "astar_search with h_ignore_preconditions" looks optimal.

Air Cargo Problem 3

Type of Algorithm	Name	Optimally	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed[sec]
Uninformed	breadth_first_search	Yes	14663	18098	129631	12	170.116
	breadth_first_tree_search	No	-	-	-	-	inf
	depth_first_graph_search	No	627	628	5176	596	5.349
	depth_limited_search	No	-	-	-	-	inf
	uniform_cost_search	Yes	18235	18237	159716	12	92.279
Informed	recursive_best_first_search with h_1	No	-	-	-	-	inf
	greedy_best_first_graph_search with h_1	No	5614	5616	49429	22	28.658
	astar_search with h_1	Yes	18235	18237	159716	12	92.797
	astar_search with h_ignore_preconditions	Yes	5040	5042	44944	12	26.600
	astar_search with h_pg_levelsum	No	318	320	2934	12	543.656

Comments

- "breadth_first_tree_search", "depth_limited_search" and "recursive_best_first_search" could not reach to the solution for calculation time.
- "depth_first_graph_search" and "greedy_best_first_graph_search" could not take the optimal solution which is 12 length, because the problem space is more bigger than Problem 1.
- "astar_search with h_pg_levelsum" could reach the optimal solution, however the elapsed time to get there is delayed.
- The other search method looks nice to me especially "astar_search with h_ignore_preconditions" looks optimal.

Conclusion

- The most recommendable algorithm is "astar_search with h_pg_levelsum" in reaching to the optimal solution and time elapsed.
- The other method which can take in the solution seems like a good but the time is not optimal in this experiments.
- In all cases, "depth_first_graph_search" and "depth_limited_search" could not reach the optimal solutions. I think this is because depth first search algorithm is not suitable for this type of problem. This type of problem can take many patterns and deeper nodes with the meaningless movement of airplane i.g. "To make a round-trip unnecessarily." That's why the optimal solution is not in the thick node of the problem space, and the depth first search did not work well on this issue.