Heuristics analysis:
Planning Search Problems

Introduction

The aim of this project was to experiment with different heuristics in the context of planning search agent programming, when trying to solve logistics planning problems. Specifically, we were implementing a logistics planning system for an hypothetical air cargo transportation company.

In this analysis, we reflect on the experiment results for the abovementioned problem, when solving it using graph search and heuristics such as breadth-first, depth-first, greedy search, A*. We compare three versions of the problems with different levels of complexity and reflect on implications of the results, based on metrics as optimality of the solution, computational expensiveness and time to solve the problem.

The problem setting

The Air Cargo domain is defined by the following action schema.

Action(Load(c, p, a),
PRECOND: At(c, a) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
EFFECT: ¬ At(c, a) ∧ In(c, p))

Action(Unload(c, p, a),
PRECOND: In(c, p) ∧ At(p, a) ∧ Cargo(c) ∧ Plane(p) ∧ Airport(a)
EFFECT: At(c, a) ∧ ¬ In(c, p))

Action(Fly(p, from, to),
PRECOND: At(p, from) ∧ Plane(p) ∧ Airport(from) ∧ Airport(to)
EFFECT: ¬ At(p, from) ∧ At(p, to))

The 3 Problems

Each problem is defined with the above mentioned Air Cargo domain, but with different initial states and goals. As we can see, problem 1 is the simplest of all three, and problem 3 is the most complex of the problems.

Problem 1

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

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

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))

Experimental results

For both problem 2 and 3, some search strategies (breadth-first, depth-limited, recursive best first with h1, A* with level-sum) did not complete execution within 10 minutes. As per the instruction of the project, I cancelled their execution and marked them in red in the result tables.

In the result tables, I marked in green the best-performing strategies for the three interesting dimensions: number expansions (computational complexity), time to solution and plan length (optimality).

Searches	**Problem1**				
	Expansions	Goal Tests	New nodes	Time (s)	Plan length
breadth-first	43	56	180	0.017	6
breadth-first tree	1458	1459	5960	0.54	6
depth-first graph	12	13	48	0.0045	12
depth-limited	101	217	414	0.071	50
uniform cost	55	57	224	0.021	6
recursive best first h1	4229	4230	17029	1.56	6
greedy best first graph h1	7	9	28	0.0028	6
A* h1	55	57	224	0.02	6
A* ignore-preconditions	41	43	170	0.02	6
A* level-sum	55	57	224	2.89	6

Searches	**Problem2**				
	Expansions	Goal Tests	New nodes	Time (s)	Plan length
breadth-first	2845	4034	23638	3.95	9
breadth-first tree				600+	
depth-first graph	63	64	399	0.067	59
depth-limited				600+	
uniform cost	4281	4283	35054	5.76	9
recursive best first h1				600+	
greedy best first graph h1	547	549	4031	0.69	16
A* h1	4281	4283	35054	5.79	9
A* ignore-preconditions	1261	1264	35054	1.96	9
A* level-sum				600+	

Searches	**Problem3**				
	Expansions	Goal Tests	New nodes	Time (s)	Plan length
breadth-first	14663	18098	128605	23.6	12
breadth-first tree				600+	
depth-first graph	3664	3665	29381	10.1	195
depth-limited				600+	
uniform cost	18223	18225	158174	29.5	12
recursive best first h1				600+	
greedy best first graph h1	5655	5657	49083	9.2	26
A* h1	18223	18225	158174	30.2	12
A* ignore-preconditions	18223	18225	158174	29.8	12
A* level-sum				600+	

Analysis

Optimal plans

The experiments yielded the following optimal plans for the three problems.

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

Analysis

Based on the results of the experiments, we see that the optimal solutions found for each problem have a length of 6, 9, and 12 steps respectively. In problem 1, we can see that nearly all strategies allowed to find an optimal solution. But as the problem gets more complex in problem 2 and 3, some strategies find suboptimal solutions: these are depth-first and greedy best-first searching strategies. But although they find suboptimal solutions, they find them in significant shorter time than the time the other strategies need to find optimal solutions.

Thus, there is a clear **tradeoff** between finding an **optimal solution** and finding a solution **fast**. Furthermore, certain strategies will fail to converge in acceptable time due to the vastness of the problem space, as we can see with the four strategies which exceeded 10 minutes in execution.

Comparing non-heuristic search results, we can see that **breadth-first** strategies yield **optimal** results but take **longer** to find them than **depth-first** searches as they need to explore every single node between the start and the first possible to find one. On the other side, **depth-first** strategies are very **fast** and need to explore way less of the search space, but they yield **suboptimal** solutions. This is no surprise and is according to the course's theory. Breadth-first strategies are **guaranteed to find the optimal solution but are computationally expensive**, whereas depth-first strategies are **very fast but have very low chance of finding optimal solutions** and have the downside of being **unbound**, meaning that in a unbounded problem they might never converge.

Greedy best-first graph strategy seems like a good compromise between both extremes, as it achieves better solutions in all three problems than depth-first strategies, but is way faster to find solutions than breadth-first in both problem 2 and 3, meaning that it scales better with complexity.

Comparing heuristic search results, we can see that all variations of A* yield optimal results (by definition for A* if the heuristic is not overestimating). The more complex the heuristics, the more computation it will require. As a result, A* with level-sum fails to find solutions in acceptable time (<10 minutes) for problems 2 and 3.

Conclusion

Overall, it appears that the best strategy for this problem setting is **Greedy best-first graph**, if you look for a compromise between speed and acceptable length of solution.

If optimality of solution is a hard requirement, then **A* with ignore-preconditions** heuristic performs best, which is according to intuition and theory.

The results of this experiment illustrate the benefits of using informed search strategies with problem-relaxing heuristics over uninformed search techniques, with the goal of searching for optimal plans. Unlike for uninformed strategies, one can try and fine-tune heuristics to the specific problem to try reaching better results, which means putting more information into the informed search. But it is important to keep in mind that there is a tradeoff between computing complex heuristics values and finding a solution fast.