Heuristic Analysis of the Air Cargo Problem

In the book Artificial Intelligence: A modern Approach, 3rd Edition, by Stuart J. Russell and Peter Norvig, algorithms for classical planning are discussed in Chapter 10. Several planning problems are discussed, including the air cargo problem which seeks to find the most optimal set of actions that will get cargo to the correct destination airport.

Below, the air-cargo problem is first setup by specifying the schema for each allowed action (load, unload, fly). The schema includes the pre conditions required to execute the action as well as the results of the action. Then, for three increasingly complicated scenarios, the initial states and the goals are described.

- 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))
```

Three uninformed algorithms (Breadth First, Depth First, Uniform Cost) are compared as well as two informed algorithms (A* with Ignore Preconditions, A* with Level-Sum).

All the algorithms were able to find optimal solutions except Depth First. The fastest solution for each problem is listed in the following table:

Problem 1 Breadth First	Problem 2 A* with Ignore Preconditions	Problem 3 A* with Ignore Preconditions
Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C1, P1, SFO)
Load(C2, P2, JFK)	Fly(P1, SFO, JFK)	Fly(P1, SFO, ATL)
Fly(P1, SFO, JFK)	Unload(C1, P1, JFK)	Load(C3, P1, ATL)
Fly(P2, JFK, SFO)	Load(C2, P2, JFK)	Fly(P1, ATL, JFK)
Unload(C1, P1, JFK)	Fly(P2, JFK, SFO)	Unload(C1, P1, JFK)
Unload(C2, P2, SFO)	Unload(C2, P2, SFO)	Load(C2, P2, JFK)
	Load(C3, P3, ATL)	Fly(P2, JFK, ORD)
	Fly(P3, ATL, SFO)	Load(C4, P2, ORD)
	Unload(C3, P3, SFO)	Fly(P2, ORD, SFO)
		Unload(C2, P2, SFO)
		Unload(C3, P1, JFK)
		Unload(C4, P2, SFO)

Comparison of the algorithms for the three problems are in the tables below. The first three algorithms in each table are uninformed and the last two are informed by automatic heuristics.

Problem 1:

Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Optimal	Time (msec)
Breadth First	43	56	180	6	Yes	22
Depth First	21	22	84	20	No	10
Uniform Cost	55	57	224	6	Yes	26
A* with Ignore Preconditions	41	43	170	6	Yes	35
A* with Level-Sum	11	13	50	6	Yes	637

Problem 2:

Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Optimal	Time (sec)
Breadth First	3343	4609	30509	9	Yes	9.8
Depth First	624	625	5602	619	No	2.4
Uniform Cost	4853	4855	44041	9	Yes	8.2
A* with Ignore Preconditions	1450	1452	13303	9	Yes	3.8
A* with Level-Sum	86	88	841	9	Yes	58.9

Problem 3:

Algorithm	Expansions	Goal Tests	New Nodes	Plan Length	Optimal	Time (sec)
Breadth First	14663	18098	129631	12	Yes	70.2
Depth First	408	409	3364	392	No	1.3
Uniform Cost	18236	18238	159726	12	Yes	36.6
A* with Ignore Preconditions	5038	5040	44926	12	Yes	14.5
A* with Level-Sum	314	316	2894	12	Yes	316.3

As the tables show, Depth First is a poor algorithm in all the problems as it returns the first solution found which most likely is not optimal.

The results confirm Russell and Norvig statement (p. 382) that searches are not efficient unless they use heuristics to cut down on the number of paths that are explored.

And as we learned from the Build a Game Playing Agent project, it can be faster to use a simpler heuristic (such as A* with Ignore Preconditions) , that computes fast, over a complicated but smarter heuristic (such as A* with Level-Sum) that requires much time to compute.