Heuristic Analysis for Project 3

Project: Implement a Planning Search

Part 1: Planning Problems

Implement methods and functions in my_air_cargo_problems.py

- *AirCargoProblem.get_actions* method including *load_actions* and *unload_actions* sub-functions
- *AirCargoProblem.actions* method
- *AirCargoProblem.result* method
- *air_cargo_p2* function
- *air_cargo_p3* function

Function implemented in the attached my_air_cargo_problems.py. All unit tests are successful

Experiment and document metrics for non-heuristic planning solution searches

Using the run_search script for non-heuristic solution searches for the three air cargo problems, the following results are acquired.

Three methods are evaluated for the problems, namely Breadth First, Depth First and Depth-limited search. It is noteworthy, that Breadth First is expanding in more nodes and takes longer time than Depth First. Depth limited on the other side performs the worst and for Problem 3 is taking a huge amount of time that it was not calculated.

Table 1: Performance Metrics across Problems grouped by Search Method

Method	Expansions	Goal Tests	New Nodes	Plan Length	Time	
Breadth First	43	56	180	6	0.02	Problem 1
	3343	4609	30509	9	11	Problem 2
	27591	35373	386305	12	376	Problem 3
Depth First Graph	21	22	84	20	0.01	Problem 1
	624	625	5602	619	2.83	Problem 2
	1368	1369	15021	1187	9.11	Problem 3
Depth Limited	101	271	414	50	0.06	Problem 1
	222719	2053741	2054119	50	732	Problem 2

Table 2: Performance Metrics across Search Method grouped by Problem

Problem	Expansions	Goal Tests	New Nodes	Plan Length	Time	
Problem 1	21	22	84	20	0.01	Depth First Graph
	43	56	180	6	0.02	Breadth First
	101	271	414	50	0.06	Depth Limited
Problem 2	624	625	5602	619	2.83	Depth First Graph
	3343	4609	30509	9	11	Breadth First
	222719	2053741	2054119	50	732	Depth Limited
Problem 3	1368	1369	15021	1187	9.11	Depth First Graph
	27591	35373	386305	12	376	Breadth First

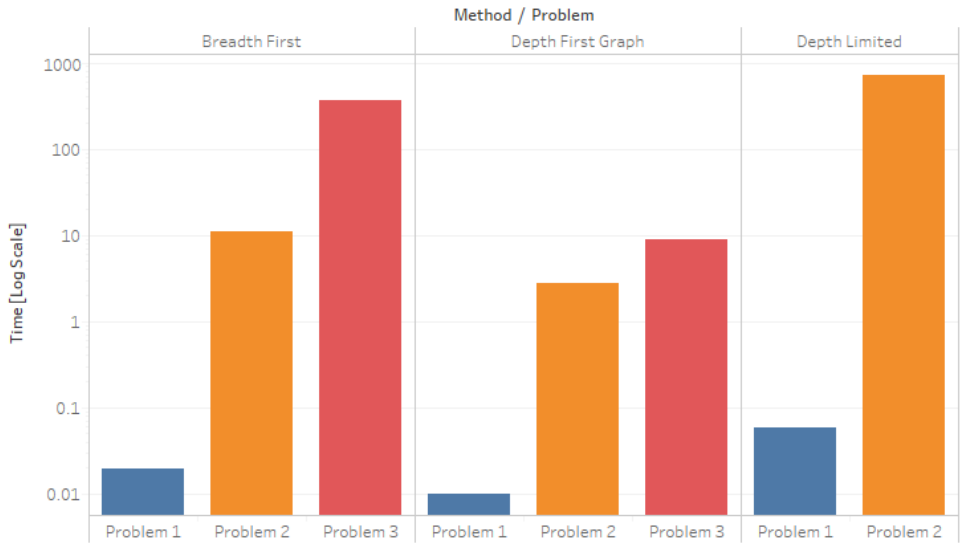

Figure 1: Time (log Scale) for each Search Method grouped by Problem

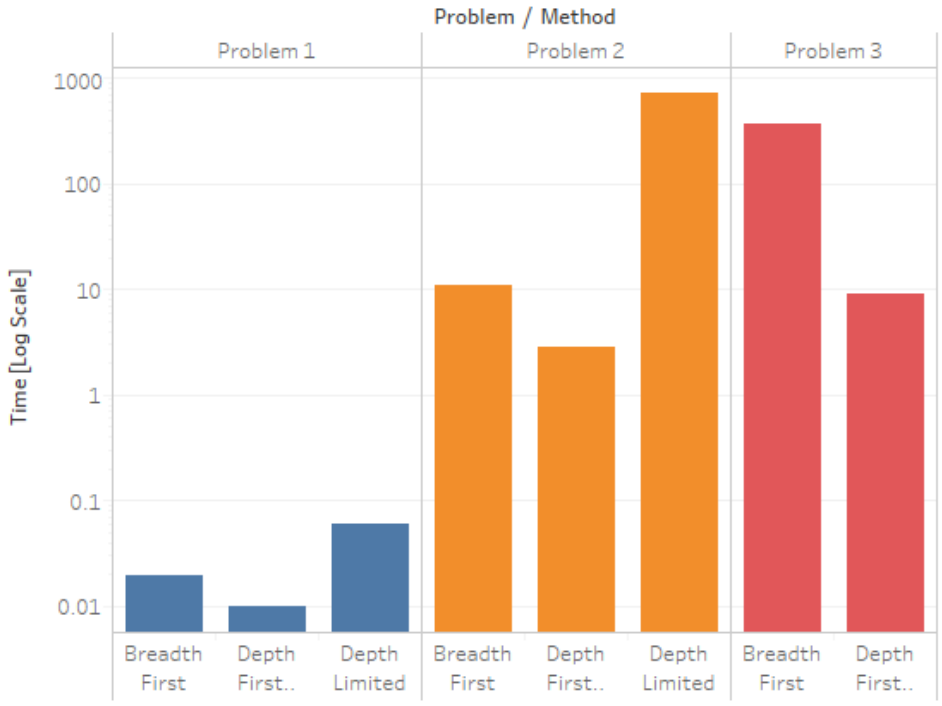

Figure 2: Time (log Scale) for each Problem grouped by Search Method

Part 2: Domain-independent heuristics

Implement heuristic method in my_air_cargo_problems.py

- *AirCargoProblem.h_ignore_preconditions method*

Implement a Planning Graph with automatic heuristics in `my_planning_graph.py`

- *PlanningGraph.add_action_level method*
- *PlanningGraph.add_literal_level method*
- *PlanningGraph.inconsistent_effects_mutex method*
- *PlanningGraph.interference_mutex method*
- *PlanningGraph.competing_needs_mutex method*
- *PlanningGraph.negation_mutex method*
- *PlanningGraph.inconsistent_support_mutex method*
- *PlanningGraph.h_levelsum method*

All methods were implemented and unit tests are successful

Experiment and document: metrics of A* searches with these heuristics

Using run search and selecting the appropriate options the following results are acquired.

Table 3: Performance Metrics across Problems grouped by A* search heuristic

Method	Expansions	Goal Tests	New Nodes	Plan Length	Time	
h ignore preconditions	41	43	170	6	0.02	Problem 1
	1506	1508	13820	9	10.6	Problem 2
h pg levelsum	11	13	50	6	8.3	Problem 1
h1	55	57	224	6	0.03	Problem 1
	4852	4854	44030	9	36.8	Problem 2

Table 4:Performance Metrics across A* search heuristic grouped by Problems

Problem	Expansions	Goal Tests	New Nodes	Plan Length	Time	
Problem 1	11	13	50	6	8.3	h pg levelsum
	41	43	170	6	0.02	h ignore preconditions
	55	57	224	6	0.03	h1
Problem 2	1506	1508	13820	9	10.6	h ignore preconditions
	4852	4854	44030	9	36.8	h1

Part 3: Written Analysis

Prior to answering the questions, for A* search with different heuristics, for Problem 3 no data are provided as the methods were taking large amount of time, more than 1000 seconds. Therefore, to conclusions will be extracted based on Problems 1 and 2. For Problem 2 also, the h_pg_levelsum heuristic was time consuming.

- Provide an optimal plan for Problems 1, 2, and 3.
    - Depth first search is achieving the solution for all the three problem in the smallest amount of time. It is noteworthy that for Problem three is one order of magnitude smaller that the breadth – first search.
- Compare and contrast non-heuristic search result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3. Include breadth-first, depth-first, and at least one other uninformed non-heuristic search in your comparison; Your third choice of non-heuristic search may be skipped for Problem 3 if it takes longer than 10 minutes to run, but a note in this case should be included.
    - The results are summarized in Tables 1 and 2. Depth First search is achieving the fastest time with the least expansions.
- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics for Problems 1, 2, and 3.
    - Ignore preconditions for the available data, is performing the best. It is puzzling in my approach the huge amount of time required for Problem 1 when using the levelsum heuristic. I ‘d rather conclude that it has to do with the implementation that being a real issue because it is 2 orders of magnitude larger than the other approaches.
- What was the best heuristic used in these problems? Was it better than non-heuristic search planning methods for all problems? Why or why not?
    - Best heuristic is h_pg_ignore_preconditions. For the Problems 1 and 2 that data is available it is worse than both breadth and depth first non-heuristic searches. I would expect the heuristic to be faster than the non-heuristic but it is possible that this has to do with the nature of the problem.
- Provide tables or other visual aids as needed for clarity in your discussion.
    - See above