Heuristic Analysis (Planning Search)

Thilina Shihan Weerathunga

July 2017

1 Introduction

In this project I implemented a planning search agent to solve deterministic logistics planning problems for an Air Cargo transport system. Optimal plans for each problem are computed using progression search algorithms.

2 Planning Problems

All the given problems are in the Air Cargo domain and detailed problem definitions can be found here [1]. Given three problems have the same action schema defined, but different initial states and goals.

- Air Cargo Action Schema:(Using classical Planning Domain Definition Language)

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))

2.1 Initial states and goals for 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))

2.2 Initial states and goals for 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))

2.3 Initial states and goals for 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))

3 Analysis

Question:
Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.

3.1 Non-Heuristic Analysis

For this analysis, I used the following algorithms and results are summarized in Table. 1.

- Breadth First Search (BFS)
- Depth First Graph Search (DFGS)
- Uniform Cost Search (UCS)

1. In terms of achieving the optimal solution both BFS and UCS are successful for all the problems (P1, P2, P3).
2. If achieving optimal solution is the main priority, search strategy should be BFS, since its execution is faster and number of node expansion is less relative to UCS.

3. In terms of execution time and number of nodes expanded, DFGS is doing a better job than BFS and UCS or all the problems (P1, P2, P3). But the solutions are not optimal (Path lengths are not optimal).

Problem	Search Algorithm	Path Length	Optimal	Num: of Node Expansions	Goal Tests	Time Elapsed (seconds)
P1	BFS	6	Yes	43	56	0.03740
P1	DFGS	12	No	12	13	0.00936
P1	UCS	6	Yes	55	57	0.04265
P2	BFS	9	Yes	3401	4672	17.87655
P2	DFGS	346	No	350	351	1.87238
P2	UCS	9	Yes	4761	4763	14.03204
P3	BFS	12	Yes	14491	17947	157.84704
P3	DFGS	1878	No	1948	1948	26.27913
P3	UCS	12	Yes	17783	17785	70.44745

Table 1: Analysis results for uninformed non-heuristic search.

3.2 Heuristic

Question: Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.

For this analysis, I used the A^* search with following heuristic. Results are summarized in Table. 2

- h_1 :- Heuristic always returns 1.
- h_ignore_preconditions : This heuristic estimates the minimum number of actions that must be carried out from the current state in order to satisfy all of the goal conditions by ignoring the preconditions required for an action to be executed.
- h_pg_level_sum : This heuristic uses a planning graph representation of the problem state space to estimate the sum of all actions that must be carried out from the current state in order to satisfy each individual goal condition.

1. In terms of execution time:
h_ignore_preconditions < h_1 < h_pg_level_sum for all P1, P2, and P3.
2. In terms of achieving the optimality all three heuristics are successful. All three strategies give the same path length for each problem.(P1:- 6, P2:- 9, P3:- 12)

3. In terms of the number of nodes expanded:
   h_pg_level_sum < h_ignore_preconditions < h_1 for all P1, P2, and P3.

Problem	Heuristic used for A^* search	Path Length	Optimal	Num: of Node Expansions	Goal Tests	Time Elapsed (seconds)
P1	h_1	6	Yes	55	57	0.04489
P1	h_ignore_precond :	6	Yes	41	43	0.03391
P1	h_pg_level_sum :	6	Yes	11	13	0.65484
P2	h_1	9	Yes	4761	4763	13.95826
P2	h_ignore_precond :	9	Yes	1450	1452	4.37375
P2	h_pg_level_sum :	9	Yes	86	88	55.08859
P3	h_1	12	Yes	17783	17785	71.06305
P3	h_ignore_precond :	12	Yes	5003	5005	21.72093
P3	h_pg_level_sum :	12	Yes	311	313	329.28807

Table 2: Analysis results for A^* heuristic search.

Question: What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?

Since all the heuristics achieved the optimal plan we can't use the optimality as the decision variable. If we have time limitations we can conclude that h_ignore_preconditions heuristic is the best. In terms of number of nodes expanded, h_pg_level_sum expanded lower number of nodes. Hence good in terms of memory usage.

If I choose h_ignore_preconditions heuristic as the best heuristic for all the three problems, I can conclude that it is better than non-heuristic search planning methods as well.
The reasons are listed below.

It performs better than non-heuristic **optimal strategies** in terms of execution time (There is one exception - P1:BFS).

It performs better than non-heuristic **optimal strategies** in terms of node expansion.

References

[1] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach Pearson Education, 2003, 2nd edition.