Heuristic Analysis

Project implements a Planning Search on Air Cargo problem using breadth first search, breadth first tree search, depth first graph search, depth limited search, uniform cost search, recursive best first search, greedy best first graph search, astar search with different heuristics.

Air Cargo Action Schema:

```
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))
```

Air Cargo Problem 1:

Two cargos and two planes were given at two respective airports(SFO & JFK). Goal is to exchange cargos between two airports.

```
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))
```

I found following results:

Air Cargo Problem 1

Search Algoritms	Time(in sec)	Plan Length	Expansion	Goal Test	New Nodes
Breadth first search	0.15	6	43	56	180
Breadth first tree search	4.33	6	1458	1459	5960
Depth first graph search	0.04	12	12	13	48
Depth limited search	0.41	50	101	271	414
Uniform cost search	0.17	6	55	57	224
Recursive best first search h_1	12.14	6	4229	4230	17029
Greedy best first graph search h_1	0.01	6	7	9	28
A* h_1	0.14	6	55	57	224
A* h_ignore_preconditions	0.11	6	41	43	170
A* h_pg_levelsum	52.98	6	47	49	193

Following is the optimal path:-

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)

Although it is a small problem, but clearly **Greedy best first graph search h_1 algorithm** has outperformed other algorithms both in terms of execution time & node expansion. Using a greedy algorithm, expand the first successor of the parent. After a successor is generated:

1.If the successor's heuristic is better than its parent, the successor is set at the front of the queue (with the parent reinserted directly behind it), and the loop restarts.

2.Else, the successor is inserted into the queue (in a location determined by its heuristic value). The procedure will evaluate the remaining successors (if any) of the parent.

Since, this algorithm is very much goal oriented thus it has required very few fluent.

Depth first search comes 2nd because while finding a path to the goal it performs some backtracking which adds some extra nodes expansion. But it wasn't able to deliver optimal solution.

Air Cargo Problem 2:

Three Cargos & three planes are given at three airports respectively. The goal is to distribute the cargos as given below.

```
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))
```

I found following results:

Air Cargo Problem 2					
Search Algoritms	**Time(in sec)**	**Plan Length**	**Expansion**	**Goal Test**	**New Nodes**
Breadth first search	50.87	9	3343	4609	30509
Breadth first tree search	Timeout				
Depth first graph search	10.04	575	582	583	5211
Depth limited search	Timeout				
Uniform cost search	161.92	9	4853	4855	44041
Recursive best first search h_1	Timeout				
Greedy best first graph search h_1	25.05	17	998	1000	8982
A* h_1	165.57	9	4853	4855	44041
A* h_ignore_preconditions	44.99	9	1506	1508	13820
A* h_pg_levelsum	Timeout				

Following is the optimal path:-

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)

After analyzing the result above, **A* h_ignore_preconditions** is the best choice on the basis of execution time and optimal path. Although, Depth first graph search & Greedy best first graph search h_1 were better but only in terms of execution time. They both were unable to find optimal path to the problem.

A* is an informed search algorithm, or a best-first search, meaning that it solves problems by searching among all possible paths to the solution (goal) for the one that incurs the smallest cost (least distance travelled, shortest time, etc.), and among these paths it first considers the ones that appear to lead most quickly to the solution. It is formulated in terms of weighted graphs: starting from a specific node of a graph, it constructs a tree of paths starting from that node, expanding paths one step at a time, until one of its paths ends at the predetermined goal node.

At each iteration of its main loop, A* needs to determine which of its partial paths to expand into one or more longer paths. It does so based on an estimate of the cost (total weight) still to go to the goal node. Specifically, A* selects the path that minimizes

f(n)=g(n)+h(n)

where n is the last node on the path, g(n) is the cost of the path from the start node to n, and h(n) is a heuristic that estimates the cost of the cheapest path from n to the goal. The heuristic is problem-specific. For the algorithm to find the actual shortest path, the heuristic function must be admissible, meaning that it never overestimates the actual cost to get to the nearest goal node.

Computation is not the major drawback of A* often runs out of memory before it runs out of time. It is considered complete and terminates to produce the best solution.

Air Cargo Problem 3:

Four cargos & two planes are given at four airports. The goal is to distribute the cargos as following.

```
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))
```

I found following results:

Air Cargo Problem 3					
Search Algoritms	**Time(in sec)**	**Plan Length**	**Expansion**	**Goal Test**	**New Nodes**
Breadth first search	404.26	12	14663	18098	129631
Breadth first tree search	Timeout				
Depth first graph search	14.08	596	627	628	5176
Depth limited search	Timeout				
Uniform cost search	Timeout				
Recursive best first search h_1	Timeout				
Greedy best first graph search h_1	244.15	22	5578	5580	49150
A* h_1	Timeout				
A* h_ignore_preconditions	347.31	12	5118	5120	45650
A* h_pg_levelsum	Timeout				

Following is the optimal path:-

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)

After analyzing the result above, **A* h_ignore_preconditions** is again the best choice on the basis of execution time and optimal path. The heuristics is the cause for increased amount of execution time. The additional complexity also helps to follow a directed approach towards the goal. I believe A* with heuristic will continue to perform better than non-heuristic algorithms as problems become more complex.

References

1. https://en.wikipedia.org/wiki/A*_search_algorithm
2. https://en.wikipedia.org/wiki/Best-first_search
3. *Artificial Intelligence: A Modern Approach* by Norvig and Russell
4. http://stackoverflow.com/questions/8374308/is-greedy-best-first-search-different-from-best-first-search