Analysis of Planning Search (NDAI)

Shubra Chowdhury

Analysis of Planning Search

This project deals with planning(PDDL) of air cargo pickup and delivery to and from multiple location. This is a deterministic implementation with three specific actions Load, Fly and unload and three specific object type Airport, Cargo and Plane.

Implementation uses various search models, domain independent Informed A* heuristics (A*search, A*search ignoring preconditions, A* search with level sum), un-informed search planning graph (depth first graph search, greedy best first graph search), breadth first search, breadth first tree search depth limited search, uniform cost search, recursive best first search.

Give 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))
```

Number of problem which requires solution is 3, and for each of these problem initial and goal state has been provided.

PROBLEM INITIAL STAGE & GOAL:

Problem 1 initial state and goal	Problem 2 initial state and goal	Problem 3 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))	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))	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))

IMPLEMENTATION METHOD:

Analysis of Planning Search (NDAI)

Shubra Chowdhury

Problem 1 interpreted and implemented in method as follows	Problem 2 interpreted and implemented in method as follows	Problem 3 interpreted and implemented in method as follows
cargos = ['C1', 'C2'] *planes = ['P1', 'P2']* *airports = ['JFK', 'SFO']* *pos = [expr('At(C1, SFO)'), expr('At(C2, JFK)'), expr('At(P1, SFO)'), expr('At(P2, JFK)'),]* *neg = [expr('At(C2, SFO)'), expr('In(C2, P1)'), expr('In(C2, P2)'), expr('At(C1, JFK)'), expr('In(C1, P1)'), expr('In(C1, P2)'), expr('At(P1, JFK)'), expr('At(P2, SFO)'),]* *goal = [expr('At(C1, JFK)'), expr('At(C2, SFO)'),]*	*cargos = ['C1', 'C2','C3']* *planes = ['P1', 'P2','P3']* *airports = ['JFK', 'SFO','ATL']* *pos = [expr('At(C1, SFO)'),expr('At(C2, JFK)'),expr('At(C3, ATL)'),expr('At(P1, SFO)'), expr('At(P2, JFK)'),expr('At(P3, ATL)'),]* *neg = [expr('At(C3, SFO)'), expr('At(C3, JFK)'),expr('In(C3, P1)'),expr('In(C3, P2)'),expr('In(C3, P3)'),expr('At(C2, SFO)'),expr('At(C2, ATL)'),expr('In(C2, P1)'), expr('In(C2, P2)'),expr('In(C2, P3)'),expr('At(C1, JFK)'),expr('At(C1, ATL)'),expr('In(C1, P1)'),expr('In(C1, P2)'),expr('In(C1, P3)'),expr('At(P1, JFK)'), expr('At(P1, ATL)'),expr('At(P2, SFO)'),expr('At(P2, ATL)'),expr('At(P3, SFO)'),expr('At(P3, JFK)'),]* *goal = [expr('At(C1, JFK)'),expr('At(C2, SFO)'),expr('At(C3, SFO)'),]*	*cargos = ['C1', 'C2','C3','C4']* *planes = ['P1', 'P2']* *airports = ['JFK', 'SFO','ATL','ORD']* *pos = [expr('At(C1, SFO)'),expr('At(C2, JFK)'),expr('At(C3, ATL)'),expr('At(C4, ORD)'),expr('At(P1, SFO)'),expr('At(P2, JFK)'),]* *neg = [expr('At(C4, SFO)'),expr('At(C4, JFK)'),expr('At(C4, ATL)'),expr('In(C4, P1)'),expr('In(C4, P2)'),expr('At(C3, SFO)'),expr('At(C3, JFK)'),expr('At(C3, ORD)'), expr('In(C3, P1)'),expr('In(C3, P2)'), expr('At(C2, SFO)'), expr('At(C2, ATL)'),expr('At(C2, ORD)'),expr('In(C2, P1)'),expr('In(C2, P2)'),expr('At(C1, JFK)'), expr('At(C1, ATL)'), expr('At(C1, ORD)'),expr('In(C1, P1)'),expr('In(C1, P2)'),expr('At(P1, JFK)'),expr('At(P1, ATL)'),expr('At(P1, ORD)'),expr('At(P2, SFO)'), expr('At(P2, ATL)'),expr('At(P2, ORD)'),]* *goal = [expr('At(C1, JFK)'),expr('At(C2, SFO)'), expr('At(C3, JFK)'),expr('At(C4, SFO)'),]*

OPTIMUM SOLUTION:

Optimum Solution for Problem 1	Optimum Solution for Problem 2	Optimum Solution for 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) PLAN Length 6	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) PLAN Length 9	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) PLAN Length 12

Analysis of Planning Search (NDAI)

Shubra Chowdhury

Uninformed Planning Algorithms Analysis:

Uninformed planning strategies solely relies on the problem definition, it cannot provide any additional information about the action or state. It correctly identifies the state including the goal state and based on that looks for its successors.

My analysis will be based on fastest execution with least amount of resource used. For speed I will consider execution time and for resources used I will consider traversing (plan length) and memory (expansion).

Problem 1 Results:

P1 - Uninformed Search (Non-Heuristic Search)						
Search Algo	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed in seconds**	**Optimal**
breadth_first_search	43	56	180	6	0.030813854	Yes
breadth_first_tree_search	1458	1459	5960	6	0.924092619	Yes
depth_first_graph_search	12	13	48	12	0.00860885	No
depth_limited_search	101	271	414	50	0.089985242	No
uniform_cost_search	55	57	224	6	0.037300464	Yes
recursive_best_first_search h_1	4229	4230	17029	6	2.729997769	Yes
greedy_best_first_graph_search h_1	7	9	28	6	0.005518931	Yes

For problem 1 out of the 7 search strategies mentioned above 2 didn't work out to be optimum these are depth search graph and depth limited. The plan lengths are 6 steps and 44 steps higher than the optimal plan.

Although the plan length for recursive best first is optimal but it is **most expensive strategy with slowest speed**, it has highest expansion of 4229 unit and highest time of 2.73 seconds.

Of the remaining 4 strategies, Greedy best is the best strategy and I can say is the **Most Optimal strategy** it is the **Fastest Strategy** with time of 0.006 seconds with just 7 units of resource usage, next is breath first followed by Uniform cost and finally breath first tree. Note out of these second-tier strategies breath first tree is the slowest and most expensive.

Analysis of Planning Search (NDAI)

Shubra Chowdhury

Problem 2 Results:

P2 - Uninformed Search (Non-Heuristic Search)						
Search Algo	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed in seconds**	**Optimal**
breadth_first_search	3343	4609	30509	9	13.37229552	Yes
breadth_first_tree_search					Runtime more than 3 hours	
depth_first_graph_search	582	583	5211	575	3.17927075	No
depth_limited_search	222719	2053741	2054119	50	1392.878279	No
uniform_cost_search	4853	4855	44041	9	11.61663462	Yes
recursive_best_first_search h_1					Runtime more than 3 hours	
greedy_best_first_graph_search h_1	998	1000	8982	17	2.770988898	No

For problem set 2 there is no data available for breadth first tree and recursive best first search strategies. For both of these strategies I had to stop execution of the program after 3 hours.

Breadth first and Uniform cost search strategies have the best Plan Length, uniform search yielded **fastest speed** but has higher resource consumption than breadth first. Depth first graph has the least resource consumption while depth limited has the maximum resource and the slowest speed , this can be the worst strategy. For plan 2 uniform cost is close to optimal strategy.

Problem 3 Results:

P3- Uninformed Search (Non-Heuristic Search)						
Search Algo	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds	Optimal
breadth_first_search	14663	18098	129631	12	120.7390469	Yes
breadth_first_tree_search					Time beyond 3 hours	
depth_first_graph_search	627	628	5176	596	3.248967147	No

depth_limited_search						
uniform_cost_search	18151	18153	159038	12	65.30872264	Yes
recursive_best_first_search h_1					Time beyond 3 hours	
greedy_best_first_graph_search h_1	5398	5400	47665	26	18.04695123	No

For problem set 3 there is no data available for breadth first tree and recursive best first search strategies. For both of these strategies I had to stop execution of the program after 3 hours.

Uniform cost is the best optimal strategy this has the shortest plan length and minimum time for execution however this has higher resource usage than breadth first search strategy. Breadth first has the same path length as that of uniform cost and is less resource intensive than the uniform cost but its speed is almost half as slow as that of uniform cost.

Depth first graph has the longest plan length while speed is faster than greedy best first graph.

Informed (Heuristic) Planning Algorithms Analysis

Informed (Heuristic) search strategy uses problem specific knowledge beyond the definition of problem. My analysis will be based on fastest execution with least amount of resource used. For speed, I will consider execution time and for resources used I will consider traversing (plan length) and memory (expansion).

Problem 1 Result :

P1 - Informed Search (Heuristic Search)						
Search Algo	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds	Optimal

astar_search h_1	55	57	224	6	0.037068358	Yes
astar_search h_ignore_preconditions	41	43	170	6	0.038544624	Yes
astar_search h_pg_levelsum	11	13	50	6	0.561234534	Yes

For problem 1 all 3 strategies has the path length of 6 , considering the speed A* search h1 is the best it has the fastest time of execution. From the resource usage perspective A* Search h-pg level-sum is the best it has the least memory usage with just 11 units which is one fifth of A* Search and almost close to one fourth of A* Search Ignore Preconditions , however the speed is 1/15th of A* Search and /14th of A* Search Ignore Preconditions. Based on the mentioned analysis I would suggest A* search h1 is the optimal solution for Problem 1

Problem 2 Result :

P2 - Informed Search (Heuristic Search)						
Search Algo	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds	Optimal
astar_search h_1	4853	4855	44041	9	13.86266886	Yes
astar_search h_ignore_preconditions	1450	1452	13303	9	4.189866159	Yes
astar_search h_pg_levelsum	86	88	841	9	50.26162805	Yes

For problem 2 all 3 strategies has the path length of 9, considering the speed A* Search Ignore Preconditions is the best it has the fastest time of execution, it is 3 times faster than A* Search and 12 times faster than A* Search h-pg level-sum . From the resource usage perspective A* Search h-pg level-sum is the best it has the least memory usage with just 86 units which is 56 times less than of A* Search and almost close to 17 times less than that of A* Search Ignore Preconditions. Based on the mentioned analysis I would suggest A* Search Ignore Preconditions is the optimal solution for Problem 1

Analysis of Planning Search (NDAI)

Shubra Chowdhury

Problem 3 Result :

P3- Informed Search (Heuristic Search)						
Search Algo	Expansions	Goal Tests	New Nodes	Plan length	Time elapsed in seconds	Optimal
astar_search h_1	18151	18153	159038	12	81.48871616	Yes
astar_search h_ignore_preconditions	5038	5040	44926	12	20.11875494	Yes
astar_search h_pg_levelsum	314	316	2894	12	245.6921209	Yes

For problem 3 all 3 strategies has the path length of 12, considering the speed A* Search Ignore Preconditions is the best it has the fastest time of execution, it is 4 times faster than A* Search and 12 times faster than A* Search h-pg level-sum . From the resource usage perspective A* Search h-pg level-sum is the best it has the least memory usage with 314 units which is 57 times less than of A* Search and almost close to 16 times less than that of A* Search Ignore Preconditions. Based on the mentioned analysis I would suggest A* Search Ignore Preconditions is the optimal solution for Problem 1

Informed Vs Uninformed Search Strategies:

While comparing the informed and uninformed strategy we see that the with less number of object (less number of airports, cargo and flights) uninformed search works better. In problem 1 we have 2 cargo , 2 plane and 2 airport , however as the number of object increases and the search gets complicated Heuristic search works better , here in problem 2 which has 3 cargo , 3 plane , 3 airport and problem 3 which has 4 cargo , 2 plane and 4 airport A* Search is the better search strategy.

For Problem 1

P1						
Search Algo	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed in seconds**	**Best/Optimal**
greedy_best_first_graph_search h_1	7	9	28	6	0.005518931	Yes
astar_search h_1	55	57	224	6	0.037068358	

Analysis of Planning Search (NDAI)

Shubra Chowdhury

For Problem 2

P2						
Search Algo	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed in seconds**	**Best/Optimal**
uniform_cost_search	4853	4855	44041	9	11.61663462	
astar_search h_ignore_preconditions	1450	1452	13303	9	4.189866159	Yes

For Problem 3

P3						
Search Algo	**Expansions**	**Goal Tests**	**New Nodes**	**Plan length**	**Time elapsed in seconds**	**Best/Optimal**
uniform_cost_search	18151	18153	159038	12	65.30872264	
astar_search h_ignore_preconditions	5038	5040	44926	12	20.11875494	Yes

The comparative result above shows that for complicated search informed (Heuristic) search strategies provide better results with optimal plans. Both speed and memory usage is much lower than the uninformed search strategy. Different types of Heuristics can be used to tailor memory and speed to balance the optimal use of resources.