Heuristic Analysis

Implement a Planning Search

Jaehwan Park

Here is a result table of planning search for `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3` problems with uninformed and heuristic searches. Also, each optimal solution was provided.

1. Result of experiment

Search	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed	Optimal
Breadth first search	43	56	180	6	0.035s	True
Depth first graph search	21	22	84	20	0.016s	False
Uniform cost search	55	57	224	6	0.040s	True
Greedy best first graph search (h1)	7	9	28	6	0.006s	True
Astar search (h1)	55	57	224	6	0.041s	True
Astar search (h_ignore_preconditions)	41	43	170	6	0.040s	True
Astar search (h_pg_levelsum)	11	13	50	6	12.345s	True

Table 1. Air Cargo Problem 1

<Optimal solution>

Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

<Test command>

$ python run_search.py -p 1 -s 1 3 5 7 8 9 10

Search	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed	Optimal
Breadth first search	3343	4609	30509	9	14.979s	True
Depth first graph search	624	625	5602	619	3.761s	False
Uniform cost search	4853	4855	44041	9	13.342s	True
Greedy best first graph search (h1)	998	1000	8982	15	2.734s	False
Astar search (h1)	4853	4855	44041	9	13.450s	True
Astar search (h_ignore_preconditions)	1450	1452	13303	9	4.786s	True
Astar search (h_pg_levelsum)	86	88	841	9	6928.199s	True

Table 2. Air Cargo Problem 2

<Optimal solution>

Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

<Test command>

$ python run_search.py -p 1 -s 1 3 5 7 8 9 10

Search	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed	Optimal
Breadth first search	14663	18098	129631	12	125.849s	True
Depth first graph search	408	409	3364	392	2.313s	False
Uniform cost search	18235	18237	159716	12	67.714s	True
Greedy best first graph search (h1)	5614	5616	49429	22	21.284s	False
Astar search (h1)	18235	18237	159716	12	77.719s	True
Astar search (h_ignore_preconditions)	5040	5042	44944	12	22.671s	True
Astar search (h_pg_levelsum)	-	-	-	-	-	-

Table 3. Air Cargo Problem 3

<Optimal solution>

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)

<Test command>

python run_search.py -p 3 -s 1 3 5 7 8 9
python run_search.py -p 3 -s 10

2. Analysis

2.1. Uninformed search

* Breadth first search: This search provides an optimal solution, but it has a problem on time execution. Moreover, it requires huge memory consumption from many expansions. So, this search has a trouble to solve complex problem.

* Depth first graph search: This search cannot provide an optional solution. On the table, this search gave longer path length than other searches. But, it takes an advantage to use small memory consumption from small expansions. Also, it has short execution time even though it is not optimal.

* Uniform cost search: This search provides an optimal solution, but it also consumed huge memory like breadth first search. Number of expansions of this search is bigger than that breadth first search.

2.2. Heuristic search

* Greedy best first graph search with h1: It cannot provide an optimal solution. Even though this search takes short execution time and small memory consumption, it provides longer plan length than other searches.

* Astar search with h1: This search provides almost same result with uniform cost search. So, it gave an optimal solution, but consumed huge memory.

* Astar search with h_ignore_preconditions: It provides an optimal solution, and relatively uses small memory consumption from small expansions. Also, it takes shorter time to execute than other searches.

* Astar search with h_pg_levelsum: It provides an optimal solution, and takes relatively small memory from small expansions. But, its execution time is too long to finish problem 3.

2.3. Conclusion

Even though any one search is always not best solution in all cases, Astar search with h_ignore_preconditions is balanced best algorithm to find a solution in this case. It provides an optimal solution and quick execution, and consumed small memory. Astar search with h_pg_levelsum can provide good solution with small memory usage, but it takes too long time to use in real situation. If considering time consumption, Astar search with h_ignore_preconditions is best solution.