Analysis of search methods and heuristics for the air cargo problems

An analysis by Kris Roosen as requested by Udacity for the 3rd project in the AIND program.

Problem description

The goal of this paper is to analyze the performance of several search algorithms on a problem called "Air cargo problem" which is defined within the 3rd project of the 1st term of the *Artificial Intelligence Nanodegree* at Udacity. The setup was borrowed from AIMA chapter 10 (3rd ed.) by Russell and Norvig. Some of the tested search algorithms are uninformed and some are informed by a heuristic function.

The algorithms that are tested are *Breadth First Search* (BFS), *Depth First Search* (DFS), *Uniform Cost Search* (UCS) and *A-star Search* (A*).

The heuristic functions are called *ignore_preconditions* (IP) and *levelsum* (LS) and will be described in more detail below.

Each problem consists of a number of airports, planes and pieces of cargo. A problem is described using classical planning and an example is presented in the following picture:

Heuristic functions

ignore_preconditions: this function is defined by relaxing the given problem. For this case, all preconditions for all actions are removed. This means that any action can be performed in any situation and still achieve its described effects. The presented implementation doesn't take into account that an action could result in multiple goals being achieved. This choice was made to speed up the searching. All the test problems are defined such that any action can only achieve a single goal at a time. This means that the heuristic function is admissible, although applicable for a smaller range of problems.

Levelsum: the problem is first defined as a planning graph. The advantage of a graph is that problems with infinite search trees can be presented in a compact yet complete way. The graph is presented in several levels, alternating between state levels and action levels (S0, A0, S1, A1, …). More information about this can be found in the AIMA book. The LS heuristic is calculated by summing the first levels in which one of the goals can be achieved for all goals. This works best if the goals are independent. For the presented problems, this function is **not** admissible. However, as the AIMA book explains, this is usually ignored with still pretty good results. As you can see in the appendix, this function still resulted in an optimal solution for each of the problems.

Results

In the table below, the most useful indicators and their values are presented for each of the tests. In the appendix at the end of this document you can find the returned paths.

Problem 1	**BFS**	**DFS**	**UCS**	**A* IP**	**A* LS**
Expansions	43	21	55	41	11
Goal tests	56	22	57	43	13
New nodes	180	84	224	170	50
Run time [s]	0,028	0,013	0,037	0,024	2,039
Plan length	6	20	6	6	6
Optimal?	Yes	No	Yes	Yes	No

Problem 2	**BFS**	**DFS**	**UCS**	**A* IP**	**A* LS**
Expansions	3343	624	4852	1506	86
Goal tests	4609	625	4854	1508	88
New nodes	30509	5602	44030	13820	841
Run time [s]	13,208	3,274	46,807	14,131	208,109
Plan length	9	619	9	9	9
Optimal?	Yes	No	Yes	Yes	No

Problem 3	**BFS**	**DFS**	**UCS**	**A* IP**	**A* LS**
Expansions	14663	408	18236	5118	404
Goal tests	18098	409	18238	5120	406
New nodes	129631	3364	159726	45650	3718
Run time [s]	99,618	1,631	448,872	82,912	1368,381
Plan length	12	392	12	12	12
Optimal?	Yes	No	Yes	Yes	No

Discussion

Optimality

BFS and UCS provide an optimal solution. This is implicitly true by the definition of the algorithms.

DFS doesn't usually result in an optimal solution for the same reason.

A* provides an optimal solution if the heuristic function is admissible. This is true for the IP function but **not** for the LS function for the presented problems, although it returns an optimal plan for the test cases.

Expansions, goal tests, new nodes, run time

In the results table, it clearly shows that DFS outperforms most other algorithms, although it doesn't provide an optimal solution. This can be explained by the branching factor. It is small so that the chance of reaching a goal state is relatively high.

The A* LS algorithm results less expansions, goal tests and nodes created, however a lot more time is spent calculating the heuristic value. This can probably be sped up by implementing it more efficiently. The heuristic function creates a new graph at every state which is very expensive.

Conclusion

When an optimal solution is required and the limiting factor is running time, I recommend using the A* algorithm with the *ignore_preconditions* heuristic function.

APPENDIX – SOLUTION PATHS

PROBLEM 1

BFS *Plan length: 6*

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

DFS *Plan length: 20*

Fly(P1, SFO, JFK) → Fly(P2, JFK, SFO) → Load(C2, P1, JFK) → ... → Fly(P1, JFK, SFO) → Fly(P2, SFO, JFK) → Unload(C2, P1, SFO)

UCS *Plan length: 6*

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

A* search with the ignore_preconditions heuristic *Plan length: 6*

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

A* search with h_levelsum heuristic *Plan length: 6*

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

PROBLEM 2

BFS *Plan length: 9*

Load(C1, P1, SFO) → Load(C2, P2, JFK) → 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)

DFS *Plan length: 619*

Fly(P3, ATL, SFO) → Fly(P1, SFO, ATL) → Fly(P3, SFO, JFK) → … → Fly(P1, ATL, JFK) → Fly(P3, SFO, JFK) → Unload(C3, P2, SFO)

UCS *Plan length: 9*

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

A* search with ignore_preconditions heuristic *Plan length: 9*

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

A* search with h_levelsum heuristic *Plan length: 9*

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

PROBLEM 3

BFS *Plan length: 12*

Load(C1, P1, SFO) → Load(C2, P2, JFK) → 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)

DFS *Plan length: 392*

Fly(P1, SFO, ORD) → Fly(P2, JFK, ORD) → Fly(P1, ORD, ATL) → … → Fly(P2, ATL, JFK) → Fly(P1, ATL, JFK) → Unload(C3, P1, JFK)

UCS *Plan length: 12*

Load(C1, P1, SFO) → Load(C2, P2, JFK) → Fly(P1, SFO, ATL) → Load(C3, P1, ATL) → Fly(P2, JFK, ORD) → Load(C4, P2, ORD) → Fly(P2, ORD, SFO) → Fly(P1, ATL, JFK) → Unload(C4, P2, SFO) → Unload(C3, P1, JFK) → Unload(C2, P2, SFO) → Unload(C1, P1, JFK)

A* search with ignore_preconditions heuristic *Plan length: 12*

Load(C2, P2, JFK) → Fly(P2, JFK, ORD) → Load(C4, P2, ORD) → Fly(P2, ORD, SFO) → Unload(C4, P2, SFO) → Load(C1, P1, SFO) → Fly(P1, SFO, ATL) → Load(C3, P1, ATL) → Fly(P1, ATL, JFK) → Unload(C3, P1, JFK) → Unload(C2, P2, SFO) → Unload(C1, P1, JFK)

A* search with h_levelsum heuristic *Plan length: 12*

Load(C2, P2, JFK) → Fly(P2, JFK, ORD) → Load(C4, P2, ORD) → Fly(P2, ORD, SFO) → Load(C1, P1, SFO) → Fly(P1, SFO, ATL) → Load(C3, P1, ATL) → Fly(P1, ATL, JFK) → Unload(C4, P2, SFO) → Unload(C3, P1, JFK) → Unload(C2, P2, SFO) → Unload(C1, P1, JFK)