Planning

I. Uninformed searches analysis and experiment

Problem 1

Objects: 2 Cargos, 2 Planes, 2 Airports

Fluents = 2C * 2A + 2P * 2A + 2C * 2P = 12

States = 2^{12} = 4096 (Each fluent receive True of False value)

Search	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	43	56	180	6	0.02518
depth_first_graph_search	21	22	84	20	0.01158
uniform_cost_search	55	57	224	6	0.02711

Problem 2

Objects: 3 Cargos, 3 Planes, 3 Airports

Fluents = 3C * 3A + 3P * 3A + 3C * 3P = 27

States = 2^{27} = 134217728 (Each fluent receive True of False value)

Search	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	3343	4609	30509	9	9.52302
depth_first_graph_search	624	625	5602	619	2.35073
uniform_cost_search	4853	4855	44041	9	8.16885

Problem 3

Objects: 4 Cargos, 2 Planes, 4 Airports

Fluents = 4C * 4A + 2P * 4A + 4C * 2P = 32

States = 2^{32} = 4294967296 (Each fluent receive True of False value)

Search	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
breadth_first_search	14663	18098	129631	12	71.02692
depth_first_graph_search	408	409	3364	392	1.19934
uniform_cost_search	18235	18237	159716	12	39.517

Summation:

- breadth_first_search and uniform_cost_search provided mostly the same results in term of time and space. The reason for that is the path_cost function of uniform_cost_search is "c + 1" which have the same meaning of expanding node list only next one level.
- breadth_first_search and uniform_cost_search are more expensive than what depth_first_graph_search consumed (time, space) but they provide optimal solution while depth_first_graph_search does not.

II. Domain-independent heuristics analysis and experiment

Search	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed
Problem 1					
h_1	55	57	224	6	0.02887
h_ignore_preconditions	41	43	170	6	0.02166
h_pg_levelsum	11	13	50	6	0.31286
Problem 2					
h_1	4853	4855	44041	9	8.08372
h_ignore_preconditions	1450	1452	13303	9	2.57244
h_pg_levelsum	86	88	841	9	27.2884
Problem 3					
h_1	18235	18237	159716	12	36.5694
h_ignore_preconditions	5040	5042	44944	12	10.2602
h_pg_levelsum	316	318	2912	12	127.603

Summation:

- A* use cost function f(n) = g(n) + h(n), with h(n) = constant in h_1 heuristic, A* become uniform_cost_search that is the reason why they provide same results.
- By theory, A* with h_ignore_preconditions heuristic is admissible and guarantees to provide an optimal solution. Whereas, A* with

h_ignore_precondition heuristic are not admissible and does not provide an optimal solution. However, when running search test with small planning problem, we can see h_pg_levelsum also perform very well in term of finding optimal solution.

- From the table, we can see h_ignore_preconditions expands more nodes and perform more goal tests than h_pg_levelsum. But h_pg_levelsum consumed more time of calculating the planning graph which may be very expensive in term of time and space for large problem.

III. Conslusion

- It is obvious that h_ignore_preconditions is in average the best heuristic by far for above problems. h_ignore_preconditions is admissible so it guarantees that the solution will be optimal. Based on good heuristic, we also cut many redundant search branches to save a lot of space and to find solution faster. If we only see the result table, we could realize that h_pg_levelsum expands less nodes. But the table did not show us that h_pg_levelsum need to create a planning graph which is also very expensive in term of time and space, that is the reason why h_ignore_preconditions outperforms h_pg_levelsum in term of performance.
- It will not be better than non-heuristic search planning methods for all problems. It is only much more better than in average if we solve random problems. There are some reasons of explaining above statement:
  - From the mathematic represent of the cost function $f(n) = g(n) + h(n)$, we could see heuristic search is the supper set of non-heuristic search(when $h = 0$) then in most cases, heuristic search can bring us to the goal faster than non-heuristic when f(heuristic) grow faster than f(non-heuristic).
  - In some cases, depth_first_graph_search could found the optimal solution in their first recursive iteration then it will be better than heuristic search in those cases.