2017

HEURISTIC ANALYSIS of Planning Search Project

ARTIFICIAL INTELLIGENCE NANODEGREE, UDACITY

WALEED ALZOGHBY

waleed.alzoghby@outlook.com

Table of Contents

Introduction

In this project, I have defined a group of problems in classical PDDL (Planning Domain Definition Language) for the air cargo domain, setting up the problems for search, experiment with various automatically generated heuristics, including planning graph heuristics, to solve the problems.

So, the goal of this project to build a planning search agent to solve deterministic logistics problems for Air Cargo transport system.

Classical PDDL problems

Here is a listing describes the given problems:

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	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))

Non-Heuristic Planning Searches

As per section 3.4 (AI A modern Approach 3rd edition) Uninformed search (Non-Heuristic) means that the strategies have no additional information about states beyond that provided in the problem definition. All they can do is generate successors and distinguish a goal state from a non-goal state. All search strategies are distinguished by the order in which nodes are expanded.

I have executed the following uninformed planning searching algorithms:

1. breadth_first_search
2. depth_first_graph_search
3. uniform_cost_search

Against all provided problems Problem 1, Problem 2 and Problem 3 to calculate the following:

1. Speed (Execution time)
2. Memory Usage (Node Expansions)
3. Optimality (Optimal length)

The results will be shown in the next section.

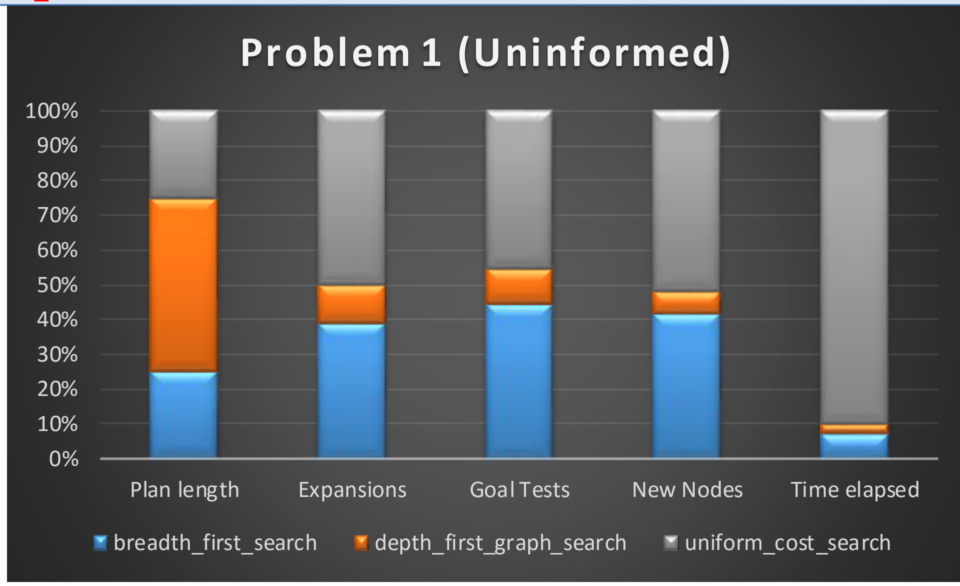
Non-Heuristic Planning Searches Results

In this section I will show the results obtained from my AI planning agent on problem 1, problem 2 and problem 3 runs.

Note: the marked **algorithms with red** color have the **optimal plans.**

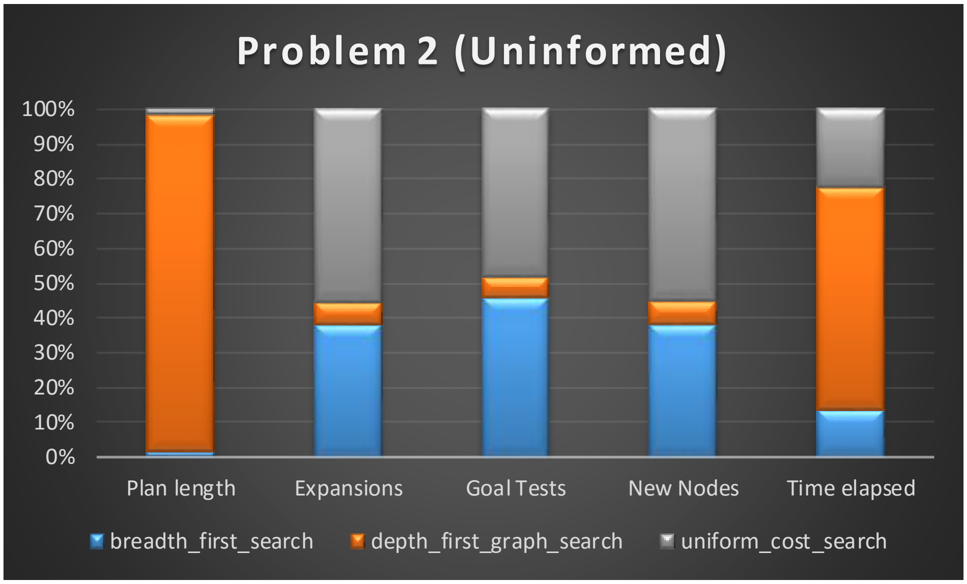
Problem 1

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
breadth_first_search	6	43	56	180	0.0301
depth_first_graph_search	12	12	13	28	0.0106
uniform_cost_search	6	55	57	224	0.375



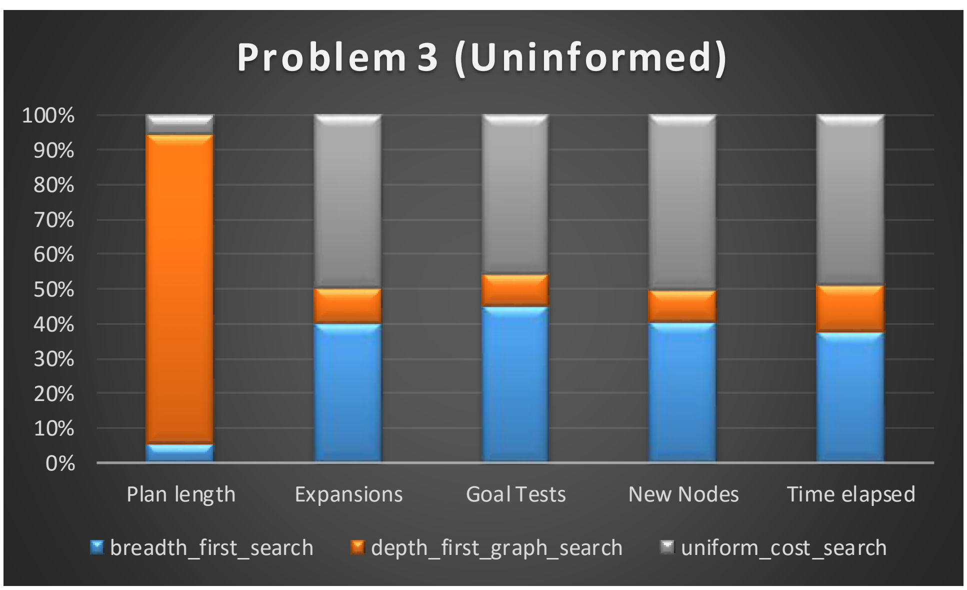
Problem 2

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
breadth_first_search	9	3343	4609	30509	12.303
depth_first_graph_search	575	582	583	5211	57.481
uniform_cost_search	9	4853	4855	44041	19.96



Problem 3

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
breadth_first_search	12	14663	18098	128566	43.649
depth_first_graph_search	194	3660	3661	29350	15.827
uniform_cost_search	12	18234	18236	158298	56.242



We can conclude from the provided results and charts the following:

1. The **optimal plans** can be obtained from "Breadth First Search" and "Uniform Cost Search"
2. The "Depth First Graph Search" **is the best** in terms of **memory usage and time execution**.
3. In terms of **fastest, less memory usage and optimal actions length** so "Breadth First Search" is the best.

My results proves the demonstration of uninformed searching algorithms comparisons in section 3.4.7 (AI A modern Approach 3rd edition)

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.

Heuristic Planning Searches

As per section 3.5 (AI A modern Approach 3rd edition) an informed search strategy—one that uses problem-specific knowledge beyond the definition of the problem itself—can find solutions more efficiently than can an uninformed strategy.

I have executed the following informed planning searching algorithms:

1. astar_search h_1
2. astar_search h_ignore_preconditions
3. astar_search h_pg_levelsum

Against all provided problems Problem 1, Problem 2 and Problem 3 to calculate the following:

4. Speed (Execution time)
5. Memory Usage (Node Expansions)
6. Optimality (Optimal length)

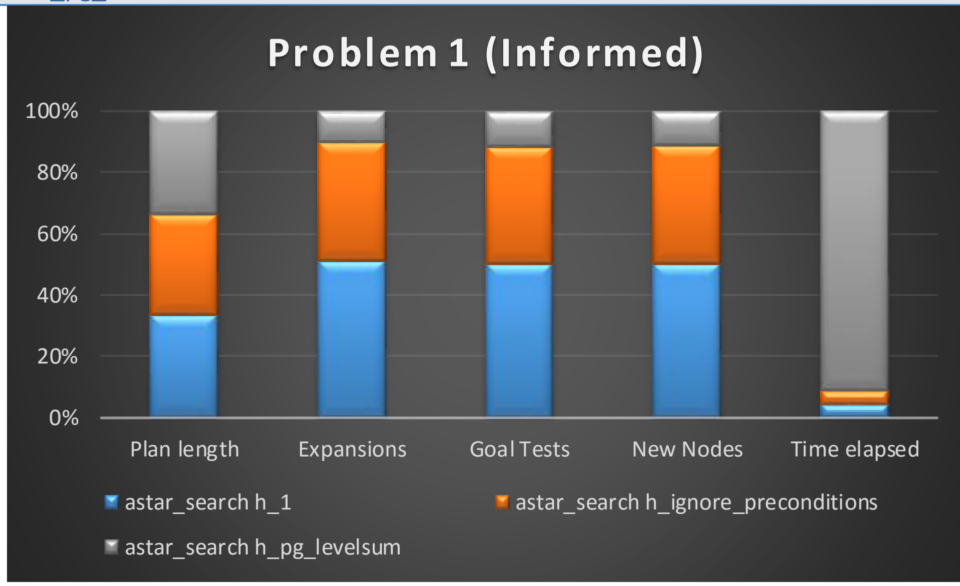
The results will be shown in the next section.

Heuristic Planning Searches Results

In this section I will show the results obtained from my AI planning agent on problem 1, problem 2 and problem 3 runs.

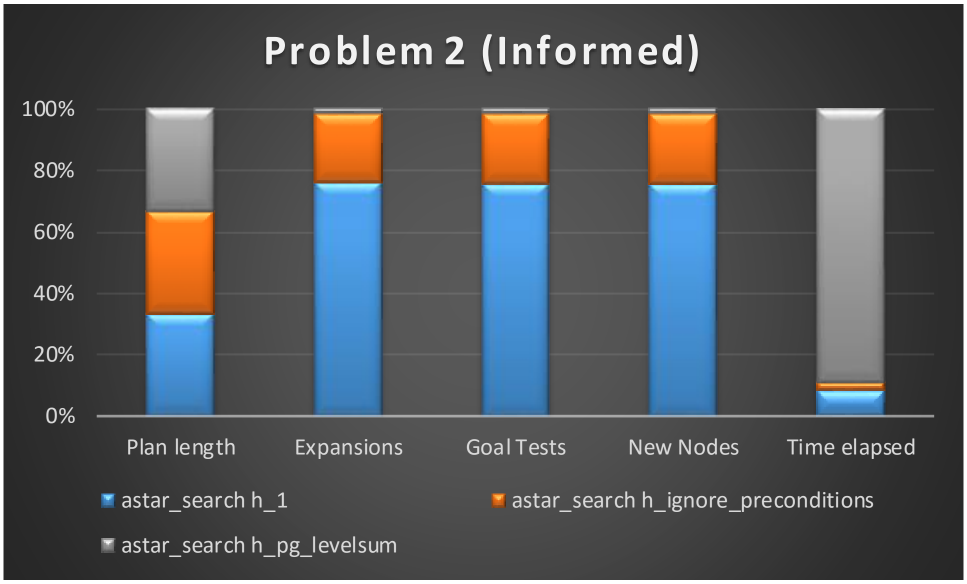
Problem 1

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
astar_search h_1	6	55	57	224	0.041
astar_search h_ignore_preconditions	6	41	43	170	0.0401
astar_search h_pg_levelsum	6	11	13	50	0.827



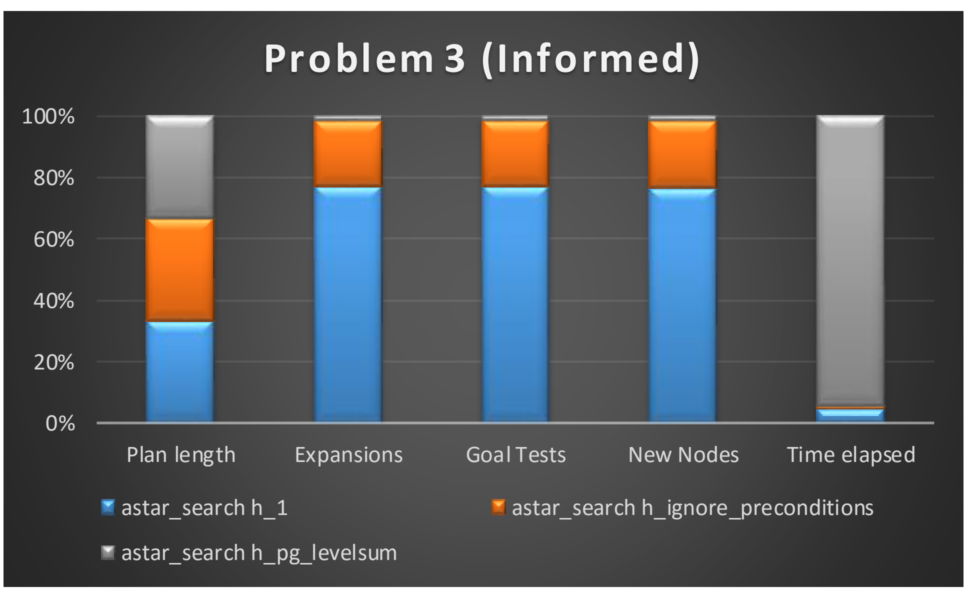
Problem 2

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
astar_search h_1	9	4853	4855	44041	18.002
astar_search h_ignore_preconditions	9	1450	1452	13303	6.106
astar_search h_pg_levelsum	9	86	88	841	196.672



Problem 3

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
astar_search h_1	12	18234	18236	158298	58.946
astar_search h_ignore_preconditions	12	5038	5040	44746	18.381
astar_search h_pg_levelsum	12	366	368	3364	1231.316



We can conclude from the provided results and charts the following:

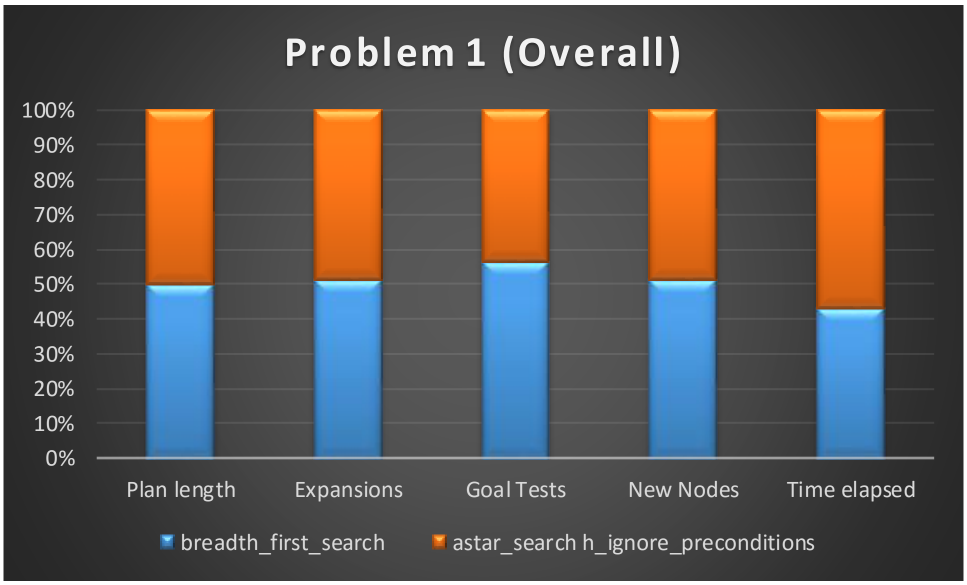
1. The **optimal plans** can be obtained from “astar_search h_1”, “astar_search h_ignore_preconditions” and “astar_search h_pg_levelsum”
2. The “astar_search h_pg_levelsum” is the **slowest heuristic** if we compared it with other heuristics execution time.
3. The **best memory usage** achieved by “astar_search h_pg_levelsum” heuristic.
4. In terms of **fastest and less memory usage** so “astar_search h_ignore_preconditions” is **the best.**

The Conclusion

We came out with some results which is “Breadth First Search” and “astar_search h_ignore_preconditions” are the best of all, so now I will do a comparison of both used algorithms to see which one is the best overall.

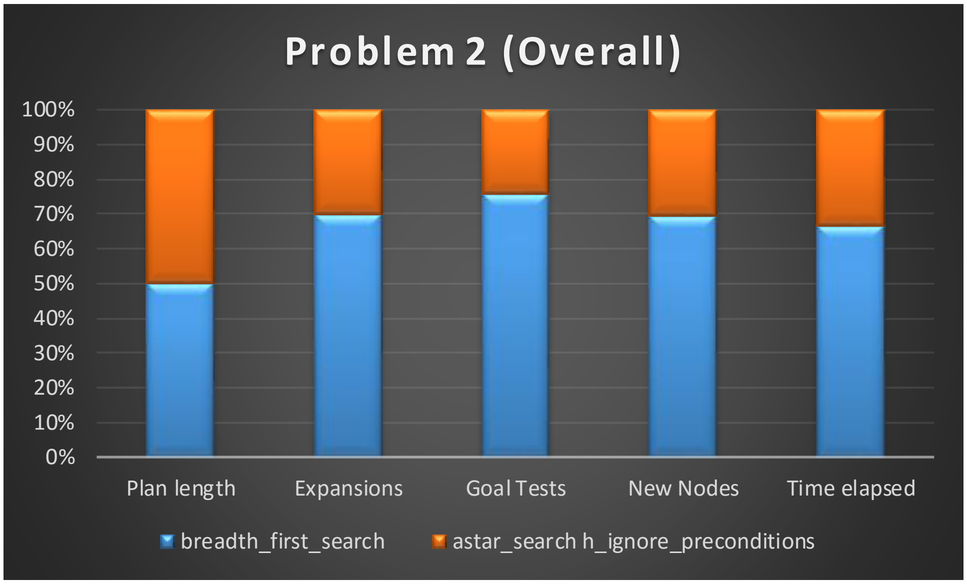
Problem 1

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
breadth_first_search	6	43	56	180	0.0301
astar_search h_ignore_preconditions	6	41	43	170	0.0401



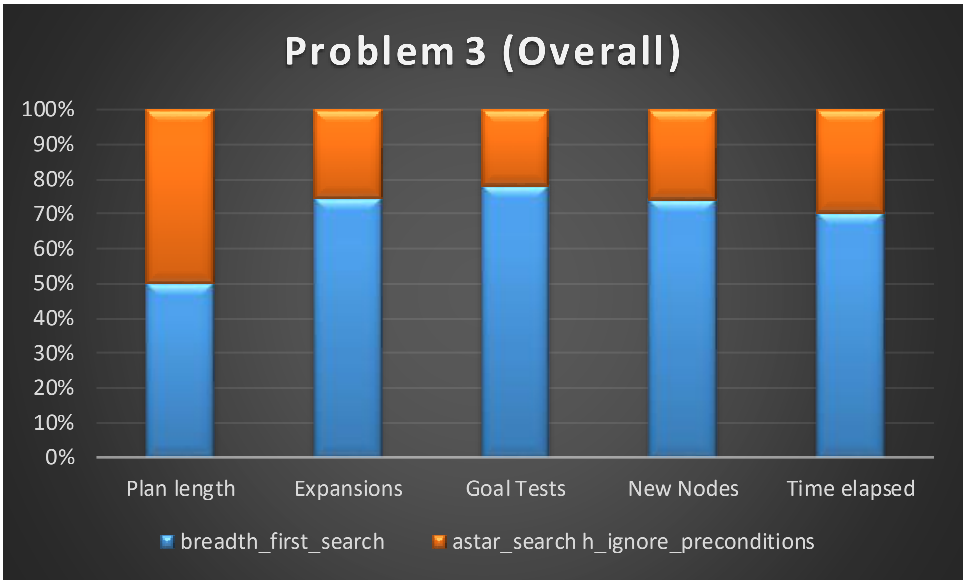
Problem 2

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
breadth_first_search	9	3343	4609	30509	12.303
astar_search h_ignore_preconditions	9	1450	1452	13303	6.106



Problem 3

Algorithm	Plan length	Expansions	Goal Tests	New Nodes	Time elapsed
breadth_first_search	12	14663	18098	128566	43.649
astar_search h_ignore_preconditions	12	5038	5040	44746	18.381



We can conclude from the provided results and charts the following:

1. The **optimal plans** can be obtained from "breadth_first_search" and "astar_search h_ignore_preconditions".
2. In terms of **fastest and less memory usage** so "astar_search h_ignore_preconditions" is **the best.**

Here is the set of actions for "breadth_first_search" and "astar_search h_ignore_preconditions" algorithms.

Algorithm	Problem 1	Problem 2	Problem 3
breadth_first_search	Load(C2, P2, JFK) Load(C1, P1, SFO) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK)	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)	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)
astar_search h_ignore_preconditions	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO)	Load(C1, P1, SFO) Fly(P1, SFO, JFK) Unload(C1, P1, JFK) Load(C2, P2, JFK) Fly(P2, JFK, SFO) Unload(C2, P2, SFO) Load(C3, P3, ATL) Fly(P3, ATL, SFO) Unload(C3, P3, SFO)	Load(C2, P2, JFK) Fly(P2, JFK, ORD) Load(C4, P2, ORD) Fly(P2, ORD, SFO) Unload(C4, P2, SFO) Unload(C2, P2, SFO) Load(C1, P1, SFO) Fly(P1, SFO, ATL) Load(C3, P1, ATL) Fly(P1, ATL, JFK) Unload(C3, P1, JFK) Unload(C1, P1, JFK)