Planning Search Heuristic Analysis

This report is a summarization of my findings for Project 3, a classical planning problem involving Air Cargo. This project began by implementing non-heuristics planning methods then running the uninformed planning searches on the Air Cargo planning problems using *breadth_first_search*, *depth_first_graph_search* and *uniform_cost_ search*. The second phase of the project involved implementing domain independent heuristics to guide the search. This report contains is the definition and a discussion of findings of the three Air Cargo planning problems.

Air Cargo Problem 1:

The first problem involves 2 planes, 2 airports, and 2 pieces of cargo, this can be described with the initial state and goal defined as:

```
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))
```

After running both my the uniformed and heuristic searches against this problem I found the following results:

Air Cargo Problem 1

Search Algorithm	Time (sec)	Plan Length	Expansions	Goal Tests	New Nodes
Breadth First Search	0.056	6	43	56	180
Breadth First Tree Search	0.895	6	1458	1459	5960
Depth First Graph Search	0.008	12	12	13	48
Depth Limited Search	0.112	50	101	271	414
Uniform Cost Search	0.054	6	55	57	224
Recursive Best First Search H1	2.473	6	4229	4230	17029
Greedy Best First Search H1	0.009	6	7	9	28
A* Search H1	0.048	6	55	57	224
A* Search H_ignore preconditions	0.030	6	41	43	170
A* Search H_PG_levelsum	2.850	6	11	13	50

This table shows that a number of the searches were able to find the shortest plan length in a reasonable (less than 10 minutes) amount of running time. The optimal plan to problem 1 is found to be the following sequence of actions:

```
1. Load(C1, P1, SFO) 2.
Load(C2, P2, JFK) 3.
Fly(P1, SFO, JFK)
4. Fly(P2, JFK, SFO)
5. Unload(C1, P1, JFK) 6. Unload(C2, P2, SFO)
```

From analyzing the Air Cargo Problem 1 table the **Greedy Best First Search H1** algorithm is shown as the best choice of algorithm for this small planning problem. It is evident that Greedy Best First Search H1 has a superior performance in terms of running time and least node expansions while producing the optimal plan. The Greedy Best First search algorithm works by trying to expand the node closest to the goal first, based on the premise that it is likely to lead to a solution quickly [Norvig 93]. The greedy algorithm proceeds then at each step tries to get a close as possible to the goal. Because this plan has very few fluents, the greedy algorithm has little work to do in order to achieve this optimal solution.

In this problem Depth First Search had also shown interesting characteristics in the time metric however it did not product an optimal path. Depth First Search always expands the deepest nodes the frontier of the search tree. [Norvig 86] while on finding a path to the goal it often preforms some backtracking which adds additional nodes to the path. As depicted in the table above many of the algorithms here preformed very well due to the smallish search space. Although the Greedy Best First Search H1 uses a heuristic which increases the complexity of the algorithm a non-heuristic planning solution could be deemed an acceptable choice as it can easily find the optimal plan without the need for an informed search.

Air Cargo Problem 2:

The second air cargo problem when compared to the first contains an additional plane, piece of cargo, and airport elements which increases the search space. The problem initial state and goal is defined as:

```
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))
```

After running both my the uniformed and heuristic searches against this problem I found the following results summarized the table.

Air Cargo Problem 2

	Time (sec)	Plan Length	Expansions	Goal Tests	New Nodes
Breadth First Search	11.241	9	3343	4609	30509
Breadth First Tree Search			**Timeout!**		
Depth First Graph Search	2.705	575	582	583	5211
Depth Limited Search			**Timeout!**		
Uniform Cost Search	36.071	9	4761	4763	43206
Recursive Best First Search H1			**Timeout!**		
Greedy Best First Search H1	2.656	15	550	552	4950
A* Search H1	34.977	9	4761	4763	43206
A* Search H_ignore preconditions	10.7430	9	1506	1508	13820
A* Search H_PG_levelsum			**Timeout!**		

From analyzing the table we can see **Breadth First Search** is the best choice of search algorithm in terms of running time and producing an optimal plan. Breadth First Search is based on a simple strategy in which the root node is expanded first then all the successors (level neighbors) of the root node are expanded next.[Norvig 81] This is guaranteed to find the optimal path if the cost of all actions are the same. [Norvig 82] The search space required for this problem was still small enough to be held memory. In a larger search space memory is often a limiting for Breadth First Search over is its execution time. So with ample memory size, and small search space this algorithm demonstrates great performance.

The Breadth First Search algorithm reached the optimal solution in the shortest amount of running time. The found optimal plan for problem 2 can be defined by the 9 action steps:

```
1. Load(C2, P2, JFK) 2.
Load(C1, P1, SFO) 3.
Load(C3, P3, ATL)
4. Fly(P2, JFK, SFO)
5. Unload(C2, P2, SFO)
6. Fly(P1, SFO, JFK)
7. Unload(C1, P1, JFK) 8. Fly(P3, ATL, SFO)
9. Unload(C3, P3, SFO)
```

This results of problem 2 are quite a bit more interesting when compared to the results of problem 1. While the Greedy Best First Search H1 and Depth First Graph Search produced the shortest amount of running time respectively, neither of these algorithms found the optimal path. In this case h1 heuristic guided the agent to a suboptimal plan. Utilizing a different heuristic the greedy algorithm may have been able to find the optimal plan. The suboptimal plane disqualifies these two searches as good candidates because the length of the plan is longer than necessary.

We can also see a very interesting case in similar performance metrics in Expansions, Goal Tests, and New Nodes, in nearly the same amount of running time between the Uniform Cost Search (uninformed search strategy) when compared to the A* search with the H1 heuristic. Uniform Cost Search expands nodes in order of their optimal path cost. [AIMA] always finds the optimal solution unless it gets stuck in an infinite loop. A* Search H1(heuristic based) is similar in that expands the node of the lowest f-cost for expansion, fans out from the start node. In this case adding nodes in concentric costs [Norvig 97]. While powerful it is not the answer to all searching needs in that the number of states

within the goal contour search space is still exponential in the length of solution. [Norvig 98] Computation is not the major drawback of A* often runs out of memory before it runs out of time. is considered complete and terminates to produce the best solution. Problem 2 portrays a search space small enough that a non-heuristic search is the best choice but closely rivaled by a heuristic based search.

Air Cargo Problem 3:

This problem has 2 airplanes, 4 airports, and and 4 pieces of cargo. This problem creates the largest search space of the three problems. The initial state and goal is defined as:

```
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))
```

After running both my the uniformed and heuristic searches against this problem I found the following results summarized in the table:

Air Cargo Problem 3

	Time (sec)	Plan Length	Expansions	Goal Tests	New Nodes
Breadth First Search	81.911	12	14663	18098	129631
Breadth First Tree Search			**Timeout!**		
Depth First Graph Search	2.584	596	627	628	5176
Depth Limited Search			**Timeout!**		
Uniform Cost Search	298.425	12	18223	18225	159618
Recursive Best First Search H1			**Timeout!**		
Greedy Best First Search H1	83.863	22	5578	5580	49150
A* Search H1	315.739	12	18223	18225	159618
A* Search H_ignore preconditions	71.521	12	5118	5120	45650
A* Search H_PG_levelsum			**Timeout!**		

From analyzing Air Cargo Problem 3 table we can see **A* Search H_ignore_preconditions** is the best choice of algorithm. The ignore preconditions heuristic relaxes the search constraints by allowing every action to happen applicable in every state. This search ran in the least amount of time and produce the optimal path in 12 steps :

```
1. Load(C2, P2, JFK) 2.
Fly(P2, JFK, ORD) 3.
Load(C4, P2, ORD)
4. Fly(P2, ORD, SFO)
5. Unload(C4, P2, SFO)
6. Load(C1, P1, SFO) 7. Fly(P1, SFO, ATL) 8. Load(C3, P1, ATL)
9. Fly(P1, ATL, JFK)
10.Unload(C3, P1, JFK) 11.Unload(C2,
P2, SFO) 12.Unload(C1, P1, JFK)
```

Interestingly this just beat Breadth First Search in terms of running time performance, however the number of expansions was significantly less. This is due to additional computation cost of the heuristics based search adds an increased amount of running per expansion. However this increased complexity also creates guidance requiring fewer expansions, in turn creating superior results. I believe as the complexity and an expansion of the search space increases heuristic based A* search methods would continually produce better results than the nonheuristic based search methods.

Reference

Norvig, Peter. Russell, Stuart J. Artificial Intelligence A Model Approach: Third Edition. Pearson Education 2015.