Tibor Zahorecz
Udacity Artificial Intelligence Nanodegree
Implement a Planning Search

The below analysis is about implementation of the planning search agent to solve planning problems for an Air Cargo transport system. Adopting uninformed non-heuristic search methods, then implement domain-independent heuristic search methods. (This project includes skeletons for the classes and functions needed to solve deterministic logistics planning problems for an Air Cargo transport system using a planning search agent. With progression search algorithms like those in the navigation problem from lecture, optimal plans for each problem will be computed. Unlike the navigation problem, there is no simple distance heuristic to aid the agent. Instead, you will implement domain-independent heuristics.)

Uninformed Search Strategies Analysis

Using Breadth_First_Search, Breadth_First_Tree_Search, Depth_First_Graph_Search, Depth_Limited_Search, and Uniform_Cost_Search. Judged on the basis of optimality, time complexity and space complexity.
Uninformed search (called blind search) have no additional information about states. All they can do is generate successors and distinguish a goal state from a non-goal state. We compare the performance of seven uninformed search strategies in terms of speed, memory usage and optimality.

Problem 1 initial state – goal, results:

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))

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes	Optimal
Uninformed	breadth_first_search	0.0508	6	Yes	43	56	180	Yes
	breadth_first_tree_search	1.10976	6	Yes	1458	1459	5960	Yes
	depth_first_graph_search	0.01090	12	No	12	13	48	No
	depth_limited_search	0.11020	50	No	101	271	414	No
	uniform_cost_search	0.0436	6	Yes	55	57	224	Yes
	recursive_best_first_search	3.91439	6	Yes	4229	4230	17029	Yes
	greedy_best_first_graph_search	0.00818	6	Yes	7	9	28	Yes

Optimal plan length is 6 - best strategy is greedy_best_first_graph_search.

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 initial state – goal, results:

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))

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Uninformed	breadth_first_search	19.487	9	Yes	3346	4612	30534
	breadth_first_tree_search	N.A.					
	depth_first_graph_search	3.000*	wrong	No	TBD	TBD	TBD
	depth_limited_search	1038.369	50	No	213491	1967093	1967471
	uniform_cost_search	13.339	9	Yes	4853	4855	44041
	recursive_best_first_search	N.A.					
	greedy_best_first_graph_search	2.913	21	No	998	1000	8982

Optimal plan length is 9 – best algorith is uninformed_cost_search.

Load(C1, P1, SFO)

Load(C2, P2, JFK)

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 initial state – goal, results:

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))

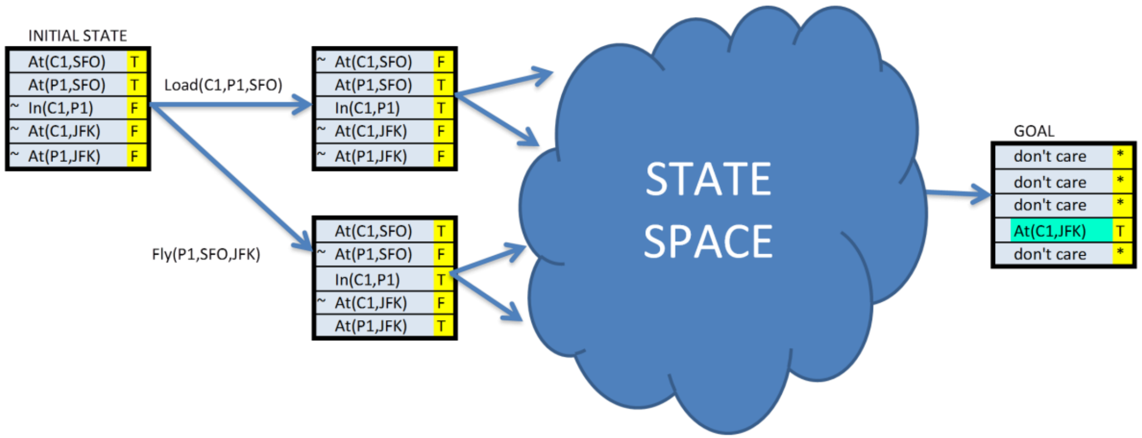
The search space for this problem is even larger than problem 2, and I was not able to run all search algorithm (timing issue).

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Uninformed	breadth_first_search	153.236	12	Yes	14663	18098	129631
	breadth_first_tree_search	N.A.					
	depth_first_graph_search	33.000*	wrong	No	TBD	TBD	TBD
	depth_limited_search	N.A.					
	uniform_cost_search	76.395	12	Yes	18235	18237	159716
	recursive_best_first_search	N.A.					
	greedy_best_first_graph_search	23.704	22	No	5614	5616	49429

The oprimal plan length is 12 – best performed algorithm is uniform_cost_search.

ANALYSIS

BFS (breadth_first_search algorithm expands all nodes at the frontier of the search graph before going deeper) and **UCS** (uniform_cost_search algorithm is guaranteed to find the path with the cheapest total cost) are the two uninformed search strategies make an optimal action plan.



Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes^a	Yes^{a,b}	No	No	Yes^a	Yes^{a,d}
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes^c	Yes	No	No	Yes^c	Yes^{c,d}

Figure 3.21 Evaluation of tree-search strategies. b is the branching factor; d is the depth of the shallowest solution; m is the maximum depth of the search tree; l is the depth limit. Superscript caveats are as follows: ^{a} complete if b is finite; ^{b} complete if step costs $\geq \epsilon$ for positive ϵ; ^{c} optimal if step costs are all identical; ^{d} if both directions use breadth-first search.

Informed (Heuristic) Search Strategies Analysis

Problem 1 results:

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Informed	A* Search with h_1	0.04690	6	Yes	55	57	224
	A* Search with h_ignore_preconditions	0.04610	6	Yes	41	43	170
	A* Search with h_pg_levelsum	1.131219	6	Yes	11	13	50

Problem 2 results:

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Informed	A* Search with h_1	15.063	9	Yes	4853	4855	44041
	A* Search with h_ignore_preconditions	5.6889	9	Yes	1450	1452	13303
	A* Search with h_pg_levelsum	196.639	9	Yes	86	88	841

Problem 3 results:

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Informed	A* Search with h_1	92.4660	12	Yes	18235	18237	159716
	A* Search with h_ignore_preconditions	28.7670	12	Yes	5040	5042	44944
	A* Search with h_pg_levelsum	1510.236	12	Yes	318	320	2934

All algorithms produce optimal action plan, only the h1 and Ignore Preconditions heuristics return results withinthe time limit.
A* Search_h_ignore_preconditions is the best performer. 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. For small search problems **BFS** is still better!
h_pg_levelsum uses a planning graph representation of the problem state space to estimate the sum of all actions. I found this implementation slow and expensive.

Informed vs Uninformed Search Strategies

The search strategies that generate optimal plans are BFS, UCS, and A* Search heuristics.

From the results above: **A* Search_h_ignore_preconditions** heuristic would be the best choice overall for our Air Cargo problem.

Problem 1 – simple task:

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Uninformed	breadth_first_search	0.0508	6	Yes	43	56	180
	breadth_first_tree_search	1.10976	6	Yes	1458	1459	5960
	depth_first_graph_search	0.01090	12	No	12	13	48
	depth_limited_search	0.11020	50	No	101	271	414
	uniform_cost_search	0.0436	6	Yes	55	57	224
	recursive_best_first_search	3.91439	6	Yes	4229	4230	17029
	greedy_best_first_graph_search	0.00818	6	Yes	7	9	28
Informed	A* Search with h_1	0.04690	6	Yes	55	57	224
	A* Search with h_ignore_preconditions	0.04610	6	Yes	41	43	170
	A* Search with h_pg_levelsum	1.131219	6	Yes	11	13	50

Problem 2 – more complex task:

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Uninformed	breadth_first_search	19.487	9	Yes	3346	4612	30534
	breadth_first_tree_search	N.A.					
	depth_first_graph_search	3.000*	wrong	No	TBD	TBD	TBD
	depth_limited_search	1038.369	50	No	213491	1967093	1967471
	uniform_cost_search	13.339	9	Yes	4853	4855	44041
	recursive_best_first_search	N.A.					
	greedy_best_first_graph_search	2.913	21	No	998	1000	8982
Informed	A* Search with h_1	15.063	9	Yes	4853	4855	44041
	A* Search with h_ignore_preconditions	5.6889	9	Yes	1450	1452	13303
	A* Search with h_pg_levelsum	196.639	9	Yes	86	88	841

Problem 3 – complex problem:

Search Strategy	Search Algorithm	Time Elapsed	Plan Length	Optimal	Node Expansions	Goal Tests	New Nodes
Uninformed	breadth_first_search	153.236	12	Yes	14663	18098	129631
	breadth_first_tree_search	N.A.					
	depth_first_graph_search	33.000*	wrong	No	TBD	TBD	TBD
	depth_limited_search	N.A.					
	uniform_cost_search	76.395	12	Yes	18235	18237	159716
	recursive_best_first_search	N.A.					
	greedy_best_first_graph_search	23.704	22	No	5614	5616	49429
Informed	A* Search with h_1	92.4660	12	Yes	18235	18237	159716
	A* Search with h_ignore_preconditions	28.7670	12	Yes	5040	5042	44944
	A* Search with h_pg_levelsum	1510.236	12	Yes	318	320	2934

Conclusion

The results above shows the benefits of using informed search strategies with custom heuristics over uninformed search techniques when searching for an optimal plan. The benefits are significant in speed and memory usage.