Heuristic Analysis

Below is a basic outline of the problem and the metrics for running uninformed planning searches and A * planning searches with heuristics on *air_cargo_p1*, *air_cargo_p2*, and *air_cargo_p3* problems. I ran *breadth_first_search*, *depth_first_graph_search* and *uniform_cost_search* for the uninformed searches and used A * with both the *h_ignore_preconditions* and *h_pg_levelsum* heuristics for the heuristic searches. The tables below Compare and contrast the expansions, goal tests, new nodes, plan length, run time and optimality of each search strategy.

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))
```

Problem 1 initial state and goal:

```
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 initial state and goal:

```
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 initial state and goal:

```
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))
```

Optimal Plan Lengths

Problem 1

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

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

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)

Uninformed planning searches

Air Cargo Problem 1

Search	Expansions	Goal Tests	New Nodes	Plan length	Time (s)	Optimality
BFS	43	56	180	6	0.033	Yes
DFS	12	13	48	12	0.010	No
UCS	55	57	224	6	0.036	Yes

Air Cargo Problem 2

Search	Expansions	Goal Tests	New Nodes	Plan length	Time (s)	Optimality
BFS	3343	4609	30509	9	7.859	Yes
DFS	582	583	5211	575	2.878	No
UCS	4852	4854	44030	9	11.488	Yes

Air Cargo Problem 3

Search	Expansions	Goal Tests	New Nodes	Plan length	Time (s)	Optimality
BFS	14663	18098	129631	12	40.556	Yes
DFS	627	628	5176	596	3.146	No
UCS	18234	18236	159707	12	50.581	Yes

Conclusion

If finding an optimal path length is important, which I image it would be in most planning problems, then I would recommended BFS for an uninformed search strategy. Unlike DFS, its an optimal solution and it will always return the most optimal result, if one exists. While DFS generally traversed substantially fewer nodes, it seemed to be way off in regards to discovering an optimal solution. BFS also ran faster and used less memory than UCS.

A planning searches using the heuristics*

The tables below illustrate running A * searches with both the *h_ignore_preconditions* and *h_pg_levelsum* heuristics for problems *air_cargo_p1*, *air_cargo_p2*, and *air_cargo_p3.*

Air Cargo Problem 1

Search	Expansions	Goal Tests	New Nodes	Plan length	Time (s)	Optimality
HIP	41	43	170	6	0.035	Yes
h_levelsum	39	41	158	6	0.791	Yes

Air Cargo Problem 2

Search	Expansions	Goal Tests	New Nodes	Plan length	Time (s)	Optimality
HIP	1450	1452	13303	9	3.574	Yes
h_levelsum	1129	1131	10232	9	282.455	Yes

Air Cargo Problem 3

Search	Expansions	Goal Tests	New Nodes	Plan length	Time (s)	Optimality
HIP	5040	5042	44944	12	14.862	Yes
h_levelsum	---	---	---	---	---	---

~ H_pg_levelsum did not finishing within the 10 minutes limit.

The *H_ignore_preconditions* heuristic ran substantially quicker than the *h_pg_levelsum* heuristic for problem 2 (~90 times).

Conclusion and Recommendation

My search recommendation is to use A * search with the *h_ignore_preconditions* heuristic for all problems. The best performing uninformed search strategy was BFS and A * search with the *h_ignore_preconditions* heuristic ran faster and traversed fewer nodes on every problem, with the exception of problem 1, where BFS was only 0.002 seconds faster.

the ***ignore preconditions heuristic*** works by dropping all preconditions from actions which ends up adding edges to the graph. This relaxes the problem and essentially makes it easier to solve. Every action becomes applicable in every state, and any single goal fluent can be achieved in one step. For many problems an accurate heuristic is obtained by considering (1) and ignoring (2). First, we relax the actions by removing all preconditions and all effects except those that are literals in the goal. Then, we count the minimum number of actions required such that the union of those actions' effects satisfies the goal (Norvig and Russell, 2009).

After examining the results displayed in the tables above, it it evident that there are significant advantages to using informed search strategies with heuristics as opposed to using an uninformed search strategy. These advantages range from speed gains to decreases in required memory. I imagine the custom heuristics strategy becomes even more advantageous as problems grow in size and complexity. For example, if you were dealing with a very large graph you could optimize a heuristic for memory management, whereas an uninformed BFS will run the same on both small and large graphs.

References

Artificial Intelligence: A Modern Approach (2009), Peter Norvig, Stuart J. Russell