Heuristic Analysis

PLANNING SEARCH

Submitted By:-

Nishant Mittal

Artificial Intelligence Nanodegree, Udacity

Synopsis

In this project, we implemented a planning search agent to solve deterministic planning problem for an Air Cargo Transport system.

We use **planning graph** and **automatic domain independent heuristic with A* search** and compare their results/performance against several **uninformed heuristic search methods.**

Planning Problem:

We were given 3 planning problems in Air Cargo domain with 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))
```

Initial state and Goal and these 3 problems were:

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))
```

The goals described above can be reached using different plans.

Optimal path length of problem 1, 2 and 3 are 6, 9 and 12 respectively. Below are the actions taken in each problem to reach goal state:

Problem 1:

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

Problem 2:
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(C2, P2, SFO)
Unload(C1, P1, JFK)

Problem 3:
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Fly(P1, SFO, ATL)
Load(C3, P1, ATL)
Fly(P2, JFK, ORD)
Load(C4, P2, ORD)
Fly(P2, ORD, SFO)
Fly(P1, ATL, JFK)
Unload(C4, P2, SFO)
Unload(C3, P1, JFK)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)

Uninformed search strategies:

The term uniformed search strategies means that the strategies have no additional information about the states beyond that provided in problem definition. All they can do is generate successors and distinguish a goal state from non-goal state. All search strategies are distinguished by the order in which nodes are expanded.

In this section, we compare the performance of such strategies in terms of speed, memory usage and optimality.

Below are the performance results for the same:

Problem 1 result:

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
Breadth first search	Yes	6	0.08	43	56	180
Breadth first tree search	Yes	6	1.10	1458	1459	5960
Depth first graph search	No	12	0.008	12	13	48
Depth limited search	No	50	0.098	101	271	414
Uniform cost search	Yes	6	0.040	55	57	224
Recursive best first search	Yes	6	3.08	4229	4230	17029
Greedy Best First GS with h_1	**Yes**	**6**	**0.005**	**7**	**9**	**28**

Problem 2 result:

In problem 2 we cancelled data collection for Breadth first tree search; depth limited search and recursive best first search because their execution time was exceeding 10 min.

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
Breadth first search	**Yes**	**9**	15.41	3346	4612	30534
Breadth first tree search	-	-	-	-	-	-
Depth first graph search	No	1085	**9.59**	**1124**	**1125**	**10017**
Depth limited search	-	-	-	-	-	-
Uniform cost search	**Yes**	**9**	29.03	4852	4854	44030
Recursive best first search	-	-	-	-	-	-
Greedy Best First GS with h_1	Yes	21	**3.11**	**990**	**992**	**8910**

Problem 3 results:

In problem 3 we cancelled the data collection for breadth first tree search, Depth limited search and recursive best first search because their execution time was exceeding 10 min.

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
Breadth first search	**Yes**	**12**	125.76	14120	17673	123964
Breadth first tree search	-	-	-	-	-	-
Depth first graph search	No	2031	**46.31**	**5591**	**5592**	**45563**
Depth limited search	-	-	-	-	-	-
Uniform cost search	**Yes**	**12**	74.13	18235	18237	158272
Recursive best first search	-	-	-	-	-	-
Greedy Best First GS with h_1	Yes	26	**20.17**	**5673**	**5675**	**49221**

Analysis:

With the 3-problem set we found that **Breadth first search and Uniform cost search** are the only two uninformed strategies that provided optimal action plan in terms of all the 3 parameters speed (exc. Time), memory usage(no of nodes expended) and optimality (find goal state). **Depth first search** is good in terms of execution time, node expansion but not good in terms of optimality, as we can see in all the problem sets path length derived are 12, 1085 and 2031 which is not good in comparison to 6, 9 and 12 respectively. So if the main motive is speed and memory Depth First search can be given priority.

In case of finding **optimal path Breadth first search** should be given priority because it performs **faster** and **uses less memory** than uniform cost search.

Informed search strategies analysis:

Informed search strategy is the one that uses problem specific knowledge beyond the definition of problem itself – can find solutions more efficiently than uniformed strategies.

Below are the performance measures collected for problem 1, 2 and 3 using informed search strategies.

Problem 1 Results:

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
A* with h_1	Yes	6	0.03	55	57	224
A* with h_ignore_prec.	Yes	**6**	**0.044**	41	43	170
A* with h_pg_levelsum	Yes	6	0.9	**11**	**13**	**50**

Problem 2 Results:

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
A* with h_1	Yes	9	15.06	4852	4854	44030
A* with h_ignore_prec.	Yes	**9**	**5.34**	1450	1452	13303
A* with h_pg_levelsum	Yes	9	201.39	**86**	**88**	**841**

Problem 3 Results:

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
A* with h_1	Yes	12	65.47	18235	18237	158272
A* with h_ignore_prec.	Yes	**12**	**23.32**	5040	5042	44769
A* with h_pg_levelsum	Yes	12	1303.53	**387**	**389**	**3550**

Analysis:

While all heuristics yield an optimal action plan, only h1 and ignore precondition heuristics return results within 10 min max execution time. If we consider speed and optimal path length then **A* with ignore precondition heuristics** should be used as we can see in above data collection although A* with heuristics and A* with ignore precondition heuristics yield result with same path length but execution time of A* with ignore heuristics is much more efficient in execution time.

If we consider memory than A* search with level sum heuristics uses least memory but execution time much more as compared to others.

Comparison of informed and uninformed search strategies:

Problem 1:

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
Breadth first search	Yes	6	0.08	43	56	180
A* with h_ignore_prec.	**Yes**	**6**	**0.044**	**41**	**43**	**170**

Problem 2:

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
Breadth first search	Yes	9	15.41	3346	4612	30534
A* with h_ignore_prec.	**Yes**	**9**	**5.34**	**1450**	**1452**	**13303**

Problem 3:

Search Type	Optimal	Path Length	Execution time	Node expansion	Goal Test	New nodes
Breadth first search	Yes	12	125.76	14120	17673	123964
A* with h_ignore_prec.	**Yes**	**12**	**23.32**	**5040**	**5042**	**44769**

Conclusion:

Overall if we compare uninformed and informed search strategies, we could see that informed search strategies with heuristic search is more efficient in terms of speed, optimality and memory.

A* search with ignore precondition heuristics is the better choice overall.

Reference:

Artificial intelligence [Third edition] by Stuart Russell and Peter Norvig.