Heuristic Analysis

All problems are in the Air Cargo domain. They have the same action schema defined, but different initial states and goals as shown below.

– –

Action(Load(c, p, a),
PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
EFFECT: ¬ At(c, a) ∧ In(c, p))
Action(Unload(c, p, a),
PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
EFFECT: At(c, a) ∧ ¬ In(c, p))
Action(Fly(p, from, to),
PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
EFFECT: ¬ At(p, from) ∧ At(p, to))

– –

Part 1

Planning problems

For each of the 3 Planning Domain Definition Language (PDDL) problems, 8 uninformed planning searches were performed. The results are presented in the tables below.

Problem I

Initial state and goal

– –

Init (At(C1, SFO) ∧ At(C2, JFK)
∧ At (P1, SFO) ∧ At (P2, JFK)
∧ Cargo(C1) ∧ Cargo(C2)
∧ Plane(P1) ∧ Plane(P2)
∧ Airport(JFK) ∧ Airport(SFO))
Goal (At (C1, JFK) ∧ At (C2, SFO))

– –

Results

Algorithms	Expansions	Goal Tests	New Nodes	Plan Length	Time elapsed [s]
breadth_first_search	43	56	180	6	0.03694
depth_first_graph_search	21	22	84	20	0.01979
breadth_first_tree_search	1458	1459	5960	6	0.86006
depth_limit_search	101	271	414	50	0.09149
uniform_cost_search	55	57	224	6	0.05045
recursive_best_first_search h_1	4229	4230	17023	6	2.53817

greedy_best_first_graph_search h_1	**7**	**9**	**28**	**6**	**0.00749**
A*_search h_1	55	57	224	6	0.04329
A*_search h_ignore_preconditions	41	43	170	6	0.04467
A*_search h_pg_levelsum	11	13	50	6	0.81016

The optimal plan for the problem:

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

From the non - heuristic search algorithms used to solve the planning problem, Depth First Search and Depth limit search are the only 2 that did not generate an optimal p lan. The Greedy Best First Graph Search did less node expansion and the execution time is smaller than any of the other algorithms.

Problem II

Initial state and goal

———————————————————————
Init (At (C1, SFO) ∧ At (C2, JFK) ∧ At (C3, ATL)
∧ At (P1, SFO) ∧ At (P2, JFK) ∧ At (P3, ATL)
∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal (At (C1, JFK) ∧ At (C2, SFO) ∧ At (C3, SFO))
———————————————————————

Results

Algorithms	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time elapsed [s]**
breadth_first_search	3343	4609	30509	9	12.96411
depth_first_graph_search	624	625	5602	619	3.16617
breadth_first_tree_search	-	-	-	-	-
depth_limit_search	-	-	-	-	-
uniform_cost_search	**4853**	**4855**	**44041**	**9**	**10.66544**
recursive_best_first_search h_1	-	-	-	-	-
greedy_best_first_graph_search h_1	998	1000	8982	21	2.31460

A*_search h_1	4853	4855	44041	9	11.75905
A*_search h_ignore_preconditions	1450	1452	13303	9	13.25475
A*_search h_pg_levelsum	86	88	841	9	221.65698

The optimal plan for the problem:

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)

From the non-heuristic search algorithms used to solve the problem, Depth First Search, Greedy Bee First Graph Search are such that did not generate an optimal plan. Moreover, the Breadth First Search, Depth Limit, and Recursive Best Search took too much time to calculate so it was terminated. The Uniform Cost Search did less node expansion and the execution time is smaller that any other algorithms.

Problem III

Initial state and goal

———————————————————————
Init (At (C1, SFO) ∧ At (C2, JFK) ∧ At (C3, ATL) ∧ At (C4, ORD)
    ∧ At (P1, SFO) ∧ At (P2, JFK)
    ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
    ∧ Plane(P1) ∧ Plane(P2)
    ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal (At (C1, JFK) ∧ At (C3, JFK) ∧ At (C2 , SFO) ∧ At (C4, SFO))
———————————————————————

Results

Algorithms	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time elapsed [s]**
breadth_first_search	14663	18098	129631	12	148.95685
depth_first_graph_search	408	409	3364	392	2.65480
breadth_first_tree_search	-	-	-	-	-
depth_limit_search	-	-	-	-	-
uniform_cost_search	18223	18225	159618	12	70.61659
recursive_best_first_search h_1	-	-	-	-	-

greedy_best_first_graph_search h_1	5578	5580	49150	22	20.05542
A*_search h_1	**18223**	**18225**	**159618**	**12**	**64.07221**
A*_search h_ignore_preconditions	5040	5042	44944	12	14.79145
A*_search h_pg_levelsum	325	327	3002	12	1247.94145

The optimal plan for the problem:

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

From the non-heuristic search algorithms used to solve the planning problem, Depth First Search and Greedy Best First Graph Search are the only 2 that did not generate an optimal plan. Moreover, the Breadth First Tree Search, Depth Limit and Recursive Best First Search took too much time to calculate so it was terminated. The A* search has an execution time slightly smaller than any of the other algorithms.

Part 2

Domain-independent heuristic

For the 3 PDDL, 2 A* searches with domain-independent heuristics were performed. The first one called h_ignore_preconditions is taking the minimum number of actions from the current state in order to satisfy all the goal conditions by ignoring the preconditions of all the actions. The second one called h_pg_sum created and uses a planning graph to estimate the number of actions that must be undertaken from the current state in order to satisfy all the goal conditions. The results are presented in the tables below.

For problem I

Algorithms	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time elapsed [s]**
breadth_first_search	43	56	180	6	0.03694
depth_first_graph_search	21	22	84	20	0.01979

breadth_first_tree_search	1458	1459	5960	6	0.86006
depth_limit_search	101	271	414	50	0.09149
uniform_cost_search	55	57	224	6	0.05045
recursive_best_first_search h_1	4229	4230	17023	6	2.53817
greedy_best_first_graph_search h_1	7	9	28	6	0.00749
A*_search h_1	55	57	224	6	0.04329
A*_search h_ignore_preconditions	**41**	**43**	**170**	**6**	**0.04467**
A*_search h_pg_levelsum	**11**	**13**	**50**	**6**	**0.81016**

Both heuristics generated the optimal plans. The A*_search with h_pg_levelsum had less goal tests and expanded the least amount of node, but had a longer execution time compare to the A*_search with h_ignore_preconditions algorithm.
Moreover, if non-heuristic algorithms are compared with the heuristic one, the greedy_best_first_search_h1 is still the better option since it has less node expansion and the execution time is smaller than any of the other algorithms.

For problem II

Algorithms	**Expansions**	**Goal Tests**	**New Nodes**	**Plan Length**	**Time elapsed [s]**
breadth_first_search	3343	4609	30509	9	12.96411
depth_first_graph_search	624	625	5602	619	3.16617
breadth_first_tree_search	-	-	-	-	-
depth_limit_search	-	-	-	-	-
uniform_cost_search	4853	4855	44041	9	10.66544
recursive_best_first_search h_1	-	-	-	-	-
greedy_best_first_graph_search h_1	998	1000	8982	21	2.31460
A*_search h_1	4853	4855	44041	9	11.75905
A*_search h_ignore_preconditions	**1450**	**1452**	**13303**	**9**	**13.25475**
A*_search h_pg_levelsum	**86**	**88**	**841**	**9**	**221.65698**

Both heuristics generated the optimal plans. The A*_search with h_pg_levelsum had less goal tests and expanded the least amount of node, but had a longer execution time compare to the A*_search with h_ignore_preconditions algorithm.

Moreover, if non-heuristic algorithms are compared with the heuristic one, the A*_search with h_ignore_preconditions is a better option since it has less node expansion and the execution time is smaller than any of the other algorithms.

For problem III

Algorithms	Expansions	Goal Tests	New Nodes	Plan Length	Time elapsed [s]
breadth_first_search	14663	18098	129631	12	148.95685
depth_first_graph_search	408	409	3364	392	2.65480
breadth_first_tree_search	-	-	-	-	-
depth_limit_search	-	-	-	-	-
uniform_cost_search	18223	18225	159618	12	70.61659
recursive_best_first_search h_1	-	-	-	-	-
greedy_best_first_graph_search h_1	5578	5580	49150	22	20.05542
A*_search h_1	18223	18225	159618	12	64.07221
A*_search h_ignore_preconditions	**5040**	**5042**	**44944**	**12**	**14.79145**
A*_search h_pg_levelsum	**325**	**327**	**3002**	**12**	**1247.94145**

Both heuristics generated the optimal plans. The A*_search with h_pg_levelsum had less goal tests and expanded the least amount of node, but had a longer execution time compare to the A*_search with h_ignore_preconditions algorithm.
Moreover, if non-heuristic algorithms are compared with the heuristic one, the A*_search with h_ignore_preconditions is a better option since it has less node expansion and the execution time is smaller than any of the other algorithms.

CONCLUSION

To conclude, both heuristic algorithm generated the optimal plan for all the problem. Also, except for problem I, the the A*_search with h_ignore_preconditions is a better option since it has less node expansion and the execution time is smaller than any of the other algorithms. Moreover, A*_search with h_pg_levelsum did the less evaluations and expansions but the execution time is long.