Classical Planning Heuristic Analysis

David Box

Introduction

In this project I've implemented planning AI agents for fully observable, deterministic, static environments with single agents. In particular I developed planning agents for the air cargo transport "world" discussed in Artificial Intelligence: A Modern Approach (AIMA) section 10.1.1 in the 3rd edition (11.2 in the 2nd edition). To represent the world the project uses the Planning Domain Definition Language first discussed in section 10.1. Then I run uninformed planning searches (breadth first search, depth first graph search, etc) and use them as a baseline to compare their performance against the A* planning graph search using domain-independent heuristics. This provides a good comparison between "uninformed" planning and "informed" planning (distinctions discussed below).

$Action(Load(c, p, a),$
PRECOND: $At(c, a) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $\neg At(c, a) \land In(c, p))$
$Action(Unload(c, p, a),$
PRECOND: $In(c, p) \land At(p, a) \land Cargo(c) \land Plane(p) \land Airport(a)$
EFFECT: $At(c, a) \land \neg In(c, p))$
$Action(Fly(p, from, to),$
PRECOND: $At(p, from) \land Plane(p) \land Airport(from) \land Airport(to)$
EFFECT: $\neg At(p, from) \land At(p, to))$

Figure 1: PDDL description of air cargo transportation action schema

- Problem 1 initial state and goal:

```
Init(At(C1, SFO) ∧ At(C2, JFK)
        ∧ At(P1, SFO) ∧ At(P2, JFK)
        ∧ Cargo(C1) ∧ Cargo(C2)
        ∧ Plane(P1) ∧ Plane(P2)
        ∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

- Problem 2 initial state and goal:

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
        ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
        ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
        ∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
        ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

- Problem 3 initial state and goal:

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
        ∧ At(P1, SFO) ∧ At(P2, JFK)
        ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
        ∧ Plane(P1) ∧ Plane(P2)
        ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
```

Figure 2: PDDL description of air cargo transportation planning problems

Optimal Plan Lengths

Problem 1	**Problem 2**	**Problem 3**
Load(C1, P1, SFO)	Load(C1, P1, SFO)	Load(C2, P2, JFK)
Load(C2, P2, JFK)	Load(C2, P2, JFK)	Fly(P2, JFK, ORD)
Fly(P1, SFO, JFK)	Load(C3, P3, ATL)	Load(C4, P2, ORD)
Fly(P2, JFK, SFO)	Fly(P1, SFO, JFK)	Fly(P2, ORD, SFO)

Unload(C1, P1, JFK)	Fly(P2, JFK, SFO)	Load(C1, P1, SFO)
Unload(C2, P2, SFO)	Fly(P3, ATL, SFO)	Fly(P1, SFO, ATL)
Optimal Length: 6	Unload(C3, P3, SFO)	Load(C3, P1, ATL)
	Unload(C2, P2, SFO)	Fly(P1, ATL, JFK)
	Unload(C1, P1, JFK)	Unload(C4, P2, SFO)
	Optimal Length: 9	Unload(C3, P1, JFK)
		Unload(C2, P2, SFO)
		Unload(C1, P1, JFK)
		Optimal Length: 12

Table 1: Example Optimal Plans and the Optimal Lengths for each Problem

Uniformed Planning Searches Results & Analysis

These searches are considered “uninformed” due to the fact that they aren’t extracting extra information from the states. These searches are simply expanding nodes according to their heuristic until they find a goal state. For each problem I compare the relative performance of the uniformed planning searches over the following criteria: execution time, search node expansions (correlates to memory usage), path length, and whether or not the solution is optimal.

Search Type	Execution Time (s)	Node Expansions	Path Length	Optimal?
breadth_first_search	.053	43	6	Yes
depth_first_graph_search	.015	21	20	No
uniform_cost_search	.038	55	6	Yes
greedy_best_first_search	.005	7	6	Yes

Table 2: Uninformed Problem 1 Results

Search Type	Execution Time (s)	Node Expansions	Path Length	Optimal?

breadth_first_search	13.176	3346	9	Yes
depth_first_graph_search	0.309	107	105	No!!!
uniform_cost_search	11.15	4852	9	Yes
greedy_best_first_search	2.365	990	15	No

Table 3: Uninformed Problem 2 Results

Search Type	**Execution Time (s)**	**Node Expansions**	**Path Length**	**Optimal?**
breadth_first_search	92.366	14120	12	Yes
depth_first_graph_search	1.062	292	288	No!!!
uniform_cost_search	48.923	18235	12	Yes
greedy_best_first_search	15.112	5614	22	No

Table 4: Uninformed Problem 3 Results

Breadth_first_search and uniform_cost_search are the only two optimal searches. Between the two, uniform_cost_search is slightly faster but uses slightly more memory (i.e. more node expansions). Depth_first_graph_search is (comparatively) blazing fast and uses less memory but comes up with a **very** non-optimal plan. Greedy_best_first_search is (compared to the other searches) more “well-rounded”. The execution time and node expansions can be much less than breadth_first_search and uniform_cost_search. The path length, however, was not always optimal but much shorter than depth_first_graph_search.

Given that most modern computers have plenty of memory and that finding the optimal path length is often of utmost importance, uniform_cost_search is likely the best search heuristic. If finding the optimal path is not critical, then greedy_best_first_search would be a solid pick.

A* Planning Graph Search Results & Analysis

These searches are considered “informed” due to their extraction of extra information from the problem definition. The A* search uses an “admissible” heuristic by defining a relaxed problem from the original problem and estimating the associated cost. The

heuristic then uses this cost estimate to more intelligently determine the search direction. I'll look at the same result criteria as defined in the previous section.

Search Type	Execution Time (s)	Node Expansions	Path Length	Optimal?
A* search with h1 heuristic	0.060	55	6	Yes
A* search with ignore precond heuristic	0.049	41	6	Yes
A* search with level sum heuristic	1.009	11	6	Yes

Table 5: A* Problem 1 Results

Search Type	Execution Time (s)	Node Expansions	Path Length	Optimal?
A* search with h1 heuristic	11.193	4852	9	Yes
A* search with ignore precond heuristic	4.156	1450	9	Yes
A* search with level sum heuristic	193.543	86	9	Yes

Table 6: A* Problem 2 Results

Search Type	Execution Time (s)	Node Expansions	Path Length	Optimal?
A* search with h1 heuristic	48.814	18235	12	Yes
A* search with ignore precond heuristic	16.099	5040	12	Yes
A* search with level sum heuristic	796.637	325	12	Yes

Table 7: A* Problem 3 Results

All A* search heuristics solve all 3 problems optimally. Using the ignore precond heuristic resulted in the fastest times and was even more memory efficient than the second fastest heuristic (h1). The level sum heuristic should only be used when the computer is memory constrained. Level sum uses far less memory (lower node expansion count) but is **much slower** than all other heuristics.

Conclusion

Looking at Table 8, a comparison of the fastest uninformed search vs the fastest informed search, clearly shows the benefits of extracting extra information from the original problem. The A* search with ignore preconditions heuristic outperforms in speed and memory across all three problems save in speed for problem 1. Given that problem 1 requires the shortest amount of levels to solve, informed searches aren't worth the effort. For problem 3, not only is it ~3x faster but it is also ~4x more memory efficient. A* search with the ignore preconditions heuristic is the best search heuristic for the air cargo transportation planning problem.

Search Type	**Execution Time (s)**	**Node Expansions**	**Path Length**	**Optimal?**
Problem 1				
uniform_cost_search	0.038	55	6	Yes
A* search with ignore precond heuristic	0.049	41	6	Yes
Problem 2				
uniform_cost_search	11.15	4852	9	Yes
A* search with ignore precond heuristic	4.156	1450	9	Yes
Problem 3				
uniform_cost_search	48.923	18235	12	Yes
A* search with ignore precond heuristic	16.099	5040	12	Yes

Table 7: uniform_cost_search and A* search with ignore precond comparison