P3 AIND Project (Part 3 – Written Analysis)

Table of Content

Problem: Air Cargo Problem 1

Results

	Problem 1				
	expansion	goal tests	new nodes	plan length	time elapsed
breadth first search	43	56	180	6	0.03203
breadth first search tree search	1458	1459	5960	6	1.0497
depth first graph search	21	22	84	20	0.01568
depth limited search	101	271	414	50	0.09604
uniform cost search	55	57	224	6	0.04418
recursive best first search h_1	4229	4230	17023	6	3.07277
greedy best first graph search h_1	7	8	28	6	0.00605
astar search h_1	55	57	224	6	0.04709
astar search h_ignore preconditions	41	43	170	6	0.05064
astar search h_pg_levelsum	11	13	50	6	1.50045

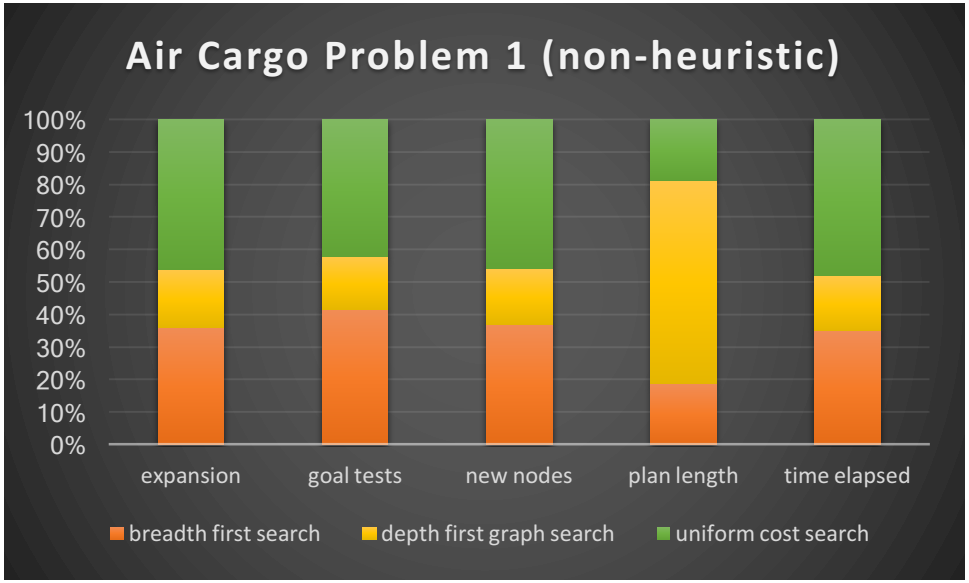
Selected methods for non-heuristic search:

1. Breadth-first search
2. Depth-first search
3. Uniform-cost search

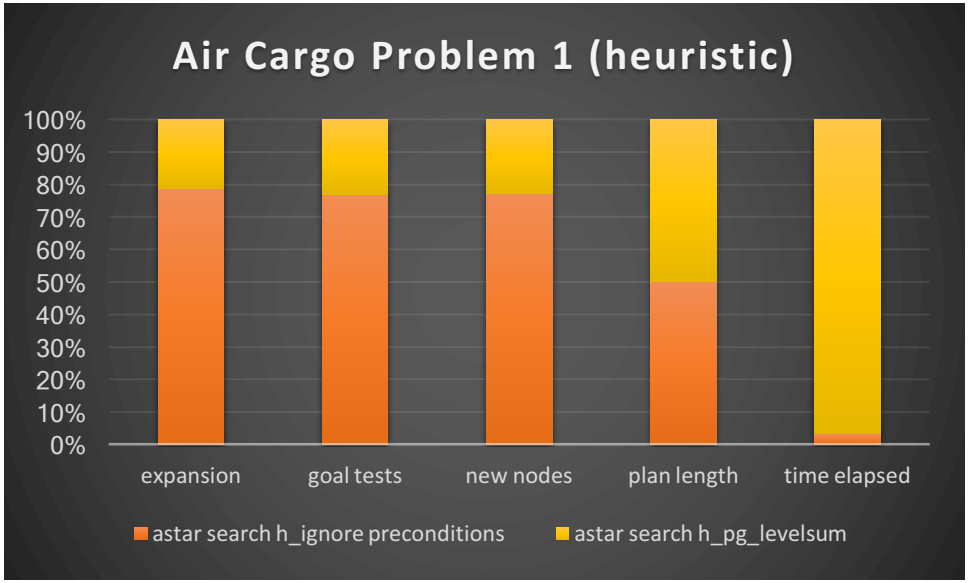
Selected methods for heuristic search:

1. A* search_ignore preconditions
2. A* search_levelsum

Stacked Bar Chart (non-heuristic search results)



Stacked Bar Chart (heuristic search results)



Optimal Plan:

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

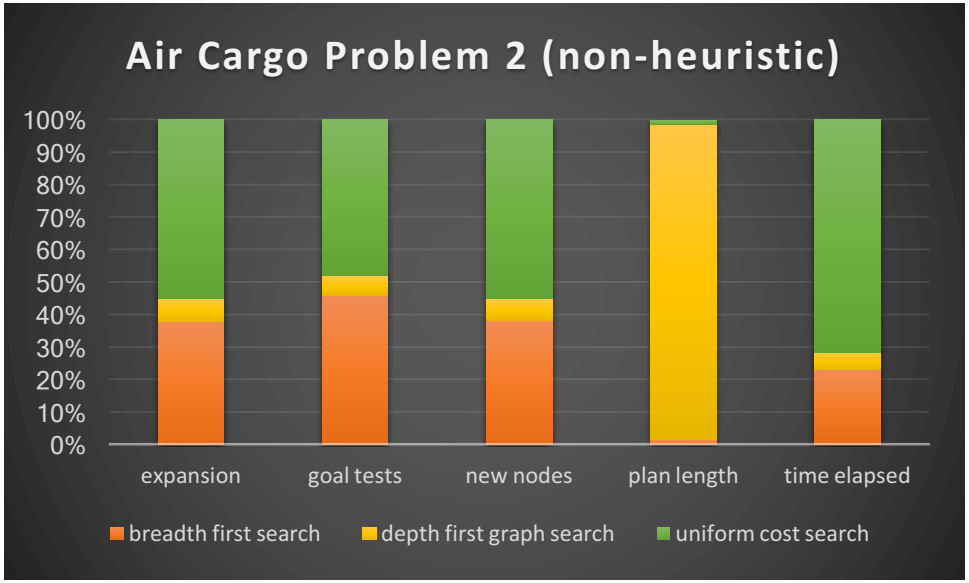
Problem: Air Cargo Problem 2

Results

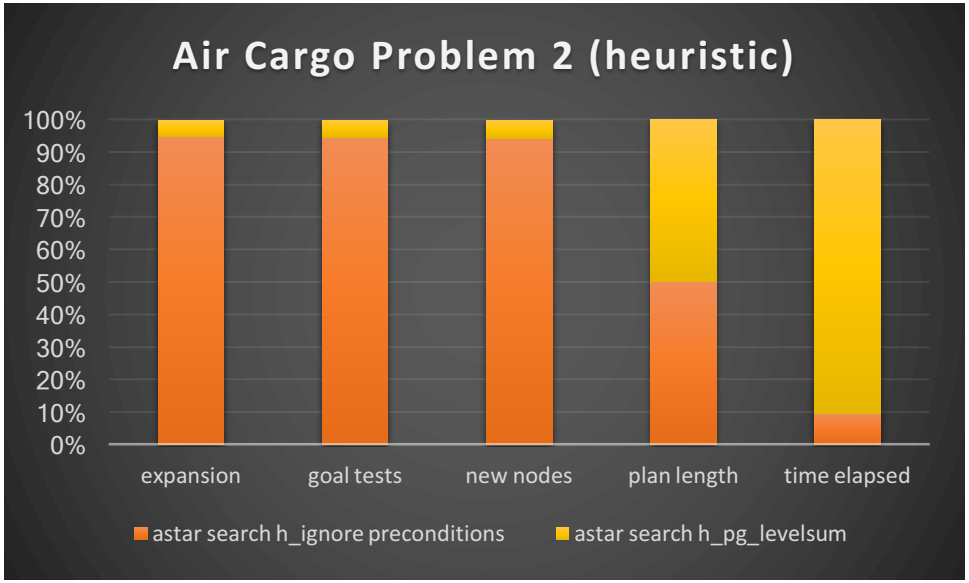
	Problem 2				
	expansion	goal tests	new nodes	plan length	time elapsed
breadth first search	3343	4609	30509	9	15.33784
breadth first search tree search	x	x	x	x	x
depth first graph search	624	625	5602	619	3.77626
depth limited search	x	x	x	x	x
uniform cost search	4853	4855	44041	9	47.89896
recursive best first search h_1	x	x	x	x	x
greedy best first graph search h_1	998	1000	8982	21	7.82641
astar search h_1	4853	4855	44041	9	48.29967
astar search h_ignore preconditions	1506	1508	13820	9	15.86626
astar search h_pg_levelsum	86	88	841	9	149.08119

Note: (x) sign means aborted due to process takes longer than 10 minutes

Stacked Bar Chart (non-heuristic search results)



Stacked Bar Chart (heuristic search results)



Optimal Plan:

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)

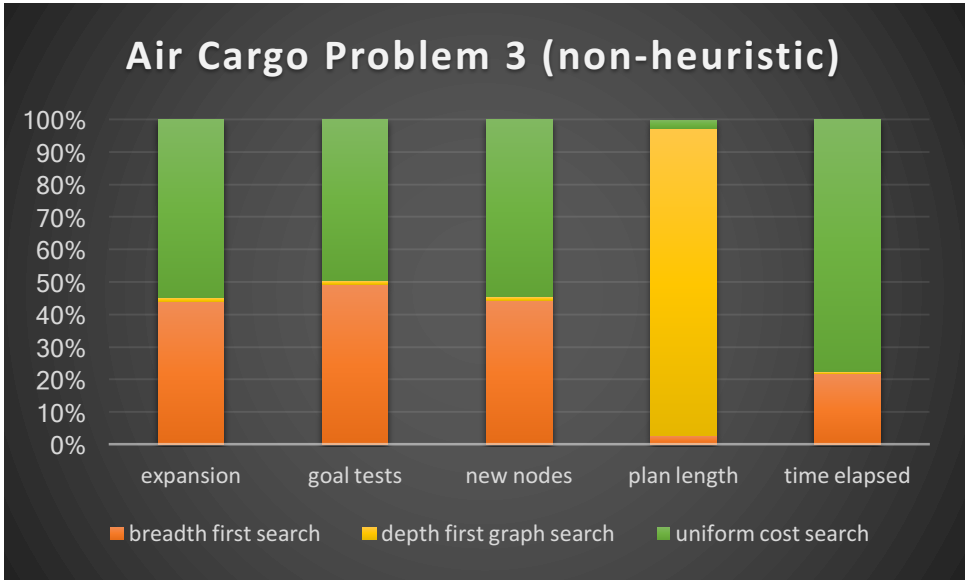
Problem: Air Cargo Problem 3

Results

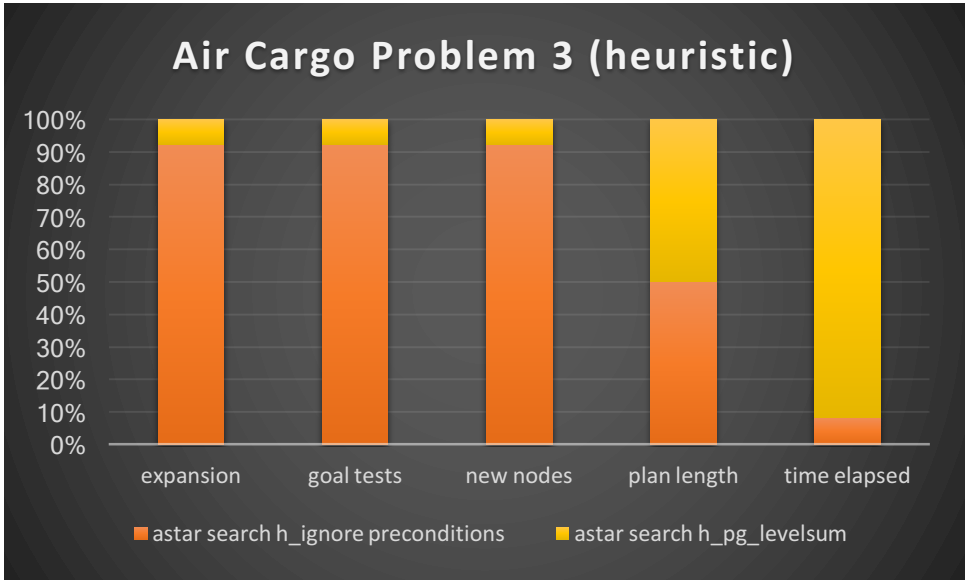
	Problem 3				
	expansion	goal tests	new nodes	plan length	time elapsed
breadth first search	14663	18098	129631	12	116.91748
breadth first search tree search	x	x	x	x	x
depth first graph search	408	409	3364	392	2.00041
depth limited search	x	x	x	x	x
uniform cost search	18234	18236	159707	12	415.145045
recursive best first search h_1	x	x	x	x	x
greedy best first graph search h_1	5605	5607	49360	22	113.16541
astar search h_1	18234	18236	159707	12	436.51399
astar search h_ignore preconditions	5118	5120	45650	12	89.38996
astar search h_pg_levelsum	414	416	3818	12	973.30732

Note: (x) sign means aborted due to process takes longer than 10 minutes

Stacked Bar Chart (non-heuristic search results)



Stacked Bar Chart (heuristic search results)



Optimal plan:

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

Analysis and Discussions

Non-heuristic search results

For non-heuristic search results, we are comparing three methods as follows:

1. Breadth-first search
2. Depth-first search
3. Uniform-cost search

The result metrics that we are measuring include expansion, goal tests, new nodes, plan length, and time elapsed.

Based on the table and charts, we can see the result metrics are very consistent among the three search methods. We can generalize our findings as follows:

1. In terms of being the quickest algorithm to get an optimal plan, depth-first search is the winner.
    a. This algorithm solves problem 1 within 0.01568 seconds (vs 0.03203 seconds using breadth-depth first and 0.04418 seconds using uniform-cost search),
    b. problem 2 within 3.77626 seconds (vs 15.33784 seconds using breadth-depth first and 47.89896 seconds using uniform-cost search), and
    c. problem 3 within 2 seconds (vs 116.91748 seconds using breadth-depth first and 415.14505 seconds using uniform-cost search).

2. However, we can argue that being the quickest may not be the most important metric. Instead, one should look into the shortest plan length because each trip is costly and one should try to minimize the length to get the lowest total cost to get an optimal plan. As it turns out, both breadth first search and uniform cost search perform equally well to solve
    a. problem 1 with plan length of 6 (vs 20 with depth-first search),
    b. problem 2 with plan length of 9 (vs 619 with depth-first search), and
    c. problem 3 with plan length of 12 (vs 392 with depth-first search).

3. Rationale:
    a. depth-first search is quick because it uses brute-force to find a plan. However, it fails to explore broader plans that may be more efficient. We can draw this conclusion based on the expansion, goal tests, and new nodes metrics. In the case of depth-first search, these metrics are much lower than breadth-first search or uniform-cost search.

b. Breadth-first depth and uniform cost depth performed almost equally. Both algorithms are able to find the most effective optimal plan (ie. shortest plan length). The similar results reflect much similarity between the two algorithms. The little difference is when the goal test is applied, where in uniform-cost search, the goal test is applied to a node when it is selected for expansion rather than when it is first generated as in breadth-first search.

Heuristic search methods

For heuristic search results, we are comparing three methods as follows:

1. A* search_ignore preconditions
2. A* search_levelsum

Similar to non-heuristic search methods, we use the same performance metrics, ie. expansion, goal tests, new nodes, plan length, and time elapsed.

Our observation between the two heuristic method finds that both “ignore preconditions” and “levelsum” heuristics are able to find the most efficient plan. The plan lengths for problem 1, 2, and 3 are equal to non-heuristics breadth-first and uniform-cost search methods.

When we compare the speed, “ignore preconditions” performed about up to 10x faster than “levelsum” heuristics, depending the complexity of the problem.

- In problem 1, “ignore precondition” took 0.05064 seconds vs “levelsum” 1.50045 seconds,
- In problem 2, “ignore precondition” took 15.86626 seconds vs “levelsum” 149.08119 seconds, and
- In problem 3, “ignore precondition” took 89.38996 seconds vs “levelsum” 973.30732 seconds,

Rationale: by ignoring the preconditions, this heuristic works much faster as any goal conditions can be achieved in one step.

Heuristic vs non-heuristic search methods

When we compare the results between heuristic and non-heuristic search methods, we can observe heuristic algorithms in general perform faster, while still giving the optimal plan. Non-heuristic forward search methods can be inefficient and require large state space. Backward search can help to expedite solution findings. However, it is hard to come up the heuristics with backward search.

As we learned from this exercise, forward search with heuristics perform better as heuristics allow to define a relaxed problem which is easier to solve.

Research Reviews

Research 1: Researchers add a splash of human intuition to planning algorithms[1]

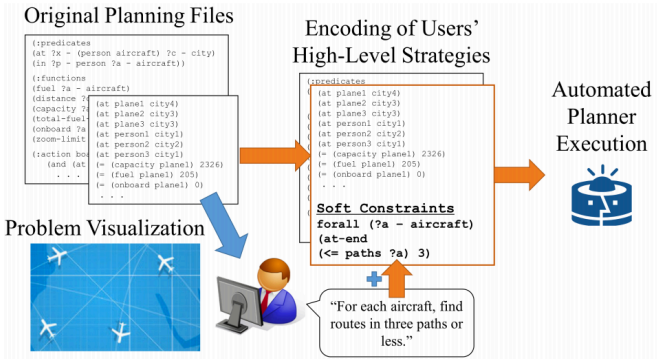
Every other year, the International Conference on Automated Planning and Scheduling hosts a competition in which computer systems designed by conference participants try to find the best solution to a planning problem. However, even the best planning algorithms still aren't as effective as human beings with a particular aptitude for problem-solving.

Researchers from MIT's Computer Science and Artificial Intelligence Laboratory are trying to improve automated planners by giving them the benefit of human intuition. Current off-the-shelf planners can take a very long time before generating a near-optimal solution. In an effort to reduce plan computation time, increase the quality of the resulting plans, and make them more interpretable by humans, the researchers explored collaborative planning techniques that actively involve human users in plan generation.

By encoding the strategies of high-performing human planners in a machine-readable form, they were able to improve the performance of competition-winning planning algorithms by 10 to 15 percent on a challenging set of problems. Specifically, they explored a framework in which users provided high-level strategies encoded as soft preferences to guide the low-level search of the planner. Through human subject experimentation, the researchers empirically demonstrated that this approach results in statistically significant improvements to plan quality, without substantially increasing computation time.

[Illustration] Human expert's "preferences" input into planning problems:

[1] "Collaborative Planning with Encoding of Users' High-level Strategies", Kim, Banks, and Shah, MIT 2017



Examples of expert's Preference inputs (ie. soft goals and soft trajectory) translation into PDDL 3.0 language:

User Strategy	Preference Encoding
"Try to fly using plane1 as much as possible"	`forall (?a - aircraft ?p - person) (always (imply (in ?p ?a) (= ?a plane1)))`
"I tried to have every passenger meet in city2"	`(sometime (forall (?p - person) (at ?p city2)))`
"Do not make any redundant trips (i.e., planes should visit each city at most once)"	`forall (?a - aircraft ?c - city) (at-most-once (at ?a ?c))`
"If plane3 visits city1 then it should visit city2"	`(sometime-after (at plane3 city1) (at plane3 city2)))`
"For each satellite, find routes in three turns or less"	`(forall (?s - satellite) (at-end (<= (turns ?s) 3))`
"Each target should have at most one satellite pointing at the same time"	`(forall ?s1 ?s2 - satellite, ?t - target) (always (implies (and (pointing ?s1 ?t) (pointing ?s2 ?t)) (= ?s1 ?s2)))`
"Whenever data storage exceeds 500, point towards the ground station within 5 time units"	`(forall (?s - satellite) (always-within 5 (>= (data ?s) 500) (pointing ?s groundstation))`

Research 2: Teaching machines to predict the future[2]

When we see two people meet, we can often predict what happens next: a handshake, a hug, or maybe even a kiss. Machines, on the other hand, have trouble making use of complex knowledge like that. Computer systems that predict actions would open-up new possibilities ranging from robots that can better navigate human environments, to emergency response systems that predict falls, to Google Glass-style headsets that feed your suggestions for what to do in different situations.

Past attempts at predictive computer-vision have generally taken one of two approaches:

1. The first method is to look at an image's individual pixels and use that knowledge to create a photorealistic "future" image, pixel by pixel
2. The second is to have humans label the scene for the computer in advance, which is impractical for being able to predict actions on a large scale.

MIT researcher instead created an algorithm that can predict "visual representations," which are basically freeze-frames showing different versions of what the scene might look like. Visual representations reveal information about the larger image, such as a certain collection of pixels that represents a human face. The team's algorithm employs "neural networks" and each of the algorithm's networks predicts a representation is automatically classified as an action. The system then merges those actions into one that it uses as its prediction. For example, three networks might predict a kiss, while another might use the fact that another person has entered the frame as a rationale for predicting a hug instead.

After training the algorithm on 600 hours of unlabeled video, the team tested it on new videos showing both actions and objects. When shown a video of people who are one second away from performing one of the four actions, the algorithm correctly predicted the action more than 43 percent of the time, which compares to existing algorithms that could only do 36 percent of the time.

In below example videos, the algorithm classifies "visual representations" into one of the four actions, ie. a hug, handshake, high-five, or kiss.

---

[2] "Anticipating Visual Representations from Unlabeled Video", Vondrick, Pirsiavach, and Torralba, MIT 2017

Research 3: Robot helps nurses schedule tasks on labor floor[3]

Today's robots in hospitals are employed to perform simple tasks such as delivering supplies and medications, but they have to be explicitly told what to do. A team from MIT's Computer Science and Artificial Intelligence Laboratory (CSAIL) thinks that this will soon change, and that robots might be most effective by helping humans perform one of the most complex tasks of all: scheduling.

MIT researchers found that a subset of workers are extremely strong schedulers, but can't easily transfer that knowledge to colleagues. A particularly tough place for scheduling are hospitals. Labor wards' head nurses have to try to predict when a woman will arrive in labor, how long labor will take, and which patients will become sick enough to require C-sections or other procedures. At Beth Israel Hospital, the head nurse has to coordinate 10 nurses, 20 patients, and 20 rooms at the same time, meaning that the number of distinct scheduling possibilities adds up to a staggering $2^{1,000,000}$, which is more than the number of atoms in the universe.

Like many AI systems, the team's robot was trained via "learning from demonstration," which involves observing humans' performances of tasks. But researchers have never been able to apply this technique to scheduling, because of the complexity of coordinating multiple actions that can be very dependent on each other. To overcome this, the team trained its system to look at

[3] "Robotic Assistance in Coordination of Patient Care", Gombolay, Shah, et al, MIT 2017

several actions that human schedulers make, and compare them to all the possible actions that are not made at each of those moments in time. From there, it developed a scheduling policy that can respond dynamically to new situations that it has not seen before. The idea is to put a robot on the labor floor and watch humans doing the different tasks as it learns to understand how to coordinate an efficient schedule.

Using the system on a Nao robot, nurses accepted the robot's recommendations 90 percent of the time. The team also demonstrated that human subjects weren't just blindly accepting advice; the robot delivered consciously bad feedback that was rejected at the same rate of 90 percent, showing that the system was trained to distinguish between good and bad recommendations.