Report: Heuristic Analysis

The Optional Sequences for the three planning problems are:

Problem1:

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

Problem2:

Load(C1,P1,SF0)
Load(C2,P2,JFK)
Load(C3,P3,ATL)
Fly(P1,SFO,JFK)
Fly(P2,JFK,SFO)
Fly(P3,ATL,SFO)

Problem3:

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

Experimental Results

Problem 1:

Method	Expansions	Goal Tests	Nodes	PlanLength	Time Taken(secs)
Breadth First	43	56	180	6	0.08
BFTreeSrch	1458	1459	5960	6	2.29
DFGraphSrch	21	22	84	20	0.03
DFLimitSrch	101	271	414	50	0.18
UniformCost	55	57	224	6	0.10
RecursiveBFS	4229	4230	17023	6	6.64

GreedyBFS-h1	55	57	224	6	0.01
AStar-h1	55	57	224	6	0.10
AStar-ignorePre	33	35	134	6	0.08
Astar-h_pg_leve	29	31	122	6	2.12

Problem 2:

Method	Expansions	Goal Tests	Nodes	PlanLength	Time Taken(secs)
Breadth First	3343	4609	30509	9	24.8
BFTreeSrch	N/A				
DFGraphSrch	624	625	5602	619	5.3
DFLimitSrch	N/A				
UniformCost	4853	4855	44041	9	59.59
RecursiveBFS	N/A				
GreedyBFS-h1	998	1000	8982	15	10.2
AStar-h1	4853	4855	44041	9	59.1
AStar-ignorePre	2076	2078	18945	9	30.8
Astar-h_pg_leve	N/A				

Problem 3:

Method	Expansions	Goal Tests	Nodes	PlanLength	Time Taken(secs)
Breadth First	3679	4734	61106	11	66.61
BFTreeSrch	N/A				
DFGraphSrch	723	724	11219	265	11.9
DFLimitSrch	N/A				
UniformCost	4933	4935	83600	11	139
RecursiveBFS	N/A				
GreedyBFS-h1	2105	2107	33841	29	48
AStar-h1	4933	4935	83600	11	136
AStar-ignorePre	2609	2611	43259	11	88.3
Astar-h_pg_leve	N/A				

Note that N/A indicates runs that took too long (longer than 30 minutes) and the run was killed.
The optimal solutions sizes for Problems 1, 2, and 3 are 6, 9, and 11, respectively.
Of the deterministic algorithms, Breadth-First-Search consistently provides optimal solutions even though the number of nodes expanded explodes with larger/more constrained problems. Of the heuristic methods,
Astar-h1 and Astar-ignorepreconditions and uniform cost perform very well, and provide optimal solutions. However, Astar-proconditions explores the fewest number of nodes and completes the fastest without sacrificing accuracy.

Uniform Cost and Astar-ignorePreConditions expand the exact same number of nodes and run the same number of goal tests, most likely because the Uniform Cost heuristic can be derived by weakening the preconditions of the problem statement as done in Astar-ignorepre.

Analysis

For very small problem sizes, most of the search techniques perform well and provide optimal solutions, Depth-first Graph Search and Depth-first Tree Search. These two search techniques provide poor solutions that are very far from optimal, even if their run time is reasonable.

The real differences between the search techniques come to the fore, when the problem size gets larger, and when problem constraints are increased. Problem 2 increases the problem size without increasing the number of constraints. That is the number of planes have been increased to match the number of items. Thus, it provides a good method for comparing the scalability of the solutions to larger problems with relaxed constraints, i.e., number of planes vs. number of items.

Depth-first search methods are clearly not practical given the number of nodes it explores for even the small problem sizes in this project. Breadth-First-Search is similarly computationally too expensive and explores too many nodes, with increasing problem size to be of practical use. However, Greedy Breadth-First-Search with h1 heuristic performs faster, though it yields solutions that are off from the optimal solution by a factor of 3.

Of the heuristic methods, variations of Astar algorithm and the uniform cost method perform extremely well, though Astar-h_pg_level takes far too long, indicating that the heuristic expands too many nodes unnecessarily, taking far longer to reach the optimal solution.

Astar-ignorepreconditions and uniform-cost have almost identical performance, and produce optimal results and expand the least number of nodes and do minimum goal checks. These two techniques seem to be clearly the best two in the suite of search methods run in the experiments.

It seems clear that the main tradeoff here is between computation cost and optimality with increasing problem sizes. Simple heuristics based on weakening preconditions along with Astar search method seems like a consistent way to get optimal search results with the least computation cost.