**Planning Search Heuristic Analysis**

This project tries to solve deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. Three different air cargo problems with increasing complexity are considered. Planning algorithm is developed and then different heuristics are implemented. Results are captured for all three problems with different variants of search algorithms and heuristic functions.

Problem Definition :

Air Cargo 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))

* 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))

**Optimal Solution:**

**Problem 1:**

Load(C2, P2, JFK)

Load(C1, P1, SFO)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

**Problem 2:**

Load(C2, P2, JFK)

Load(C1, P1, SFO)

Load(C3, P3, ATL)

Fly(P2, JFK, SFO)

Unload(C2, P2, SFO)

Fly(P1, SFO, JFK)

Unload(C1, P1, JFK)

Fly(P3, ATL, SFO)

Unload(C3, P3, SFO)

**Problem 3:**

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)

**Results:**

**Problem 1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Search Algorithm |  | Expansions | Goal Tests | New Nodes | Plan Length | Execution Time( sec) | Optimal |
| Breadth First Search | Uninformed | 43 | 56 | 180 | 6 | 0.0255 | Yes |
| Breadth First Tree Search | Uninformed | 1458 | 1459 | 5960 | 6 | 0.7728 | Yes |
| Depth First Graph Search | Uninformed | 12 | 13 | 48 | 12 | 0.0070 | No |
| Depth Limited Search | Uninformed | 101 | 271 | 414 | 50 | 0.0724 | No |
| Uniform Cost Search | Uninformed | 55 | 57 | 224 | 6 | 0.0328 | Yes |
| Recursive best First Search with h\_1 | Uninformed | 4229 | 4230 | 17029 | 6 | 2.2016 | Yes |
| Greedy Best First Graph Search with h\_1 | Uninformed | 7 | 9 | 28 | 6 | 0.0047 | Yes |
| AStar\_search h\_1 | Informed | 55 | 57 | 224 | 6 | 0.0325 | Yes |
| AStar\_search h\_ignore preconditions | Informed | 41 | 43 | 170 | 6 | 0.0247 | Yes |
| AStar\_search h\_pg\_levelsum | Informed | 11 | 13 | 50 | 6 | 1.5969 | Yes |

**Problem 2**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Search Algorithm |  | Expansions | Goal Tests | New Nodes | Plan Length | Execution Time( sec) | Optimal |
| Breadth First Search | Uninformed | 3343 | 4609 | 30509 | 9 | 10.9713 | Yes |
| Breadth First Tree Search | Uninformed |  |  |  |  |  |  |
| Depth First Graph Search | Uninformed | 582 | 583 | 5211 | 575 | 2.4999 |  |
| Depth Limited Search | Uninformed | 222719 | 2053741 | 2054119 | 50 | 761.4264 |  |
| Uniform Cost Search | Uninformed | 4852 | 4854 | 44030 | 9 | 34.0382 | Yes |
| Recursive best First Search with h\_1 | Uninformed |  |  |  |  |  |  |
| Greedy Best First Graph Search with h\_1 | Uninformed | 990 | 992 | 8910 | 15 | 5.5545 |  |
| AStar\_search h\_1 | Informed | 4852 | 4854 | 44030 | 9 | 34.7623 | Yes |
| AStar\_search h\_ignore preconditions | Informed | 1506 | 1508 | 13820 | 9 | 9.7282 | Yes |
| AStar\_search h\_pg\_levelsum | Informed | 86 | 88 | 841 | 9 | 300.9577 | Yes |

**Problem 3**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Search Algorithm |  | Expansions | Goal Tests | New Nodes | Plan Length | Execution Time( sec) | Optimal |
| Breadth First Search | Uninformed | 14663 | 18098 | 129631 | 12 | 91.9579 | Yes |
| Breadth First Tree Search | Uninformed |  |  |  |  |  |  |
| Depth First Graph Search | Uninformed | 627 | 628 | 5176 | 596 | 3.14326 |  |
| Depth Limited Search | Uninformed |  |  |  |  |  |  |
| Uniform Cost Search | Uninformed | 18235 | 18237 | 159716 | 12 | 302.9910 | Yes |
| Recursive best First Search with h\_1 | Uninformed |  |  |  |  |  |  |
| Greedy Best First Graph Search with h\_1 | Uninformed | 5614 | 5616 | 49429 | 22 | 81.2844 |  |
| AStar\_search h\_1 | Informed | 18235 | 18237 | 159716 | 12 | 309.1720 | Yes |
| AStar\_search h\_ignore preconditions | Informed | 5118 | 5120 | 45650 | 12 | 65.8119 | Yes |
| AStar\_search h\_pg\_levelsum | Informed | 404 | 406 | 3718 | 12 | 2375.8244 | Yes |

**From the results, it is clear that not all search algorithms work for Problem 2 and 3. Although all algorithms run successfully for Problem 1 but not all are optimal.**

**Following two algorithms do not complete for Problem 2 and 3 -** Breadth First Tree Search, Recursive best First Search with h\_1

Problem 2 did complete for Depth Limited Search but it takes more than 10 minutes and it has too many node expansions.

Out of the remaining successfully running search algorithms, all complete within 10 minutes for all 3 problems except AStar\_search h\_pg\_levelsum algorithm for Problem 3.

**For uninformed search**

If path length is to be considered, then

**Greedy Best First Graph Search with h\_1** gives the best performance( node expansions ( memory) and execution time) for all Problem 1.

**Breadth First Search** gives the best performance( node expansions ( memory) and execution time ) for all Problem 2 and Problem 3.

If path length is not to be considered, then Depth First Graph search gives the best performance( node expansions ( memory) and execution time ) for all three problems

**For informed search**

**For all three Problems:**

AStar\_search h\_ignore preconditions gives best **execution time**

AStar\_search h\_pg\_levelsum gives best **memory utilization** ( less node expansion).

So depending upon the selection criteria ( execution time, memory utilization) , AStar\_search h\_ignore preconditions or AStar\_search h\_pg\_levelsum can be selected.

**Informed search with h\_ignore\_preconditions** takes less time to execute, as this heuristic relaxes the pre-conditions for actions. This relaxed problem becomes easier to solve and thus it takes less time to find a solution. The solution to this relaxed problem then becomes the solution of the main problem. In relaxed mode, Now all actions are possible in each state and a single goal fluent can be achieved in one step( if there is an applicable action, otherwise the solution is not possible). Instead of ignoring all pre-conditions, selected pre-conditions can also be ignored. Ignoring preconditions helps improve the execution time but it takes more memory as now system needs to store more state action pairs.

*[Ref : AIMA – 3rd Ed, Ch10, 10.2.3 – Heuristics for Planning]*

**Uninformed Search – Depth First/ Depth Limited** search does not seem to give good results in all three problems, as DFS/DLS explores the deepest path, it keeps on searching until a goal is found. This may not be optimal path as search may not have started with the node on the optimal path. As planning problems usually have large state spaces and DFS\DLS may get stuck in non-optimal paths, DFS\DLS does not seem to be good search strategies for AI Planning. Breadth First search does work as it systematically searches one level at a time but it may not be memory efficient technique because of large state space.

Planning problems having large state spaces can be solved better with heuristic based search.

*[Ref : AIMA – 3rd Ed, Ch3, 3.4.3 – Depth First Search, 3.4.4 – Depth Limited Search; 10.2.1 - Forward (progression) state-space search]*