Dana Sheahen

Project 2 – Implement a Planning Search

Heuristics Analysis

Introduction

The purpose of this project is to compare various planning search algorithms using three different problems from the Air Cargo domain presented in the lectures and text (Russell & Norvig, 2010). This analysis paper addresses "Part 4" of the project and responds to the prompts presented in the README file associated with the project.

1. Provide an optimal plan for Problems 1, 2, and 3.
2. Compare and contrast uninformed search result metrics for Problem 1 and Problem 2. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison.
3. Provide a brief explanation of the automatic heuristics you implemented; discuss the advantages and disadvantages of each.
4. Compare and contrast informed search result metrics for Problems 1, 2, and 3
5. What was the best heuristic used in these problems? Was it better than uninformed search planning methods for all problems? Why or why not?

1. Optimal Plans

Problem 1: 6 steps

Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Problem 2: 9 steps

Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)

Problem 3: 12 steps

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)

2. Compare and contrast uninformed search P1, P2

The table below represents all the results for the three problems for comparison. The searches were run using the run_search.py utility. The plan length represents the number of actions required to reach the goal as determined by the particular planning search. The "expansions" represent the number of frontier nodes that were expanded in order to find the solution, and represent the complexity of the search. Only the A* searches include heuristics coded for the project.

Problem	Search	Heuristic	Plan length	Expansions	Time (sec)	Time (min)
1	breadth_first_search		6	43	0.030	0.0005
1	depth_first_graph_search		20	21	0.013	0.0002
1	uniform_cost_search		6	55	0.036	0.0006
1	greedy_best_first_graph_search		6	7	0.005	0.0001
1	A*	h=1 (null heuristic)	6	55	0.036	0.0006
1	A*	ignore preconditions	6	41	0.030	0.0005
1	A*	levelsum	6	11	1.069	0.0178
1	A*	setlevel	6	28	2.043	0.0340
2	breadth_first_search		9	3343	12.205	0.2034
2	depth_first_graph_search		380	386	1.525	0.0254
2	uniform_cost_search		9	4853	38.337	0.6389
2	greedy_best_first_graph_search		17	970	6.174	0.1029
2	A*	h=1 (null heuristic)	9	4853	38.662	0.6444
2	A*	ignore preconditions	9	1506	10.990	0.1832
2	A*	levelsum	9	86	100.926	1.6821
2	A*	setlevel	9	1287	1200.968	20.0161
3	breadth_first_search		12	14663	105.753	1.7625
3	uniform_cost_search		12	18236	362.171	6.0362
3	A*	h=1 (null heuristic)	12	18236	342.567	5.7094
3	A*	ignore preconditions	12	5118	73.900	1.2317
3	A*	levelsum	12	404	793.750	13.2292
3	A*	setlevel	12	5892	7828.691	130.4782

The uninformed searches are those without heuristics to aid them. These include breadth-first, depth-first, uniform-cost, and greedy-first, which were run for both Problem 1 and Problem 2.

Optimality

If only looking at the simple Problem 1 results, one might conclude that the best algorithm is greedy-first. However, greedy-first will not reliably result in an optimal plan. This is demonstrated in Problem 2 where greedy-first finds a plan with 17 steps instead of the optimal answer, which is 9 steps. For these two problems, depth-first ran quickly, but it did not give optimal results for either problem. For optimality, one could choose either breadth-first or uniform-cost searches among the non-heuristic choices.

Efficiency

Efficiency really encompasses more than one factor. Both speed and number of expansions come into play and result in trade-offs. In the case of uninformed searches such as these, the details of the problem will dictate which method is "better". Looking at speed, depth-first and greedy-first were both very fast, but not optimal. This could result in very long searches in more complex problems. Breadth-first and Uniform-cost were both optimal but a bit slower.

3. Automatic Heuristics

Three automatic heuristics were implemented for the project:

1. Ignore-preconditions – Based on the assumption of sub-goal independence, as mentioned in the README, this heuristic is simply the number of goal requirements for the problem still not met at any given node in the planning search.
2. Level-sum – The planning graph implemented creates a graph with alternating "levels" of S0, A0, S1, A1, S2, …. where S levels are literal levels and A levels are action levels. The level cost for

any given goal requirement is the level number where it first appears. The level-sum heuristic is the sum of the level costs found for each of the goal requirements. This assumes an independence of the sub-goals.

3. Set-level – This planning graph heuristic is completely admissible even if there is no sub-goal independence. It is calculated by finding the level at which all goal requirements are present and are NOT mutually exclusive to each other.

The tradeoff for these heuristics is speed vs optimality. The closer a heuristic value is to the actual distance to goal, the better it is at helping A* search lower the number of expansions, thus improving efficiency within the planning search. However, if calculating the heuristic takes a lot of time and space, that efficiency gain is lost simply in calculating the heuristic. For these three heuristics, the Ignore-preconditions heuristic is the simplest. It does not take into consideration, for example, that a plane must be loaded, then flown, then unloaded, to meet a requirement to move cargo from one airport to another, whereas level-sum and set-level will more closely calculate the number of steps needed to goal. However, it is so fast that even though it is a poor heuristic, it is useful. This may not be the case for some problems and using a more efficient implementation of the planning graph.

4. Compare and contrast informed search: P1, P2, P3

Of the three heuristics implemented, the ignore-preconditions heuristic was the fastest for all problems, but the level-sum heuristic resulted in the fewest node expansions for all problems. Set-level was slower and varied in node expansions as compared to the ignore-preconditions heuristic. For P1 and P2, set-level had fewer expansions than ignore-preconditions, but more in P3.

5. The best

The best heuristic of the three was the ignore-preconditions heuristic. It is very fast. For these problems it did not result in the fewest expansions, but its speed made it on balance the best choice. In my implementation, it was also faster than all the uninformed planning searches that I tested. This is because it struck a balance between providing a usable heuristic for the A* search, which reduced expansions compared to uninformed searches, and the speed of calculation for the heuristic.

References

Russell, S., & Norvig, P. (2010). *Artificial Intelligence: a modern approach.*