Planning Search Algorithm

Description:

Three problems are defined in classical PDDL and implemented in `my_air_cargo_problems.py`. Domain-independent heuristics are defined by implementing relaxed problem heuristics in `my_air_cargo_problems.py` and implementing Planning Graph and automatic heuristics in `my_planning_graph.py`. Various uninformed and informed planning searches are run to find solutions to each of the planning problems.

Results:

Problem 1	Plan length	Expansions	Goal Tests	New Nodes	Time Elasped
breadth_first_search	6	43	56	180	0.071800183
breadth_first_tree_search	6	1458	1459	5960	2.456702697
depth_first_graph_search	20	21	22	84	0.032539054
depth_limited_search	50	101	271	414	0.181736876
uniform_cost_search	6	55	57	224	0.090477624
recursive_best_first_search with h_1	6	4229	4230	17023	6.931618575
greedy_best_first_graph_search with h_1	6	7	9	28	0.011194378
astar_search with h_1	6	55	57	224	0.092055162
astar_search with h_ignore_preconditions	6	41	43	170	0.071862048
astar_search with h_pg_levelsum	6	11	13	50	3.507368893
Problem 2					
breadth_first_search	9	3346	4612	30534	37.14861035
breadth_first_tree_search	X	X	X	X	X
depth_first_graph_search	105	107	108	959	1.002571379
depth_limited_search	X	X	X	X	X
uniform_cost_search	9	4602	4604	41804	73.66396483
recursive_best_first_search with h_1	X	X	X	X	X
greedy_best_first_graph_search with h_1	11	15	17	139	0.135315247
astar_search with h_1	9	4815	4817	43707	76.0619257
astar_search with h_ignore_preconditions	9	1428	1430	13087	20.4587177
astar_search with h_pg_levelsuM	9	81	83	793	789.7398335
Problem 3					
breadth_first_search	12	14120	17673	124926	199.8826258
breadth_first_tree_search	X	X	X	X	X
depth_first_graph_search	288	292	293	2388	3.271364558
depth_limited_search	X	X	X	X	X
uniform_cost_search	12	17057	17059	149807	522.3161578
recursive_best_first_search with h_1	X	X	X	X	X
greedy_best_first_graph_search with h_1	26	2767	2769	25044	58.03241634
astar_search with h_1	12	17860	17862	156595	518.0393372
astar_search with h_ignore_preconditions	12	4933	4935	43845	112.6919054
astar_search with h_pg_levelsum	12	402	404	3698	6290.39393
X = Search time did not return a solution within a reasonable amout of time					

Observations:

Generally, as the complexity of a planning problem increases (as seen in the result from problem 1 to 3) the metrics increases as well. For breadth first search and uniform cost search, resulting metrics, with the exception of plan length, increase exponentially with increase in the complexity of a planning problem. On the other hand, depth first graph search has a linear increase as complexity increases. Results also show that both breadth first search and uniform cost search find optimal solutions but takes more time to compute while depth first graph search produces a non-optimal solution but takes less time to compute.

Results show that A* search with heuristics, ignore preconditions and level-sum, produce optimal solutions for all problems. Though A* search with level-sum search fewer nodes than A* search with ignore preconditions. A* search with level-sum takes more time to compute compared to A* search with ignore preconditions due to additional computation in searching for a solution in a planning graph.

With a tradeoff in computational time, the best performing heuristics using A* search is level-sum as it produces an optimal solution and searches far fewer nodes than the other heuristics. It also performs better compared to the non-heuristic search planning methods in terms of optimality and nodes expanded.

Recommended Plans:

Using A* search with level-sum, the following are optimal plans for the planning problems.

Problem 1:

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

Problem 2:

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

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

Problem 3:

```
Load(C2, P2, JFK)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P1, ATL, JFK)
Unload(C3, P1, JFK)
Unload(C4, P2, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```