Project 3. Heuristic analysis

The following provides a comparison between a number of search algorithms applied to planning problems. The algorithms have all been tested on three planning problems of increasing complexity, all in the air cargo domain. The tests have been carried out in similar conditions, on an HP EliteBook with an Intel(R) Core(TM) i7-4600U 2.10GHz CPU and 16 GB of RAM, running Ubuntu 16.04.

The performance of each algorithm is reported in terms of number of node expansions required, number of goal tests performed, number of new nodes generated, length of the plan found, and time elapsed (in seconds). The optimality of the solution is assessed in terms of the plan length.

Uninformed search for planning problems

The table below shows, for each of our three problems, the performance of three uninformed search algorithms: breadth-first search (BFS), depth-first search (DFS), and uniform-cost search (UCS). For each criterion considered, the best result is highlighted in bold.

Table 1: Comparative performance of BFS, DFS, and UCS on the three planning problems.

Prob.	Algo.	Expansions	Goal tests	New nodes	Plan length	Time (sec)
1	BFS	43	56	180	**6**	0.036
1	DFS	**12**	**13**	**48**	12	**0.009**
1	UCS	55	57	224	**6**	0.042
2	BFS	3343	4609	30509	**9**	15.272
2	DFS	**582**	**583**	**5211**	575	**3.370**
2	UCS	4852	4854	44030	**9**	13.828
3	BFS	14663	18098	129631	**12**	111.582
3	DFS	**627**	**628**	**5176**	596	**3.655**
3	UCS	18235	18237	159716	**12**	62.841

These results show that DFS consistently fares best on all counts except on the length of the plan found. Indeed, there is a striking constrast between the time taken and the number of expansions and tests performed by DFS on the one hand, and BFS and UCS on the other. The size of the problem also seems to have a considerable effect on BFS and UCS: with each increase in size (from problem 1 to problem 2, and from problem 2 to problem 3), these two algorithms take more and more time, and the expand an increasing number of nodes. In

contrast, DFS' performance is roughly the same for problem 2 and problem 3.

Nonetheless, despite its speed, DFS also consistently finds solutions which is are far from optimal (as defined above), while BFS and UCS are both able to find the optimal (i.e. shortest) solution: DFS' plan is twice as long as that of BFS and UCS for problem 1, over 60 times longer for problem 2, and about 50 times longer for problem 3. This result is not surprising, given the way DFS works: **DFS is suboptimal**, since it always explores the entire left subtree before proceding to the right; as a result, it may find a goal node which is located deeper than another goal found more to the right in the tree. Conversely, both **BFS and UCS are optimal**: they are able to find the sortest path to a goal node.

Informed (heuristic) search for planning problems

The following round of experiments tests the A* algorithm on our planning problems, using two real heuristics (ignore preconditions and level sum) and no heuristic (`h_1` - the baseline, for reference).

Table 2: Comparative performance of two heuristics and no heuristic with A* on the three planning problems.

Prob.	Heuristic	Expansions	Goal tests	New nodes	Plan length	Time (sec)
1	none	55	57	224	**6**	0.044
1	ignore precond	41	43	170	**6**	**0.041**
1	level sum	**39**	**41**	**158**	**6**	0.780
2	none	4852	4854	44030	**9**	14.645
2	ignore precond	1450	1452	13303	**9**	**4.791**
2	level sum	**1129**	**1131**	**10232**	**9**	279.071
3	none	18235	18237	159716	**12**	59.871
3	ignore precond	4951	4953	44051	**12**	**18.477**
3	level sum	**4322**	**4324**	**38475**	13	1771.243

Running A* with the `h_1` heuristic, i.e. no heuristic at all, amounts to running a UCS on the problem. The gains in performance under all criteria, including speed of execution, are impressive for the ignore-preconditions heuristic. However, while level sum lowers even further the number of expansions and goal tests, this comes at the cost of speed: the time needed for A* to find a solution using the level-sum heuristic is orders of magnitude longer than that of A* with the ignore-preconditions heuristic or with no heuristic at all. For problem 3, A* with level sum took almost 30 minutes on my machine, while with ignore preconditions it only took 18 seconds. This increase in time follows from the fact that the level

sum heuristic uses a planning graph, which is built in time polynomial in the size of the problem.

It also appears that **A* with level sum is, in fact, not optimal**. For problem 3 it returns a solution which is not the shortest path to a goal. On the other hand, A* with ignore preconditions does seem to be optimal. (Both time and plan length for A* with level sum on problem 3 have been confirmed through a few more runs.)

In conclusion, level sum scores the best results when it comes to keeping expanded nodes and goal tests to a minimum, but has the longest time and is not optimal; **A* with ignore preconditions has the best time and is optimal**.

Optimal solutions

An optimal set of solutions is given below:

Problem 1

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

Problem 2

Load(C1, P1, SFO)

Load(C2, P2, JFK)

Fly(P1, SFO, JFK)

Fly(P2, JFK, SFO)

Load(C3, P3, ATL)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

Unload(C1, P1, JFK)

Unload(C2, P2, SFO)

Problem 3

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(C1, P1, JFK)

Unload(C2, P2, SFO)