Planning:
Heuristic Analysis

Raphael Ballet

This work presents the results of the "Planning Search" project. It uses several search methods with different heuristics to solve the air cargo problem defined in classical PDDL (Planning Domain Definition Language).

1 Air Cargo Problems

The aircargo action schema is shown bellow. It is used for all three air cargo problems.

```
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))
```

Each air cargo problem differs from one another only in the definition of its initial state and objective.

- Problem 1:

```
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))
```

- Problem 2:

```
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))
```

- Problem 3:

```
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))
```

1.1 Optimal Plan

A possible optimal plan for each of the aforementioned problems are shown bellow:

- Problem 1: 6 steps

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

- Problem 2: 9 steps

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

- Problem 3: 12 steps

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

2 Planning Search

Table 1 shows the results for all search methods used to solve the three different versions of the air cargo problem. It compares the non-heuristic search methods (breadth-first-search, breadth-first-tree-search, depth-first-graph-search, depth-limited-search, and uniform-cost-search) and heuristic search methods (recursive-best-first-search, greedy-best-first-graph-search, and Astar-search), which uses a constant as heuristic (h_1, which is not a true heuristic), and two other automatically generated heuristics (ignore-preconditions and planning-graph-level-sum). Note that if the time elapsed to complete the planning search were higher than 10 minutes (600 seconds), the program was interrupted.

2.1 Non-Heuristic Search Methods

This Section compares the five different non-heuristic search methods present in Table 1.

The breadth-first-search, breadth-first-tree-search, and the uniform-cost-search methods reached the objective with an optimal plan, as expected. On the other hand, the depth-first-graph-search and the depth-limited-search reached the objective plan, but the plan length was not optimal.

The breadth-first-search method presented a considerable advantage over the other methods for all problems. Comparing it with the other search methods that achieved an optimal plan, it presented the least number of expansions, goal tests, new nodes formation, and total elapsed time to obtain the plan.

Table 1 - Air Cargo search methods comparison

	Search Methods	Expansions	Goal Tests	New Nodes	Plan Length	Time Elapsed (s)
Problem 1	breadth_first_search	43	56	180	6	0.0184
	breadth_first_tree_search	1458	1459	5960	6	0.5566
	depth_first_graph_search	12	13	48	12	0.0051
	depth_limited_search	101	271	414	50	0.0586
	uniform_cost_search	55	57	224	6	0.0236
	recursive_best_first_search (h_1)	4429	4230	17029	6	1.6279
	greedy_best_first_graph_search (h_1)	7	9	28	6	0.0031
	astar_search (h_1)	55	57	224	6	0.0237
	astar_search (h_ignore_preconditions)	41	43	170	6	0.0188
	astar_search (h_pg_levelsum)	11	13	50	6	1.2129
Problem 2	breadth_first_search	3343	4609	30509	9	9.6312
	breadth_first_tree_search	-	-	-	-	> 600
	depth_first_graph_search	582	583	5211	575	2.3096
	depth_limited_search	-	-	-	-	> 600
	uniform_cost_search	4719	4721	42839	9	33.6043
	recursive_best_first_search (h_1)	-	-	-	-	> 600
	greedy_best_first_graph_search (h_1)	455	457	4095	29	1.7331
	astar_search (h_1)	4768	4770	43291	9	32.8839
	astar_search (h_ignore_preconditions)	1462	1464	13400	9	9.1664
	astar_search (h_pg_levelsum)	82	84	801	9	119.2325
Problem 3	breadth_first_search	14663	18098	129631	12	71.0502
	breadth_first_tree_search	-	-	-	-	> 600
	depth_first_graph_search	627	628	5176	596	2.3036
	depth_limited_search	-	-	-	-	> 600
	uniform_cost_search	17190	17192	150989	12	277.6818
	recursive_best_first_search (h_1)	-	-	-	-	> 600
	greedy_best_first_graph_search (h_1)	718	720	6491	26	3.8957
	astar_search (h_1)	17151	17153	150549	12	280.9959
	astar_search (h_ignore_preconditions)	4725	4727	41861	12	55.7429
	astar_search (h_pg_levelsum)	-	-	-	-	> 600

2.2 Heuristic Search Methods

This Section compares the three different search methods present in Table 1, as well as the different heuristics used.

For problem 1, the best heuristic-based search method was the greedy-best-first-graph-search using the h_1 heuristic. It achieved the optimal plan using the least number of extensions, goal tests, new nodes generation, and time elapsed. Nevertheless, it did not achieved an optimal plan for problems 2 and 3. For these problems, the best method was the Astar-search using the h_ignore_preconditions heuristic, which reached the optimal plan. Although the Astar-search using the h_pg_levelsum presented fewer expansion nodes, goal tests, and new nodes generation, it presented a considerably higher computational burden, which is mainly due to the planning graph construction.

Comparing the Astar-search method using the h_ignore_preconditions heuristic with the breadth-first-search method, it presented fewer expansions, goal tests, and new nodes generation, but it the time elapsed to complete the planning search for both methods varied for each problem. For problem 1, the Astar-search took 0.0188 seconds against 0.0184 seconds of the breadth-first-search. For problem 2, the former took 9.1664 seconds against 9.6312 seconds of the later. Finally, for problem 3, the former took 55.7429 seconds against 71.0502. Therefore, it can be noticed that as the state space grows, more advantageous is the former method comparing to the later.