Planning Search Performance Analysis

Uninformed Search

For each problem `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3`, I ran the problem solver with the following search algorithms:

- Breadth First Search (BFS)
- Depth First Graph Search (DFGS)
- Uniform Cost Search (UCS)

Node Expansions

DFGS produced the fewest node expansions across all three problem sets.

Node Expansions			
Problem	BFS	DFGS	UCS
air_cargo_p1	43	21	55
air_cargo_p2	1923	82	2724
air_cargo_p3	14663	408	18223

Goal Tests

For the larger problem spaces, DFGS required far fewer goal tests. For the smallest problem (`air_cargo_p1`) however, BFS and UCS required fewer and were closely matched.

Goal Tests			
Problem	BFS	DFGS	UCS
air_cargo_p1	56	84	57
air_cargo_p2	2672	83	2726
air_cargo_p3	18098	409	18225

Time Elapsed

DFGS was consistently the fastest to complete each problem - averaging a 14.5% speed increase compared to BFS and 13.% compared with UCS.

Time Elapsed			
Problem	BFS	DFGS	UCS
air_cargo_p1	0.031	0.012	0.034
air_cargo_p2	5.431	0.147	5.250
air_cargo_p3	84.408	1.514	44.747

Solution Optimality

I am judging the optimality of each solution based on the **total plan length**. All algorithms are complete as they will always find a solution if one exists, so for the Air Cargo Problem the optimal plan minimizes overall length - it is cheaper, faster and more energy efficient to transport the cargo in as few steps as possible.

Plan Length			
Problem	BFS	DFGS	UCS
air_cargo_p1	6	20	6
air_cargo_p2	9	77	9
air_cargo_p3	12	392	12

BFS and UCS consistently produce the lowest (also equal) plan lengths. DFGS produces **far longer** plans, making it much less optimal.

Informed Search

To improve both performance and results of the planning algorithm, A* search can be used in order to utilise heuristic evaluation of problem states. Furthermore, a **Planning Graph** can be implemented to provide a polynomial-size approximation of all possible paths in the planning problem which can be used to generate automated, admissible heuristics. This means that the heuristics generated are not specific to each problem domain. Two different heuristic functions were used:

- Ignore Preconditions Heuristic (ignore_preconditions)
    - Estimates the minimum number of actions required to reach a goal state by ignoring the preconditions for actions to be executed.
- Planning Graph Level Sum Heuristic (pg_levelsum)
    - Uses a planning graph to estimate sum of all actions required to reach a goal state.

Node Expansions

pg_levelsum produced the fewest node expansions by a significant margin across all three problem sets.

Node Expansions		
Problem	ignore_preconditions	pg_levelsum
air_cargo_p1	41	11
air_cargo_p2	876	238
air_cargo_p3	8765040	325

Goal Tests

Pg_levelsum required the fewest goal tests, again, by a significant margin.

Goal Tests		
Problem	ignore_preconditions	pg_levelsum
air_cargo_p1	43	13
air_cargo_p2	878	240
air_cargo_p3	5042	327

Time Elapsed

Ignore_preconditions heuristic was extremely quick in comparison to pg_levelsum, solving `air_cargo_p1`, `air_cargo_p2`, and `air_cargo_p3` approximately **27, 79 and 55 times faster** respectively.

Time Elapsed		
Problem	ignore_preconditions	pg_levelsum
air_cargo_p1	0.037	1.011
air_cargo_p2	2.348	186.492
air_cargo_p3	15.842	826.310

Solution Optimality

Both algorithms produced the optimal plan lengths for all 3 problems.

Plan Length		
Problem	ignore_preconditions	pg_levelsum
air_cargo_p1	6	6
air_cargo_p2	9	9
air_cargo_p3	12	12

Discussion

Each uninformed search produced a solution to all of the given problems. DFS and BFS produce the most optimal solutions, however are computationally and spatially inefficient. DFGS on the other hand is much faster, however produces nonoptimal solutions with large plan lengths. This is because DFGS explores nodes as deeply as possible in the search tree and does not take into account whether a node is comparatively better than another. For example, whilst searching down a path, it may pass the level of the graph at which a goal state is present on the path to its right.

Informed searches also produced solutions to all given problems. Furthermore, they continued to produce optimal results with increased problem complexity. For example, with `air_cargo_p3`, BFS took 84.41s while A* search with ignore_preconditions heuristic took only 15.84s. However, although finding optimal solutions, the pg_levelsum heuristic performed poorly taking far longer to complete than any other strategy. This is because it is complex to compute, requiring a planning graph to be constructed and testing for the presence of each goal state in each level of the graph. Each time the heuristic is called, a new planning graph is constructed with $O(n(a+l)^2)$ time and space complexity with n levels, a actions and l literals [1].

For smaller planning problems, the results suggest that a **breadth-first search** approach is best - producing optimal solutions quickly. However, as problem complexity increases, implementing an informed, heuristic search will be beneficial such as **A* search with ignore_preconditions** heuristic.

[1] "Artificial Intelligence: A Modern Approach." pg. 381 http://aima.cs.berkeley.edu/.

Optimal Plans

Optimal sequences of actions produced by the recommended search algorithms:

Problem	Breadth First Search	A* ignore_preconditions
air_cargo_1	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)
air_cargo_2	Load(C1, P1, SFO) Load(C2, P2, JFK) Fly(P2, JFK, ATL) Load(C3, P2, ATL) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P2, ATL, SFO) Unload(C2, P2, SFO) Unload(C3, P2, SFO)	Load(C2, P2, JFK) Fly(P2, JFK, ATL) Load(C3, P2, ATL) Fly(P2, ATL, SFO) Unload(C3, P2, SFO) Unload(C2, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)
air_cargo_3	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)	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)