Heuristic Analysis

Optimal Plan

The optimal plan for Problem 1 is

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

The optimal plan for Problem 2 is

```
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))
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)
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

The optimal plan for Problem 3 is

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

Non-Heurstic Search Analysis

Search Algorithm	Problem	Expansions	Goal Tests	New Nodes
breadth_first_search	1	43	56	180
depth_first_search_graph	1	21	22	84
uniform_cost_search	1	55	57	224
breadth_first_search	2	3343	4609	30509
depth_first_search_graph	2	624	625	5602
uniform_cost_search	2	4852	4854	44030
breadth_first_search	3	14463	18098	129631
depth_first_search_graph	3	408	409	3364
uniform_cost_search	3	18223	18225	159

Depth First Search had the lowest expansions, goal tests, and new nodes than the other two search algorithms for all three problems. Breadth First Search and Uniform Cost Serach had similar numbers in expansions and goal tests but differed greatly in the number of new nodes.

Depth First search however was less optimal when it came to number of actions taken and total time to execute the plan.

Search Algorithm	Problem	Plan Length	Time Length (seconds)
breadth_first_search	1	6	0.056 seconds
depth_first_search_graph	1	20	0.026
uniform_cost_search	1	6	0.0635
breadth_first_search	2	6	20.43
depth_first_search_graph	2	9	5.13
uniform_cost_search	2	9	17.036
breadth_first_search	3	12	140.068
depth_first_search_graph	3	392	2.506
uniform_cost_search	3	12	77.953

Breadth First and Uniform Cost Search was able to find the more optimal plans at a higher cost of time.

Heuristic Analysis with A* Search

Heuristic	Problem	Expansions	Goal Tests	New Nodes
`h1`	1	55	57	224
`h_ignore_preconditions`	1	41	43	170
`h_pg_levelsum`	1	11	13	50
`h1`	2	4853	4854	44030
`h_ignore_preconditions`	2	1450	1452	13303
`h_pg_levelsum`	2	86	88	841
`h1`	3	18223	18225	159618
`h_ignore_preconditions`	3	5040	5042	44944
`h_pg_levelsum`	3	315	317	2902

The h1 heuristic (not a true heuristic) had the most expansions, goal tests, and new nodes. This is simple because it was an A* Search. The `h_pg_levelsum` heuristic had the lowest expansions, goal tests and new nodes. However the heuristic was significantly slower.

Search Algorithm	Problem	Plan Length	Time Length (seconds)
`h1`	1	6	0.06967
`h_ignore_preconditions`	1	6	0.061877
`h_pg_levelsum`	1	6	0.7367
`h1`	2	9	16.7752
`h_ignore_preconditions`	2	9	7.008
`h_pg_levelsum`	2	9	58.9361
`h1`	3	12	77.2223
`h_ignore_preconditions`	3	12	24.6258
`h_pg_levelsum`	3	12	300.9521

On the other hand `h_ignore_preconditions` was fast. Sometimes faster than h1 at finding the optimal plan length. This goes to show that loosening constraints in the plan can lead to strong heuristics.

Best Heuristic?

The best heuristic used in these problems was the `h_ignore_preconditions` heuristic. it was able to find the shortest plan length possible in the shortest time. It was even better than non-heuristic search planning with `bread_first_search` and `uniform_cost_search`. However it was not *faster* than `depth_first_search_graph` search planning. This is because the problem was simple enough to find a path by applying actions until the goal was satisfied. This gave `depth_first_search_graph` an advantage because to find a suboptimal plan in a very short period of time.