Heuristic analysis on planning search

1. Optimal plan for problem 1, 2, 3

Problem 1	Problem 2	Problem 3
Plan length: 6	Plan length: 9	Plan length: 12
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) 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)	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)

2. Non-heuristic (uninformed) search result metrics

We compared the speed (e.g., execution time in seconds), memory consumption (e.g., node expansions when searching), and optimality (Yes, if a solution with optimal length is found; No, otherwise). The results are as below,

- Problem 1

Strategy	Optimality	Path length	Execution time (s)	Node expansions
breadth_first_search	YES	6	0.26	43
depth_first_graph_search	NO	20	0.17	21
uniform_cost_search	YES	6	0.46	55

- Problem 2

Strategy	Optimality	Path length	Execution time (s)	Node expansions
breadth_first_search	YES	9	96.69	3343
depth_first_graph_search	NO	619	18.91	624
uniform_cost_search	YES	9	175.92	4852

- Problem 3

Strategy	Optimality	Path length	Execution time (s)	Node expansions
breadth_first_search	YES	12	535.68	14663
depth_first_graph_search	NO	392	13.93	408
uniform_cost_search	YES	12	816.84	18235

Analysis

As far as optimality, Both **Breadth First Search** and **Uniformed Cost Search** are the ideal strategies. And **Breadth First Search** is better than **Uniformed Cost Search** when considering the time elapsed and memory cost. We can also find out that **Depth First Graph Search** is not an optimal strategy as its path length is larger than the other twos. But it is much more efficient to find a solution while this solution may be not the shortest one. These features can be explained with the rigorous analysis which has been deduced in *AIMA* 3rd in section 3.4.7 as below,

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: [a] complete if b is finite; [b] complete if step costs $\geq \epsilon$ for positive ϵ; [c] optimal if step costs are all identical; [d] if both directions use breadth-first search.

3. Heuristic (informed) search result metrics

■ Problem 1

Strategy	Optimality	Path length	Execution time (s)	Node expansions
astar_search with h_ignore_preconditions	YES	6	0.36	41
astar_search with h_pg_levelsum	YES	6	0.89	11

■ Problem 2

Strategy	Optimality	Path length	Execution time (s)	Node expansions
astar_search with h_ignore_preconditions	YES	9	56.83	1450
astar_search with h_pg_levelsum	YES	9	80.33	86

■ Problem 3

Strategy	Optimality	Path length	Execution time (s)	Node expansions
astar_search with h_ignore_preconditions	YES	12	248.30	5040
astar_search with h_pg_levelsum	YES	12	405.72	316

Analysis

We conducted two informed search algorithms based on **A-Star Search** with **ignore-preconditions** and **sum-level**. The experiments show that both of them reached out optimality. But here we need to point out that we use **greedy set-cover** algorithm to implement **ignore-preconditions**. Thus, we get an inadmissible heuristic search strategy which means we cannot guarantee the optimality for every path found problem, the same to **sum-level** which is not always optimal neither. Besides the complexity of heuristic value computation is not trivial as the complexity of problem is up. the **sum-level** cost much less memory while paid much more execution time for complex heuristic computation. This feature leads us to make a trade-off between time complexity and memory cost which depends on our specific domain.

4. Informed vs. uninformed search

We compare **Breadth First Search** (BFS) with **A-Star search with ignore-preconditions** (AI) since they are the best method among our experiments in uninformed and informed respectively.

AI/BFS (%)	Path length	Execution time (s)	Node expansions
Problem 1	100.0	138.5	95.3
Problem 2	100.0	58.8	43.4
Problem 3	100.0	46.4	34.4

Analysis and *Conclusion*

In problem 2 or 3, AI is always the better choice no matter in optimality or time/space cost. But when the complexity of problem is not very high (e.g., for problem 1), we are inclined to choose BFS instead, as BFS cost less time to find an optimal solution. Another thing needs to be clarified is that if we want to solve a problem guaranteed perfectly we should also use BFS.