Optimal Sequence of Action

Problem 1	**Problem 2**	**Problem 3**
Path length: 6 Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	Path length: 9 Load(C1, P1, SFO) Load(C2, P2, JFK) Load(C3, P3, ATL) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P3, ATL, SFO) Unload(C3, P3, SFO)	Path length: 12 Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Load(C2, P1, JFK) Unload(C1, P1, JFK) Unload(C3, P1, JFK) Fly(P1, JFK, ORD) Load(C4, P1, ORD) Fly(P1, ORD, SFO) Unload(C2, P1, SFO) Unload(C4, P1, SFO)

Non-heuristic Search

Problem 1

Algorithm	Plan length	Expansions	Elapse Time
Breadth First Search	**6**	43	0.0366s
Breadth First Tree Search	**6**	1458	1.1183s
Depth First Graph Search	12	12	0.0095s
Depth Limited Search	50	101	0.1086s
Uniform Cost Search	**6**	55	0.0440s
Recursive Best First Search	**6**	4229	3.2850s
Greedy Best First Graph Search	**6**	**7**	**0.0058s**

Concerning problem 1, we could see that Greedy Best First Graph Search is the best strategy.

Problem 2

Algorithm	Plan length	Expansions	Elapse Time
Breadth First Search	**9**	3346	15.3344s
Breadth First Tree Search	--	--	> 10 min
Depth First Graph Search	1085	1124	8.3269s
Depth Limited Search	--	--	> 10 min
Uniform Cost Search	**9**	4853	13.3724s
Recursive Best First Search	--	--	> 10 min
Greedy Best First Graph Search	17	**998**	**2.7196s**

Concerning problem 2, only Breath First Search and Uniform Cost Search give the optimal plan, while Greedy Best First Graph Search is the fastest and use the least memory.

Problem 3

Algorithm	Plan length	Expansions	Elapse Time
Breadth First Search	**12**	14902	103.3441s
Breadth First Tree Search	--	--	> 10 min
Depth First Graph Search	744	**4185**	**21.5048s**
Depth Limited Search	--	--	> 10 min
Uniform Cost Search	**12**	18541	60.2664s
Recursive Best First Search	--	--	> 10 min
Greedy Best First Graph Search	22	8174	26.9854s

Concerning problem 3, only Breath First Search and Uniform Cost Search give the optimal plan, while Depth First Graph Search is the fastest and use the least memory.

Analysis

In general, only Breath First Search and Uniform Cost Search give the optimal plan for three problems. Among them, Breath First Search uses slightly less memory while Uniform Cost Search is slightly faster.

We could see that our observation align with the text book [1]. The time and space usage in uniform cost is highly depend on the cost.

3.4.7 Comparing uninformed search strategies

Figure 3.21 compares search strategies in terms of the four evaluation criteria set forth in Section 3.3.2. This comparison is for tree-search versions. For graph searches, the main differences are that depth-first search is complete for finite state spaces and that the space and time complexities are bounded by the size of the state space.

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.

If the optimality is our goal, both Breadth First Search and Uniform Cost Search are recommended. However, if speed is the critical factor, Greedy Best First Graph Search is recommended. It could still give fairly good plan.

Heuristic Search

Problem 1

Algorithm	Plan length	Expansions	Elapse Time
A* with h1 heuristic	**6**	55	0.0450s
A* with ignore precondition heuristic	**6**	41	**0.0446s**
A* with level sum heuristic	**6**	**11**	1.1420s

Problem 2

Algorithm	Plan length	Expansions	Elapse Time
A* with h1 heuristic	**9**	4853	14.6344s
A* with ignore precondition heuristic	**9**	1450	**5.4664s**
A* with level sum heuristic	**9**	**86**	196.9359s

Problem 3

Algorithm	Plan length	Expansions	Elapse Time
A* with h1 heuristic	**12**	18541	61.2895s
A* with ignore precondition heuristic	13	8328	30.0528s
A* with level sum heuristic	13	**749**	> 20 mins

Analysis

In general, A* with ignore precondition heuristic is the fastest one and give a pretty good result in term of optimality and memory usage. However, if memory is a critical resource, A* with level sum heuristic would be the choice.

Conclusion

When comparing with the result from non-heuristic algorithm, A* with ignore precondition heuristic would also be the best algorithm in term of optimality, memory usage and performance.

Reference

[1] Stuart Russell, Peter Norvig (Third Edition). "Artifical Intelligence A Modern Approach"