Classical Planning Report

Author: Cat Chenal

Introduction

Planning problems

In this project, the planning problem is that of Air Cargo Transport: given some cargo, a plane, an origin, a destination, a start state, and a goal, we want to find a way to achieve the goal optimally or according to some search heuristics. Once we deal with multipicity in the parameters, the search space of the possible actions to take become astronomical, hence infeasible even with a powerful computer. Thus, we rely on programming to implement a planning search agent to solve the problem with different approaches.
The implementation relies on the Planning Domain Definition Language (PDDL), a framework that allows a planner to codify all the needed parameters using first-order-logic:

- The initial (known) state: what has to be true in order to have a valid problem, e.g. at least one empty plane!
- The actions that are available in the state: e.g., [Un]LoadCargo, Fly, etc.
- The result of applying the action: e.g.: flying Plane1 to destination means no Plane1 at origin airport.
- The goal state: Cargo at destination.

Several of two types of searches are implemented: Uninformed and Informed. Uninformed searches only know/check whether a given state is the stated goal or not, while Informed searches employ a heuristic, or strategy, to ween out unacceptable states, and thus can apply to more complex problems.

This report analyzes Project 2 of Udacity AI Nanodegree, in which the `my_planning_graph.py` module was completed with the required code.
After successful testing, the data files for all four air cargo problems was generated as tab-separated files using modifications of the existing code.

- `run_report.py` : a copy of `run_search.py` that uses a modified `main()` function
- `_utils.run_search_rpt()` : a new function that changes the output of `_utils.run_search()`

This report will answer three questions:

1. Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?
2. Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)?
3. Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

Analysis of the output of Problems 1 and 2

All four problems differ by the number of nodes, cargo loads, airports, and specificity of goals. The Complexity column holds a problem's total number of parameters:

Air cargo problem	Cargos	Planes	Airports	Goal
1	2	2	2	2
2	3	3	3	3
3	4	2	4	4
4	5	2	4	5

The first two problems are the less complex and are used to decide on the appropriate choices of search functions and heuristic on more "real life" problems (Problems 3 and 4).
The analysis relies on three charts:

- Number of nodes expanded against number of actions in the domain
- Search time against the number of actions in the domain
- Length of the plans returned by each algorithm on all search problems

* Complexity differences

There is only one order of magnitude difference in the number of Actions between Problem 2 and Problem 1 (72 vs. 20), but this yields multiple orders of magnitude differences for both New Nodes and Search Time.

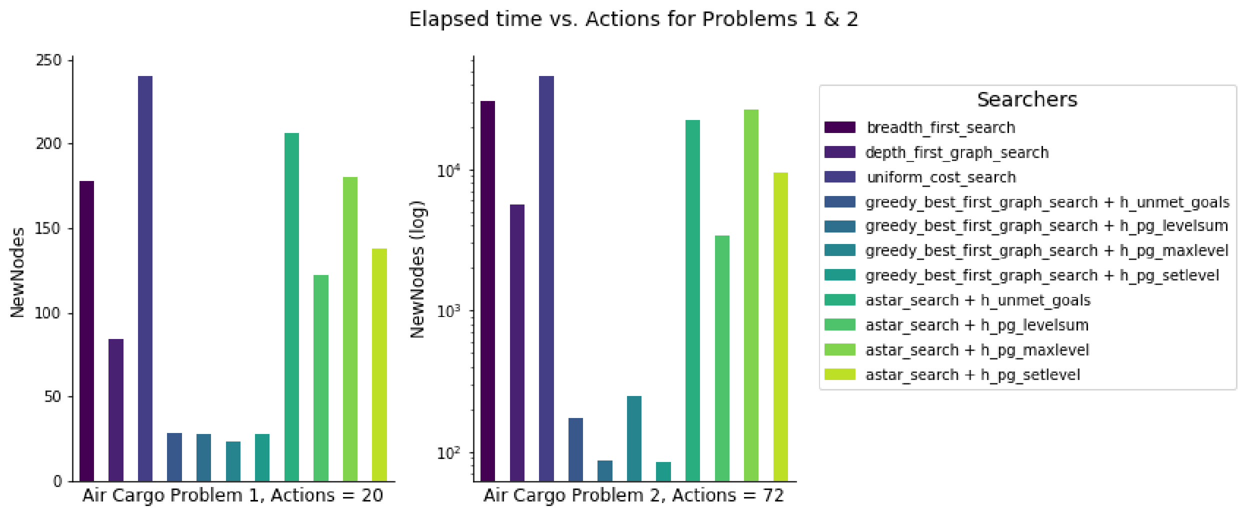

Figure 1 - NewNodes expanded in each search function for Problems 1 and 2 (log scale).

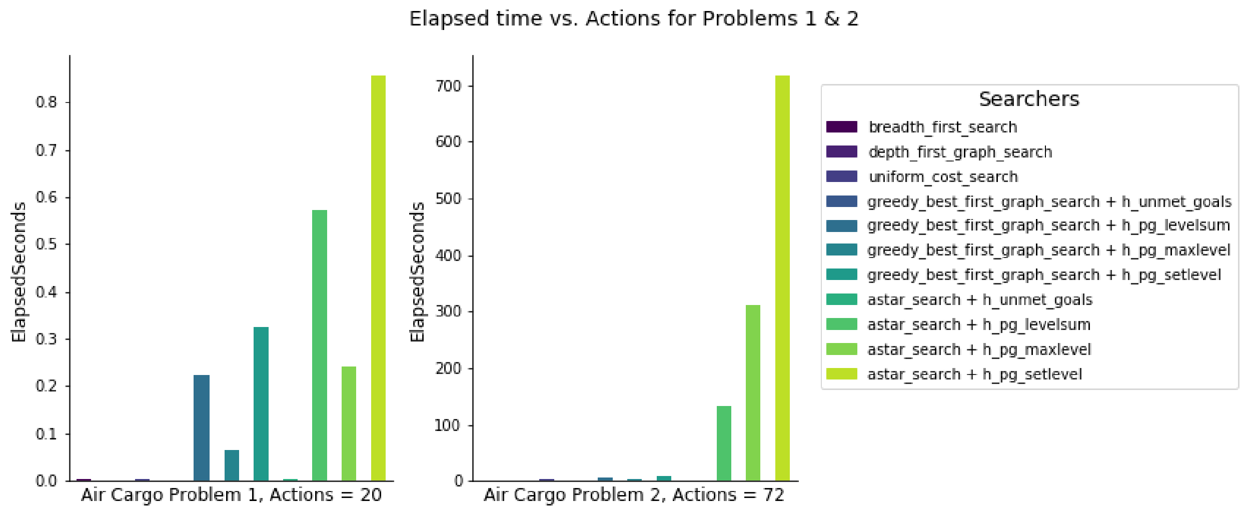

Figure 2 - ElapsedSeconds in each search function for Problems 1 and 2.

* Insights from Problems 1 and 2

On the basis of Figures 1 and 2, which show the number of new nodes created, and the time spent by each search function, respectively, the searches that are candidates for elimination for more complex problems are those at the intersection of the average-ranked costliest sets viz new nodes creation and search time.
These searches are:

```
10: astar_search h_pg_maxlevel
 8: astar_search h_unmet_goals
 3: uniform_cost_search
```

Air cargo problems 3 and 4:

After elimination of problems 3, 8, and 10, the searches performed on both Problems 3 & 4 were the following
(with the number corresponds to the number passed to `run_report.py` in the -s argument):

```
Uninformed searches
  1: breadth_first_search
  2: depth_first_graph_search
Informed searches with two different heuristics
  4: greedy_best_first_graph_search + h_unmet_goals
  5: greedy_best_first_graph_search + h_pg_levelsum
  6: greedy_best_first_graph_search + h_pg_maxlevel
  7: greedy_best_first_graph_search + h_pg_setlevel
  9: astar_search + h_pg_levelsum
  11:  astar_search + h_pg_setlevel
```

Note: Search 11 was aborted on both problems due to excessive run time (> 1 hour).
The raw data files were generated at the comman line using the `run_report.py` module, which is a copy of the original `run_search.py` module with modifications on output string formats. Note that `_utils.py.run_search` was amended accordingly.

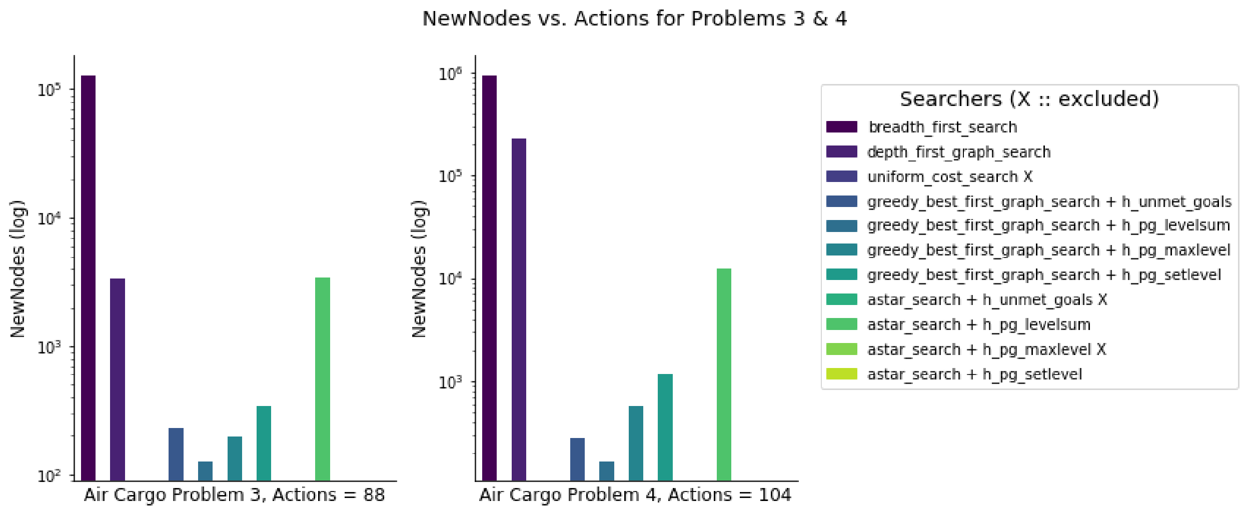

Figure 3 - NewNodes expanded in each search function for Problems 3 and 4.

`Search 11, A* with setlevel heuristic: aborted after 1 hour.`

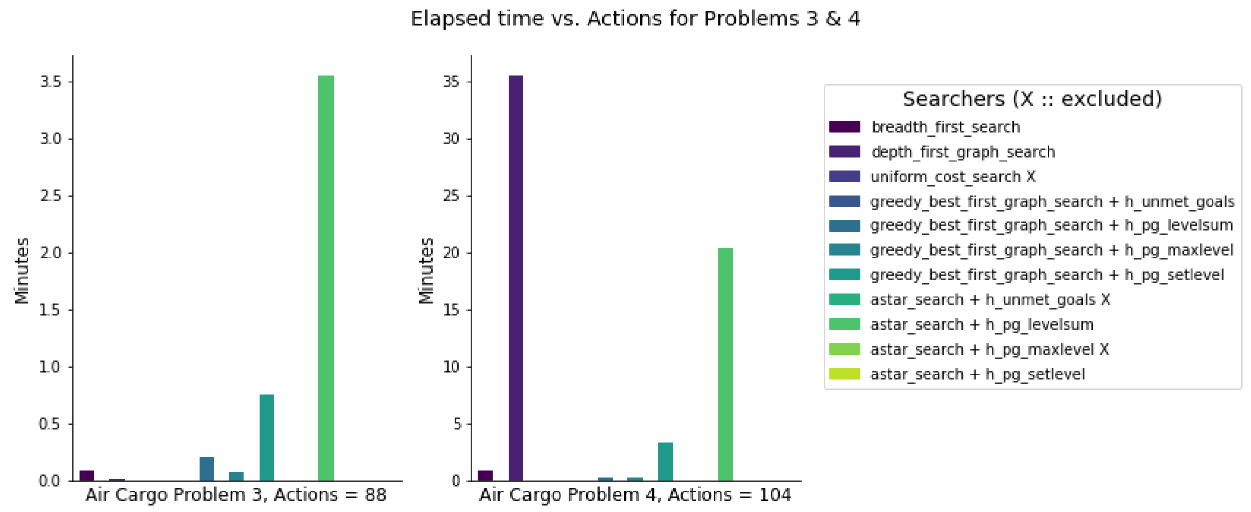

Figure 4 - Elapsed minutes in each search function for Problems 1 and 2.

`Search 11, A* with setlevel heuristic: aborted after 1 hour.`

Search function	Frequency where PlanLength >=10
greedy_best_first_graph_search	8
depth_first_graph_search	4
breadth_first_search	2
astar_search	2

Search function	Frequency where PlanLength <10
greedy_best_first_graph_search	8
astar_search	8
uniform_cost_search	2
breadth_first_search	2

Out of 36 completed searches, 44% (16), have double-digit or longer PlanLength.
In that subset, 12 (75%) involve the search functions `greedy_best_first_graph_search` and `depth_first_graph_search`. And this occurs for all Problems.

Out of 36 completed searches, 56% (20), have single-digit or longer PlanLength.
In that subset, 16 (80%) involve the search functions `greedy_best_first_graph_search` and `astar_search`. And this occurs only for Problems: 2,1.

* Anwers to questions

Question 1:

Which algorithm(s) would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

» Answer:

A domain with few actions would be like Problem 1. All search algorithms would be appropriate in this case as they all performed under 1 second.

Question 2:

Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day.)

» Answer:

For problems with more than 100 and with a search time restricted to under 1 second, algorithm **4**: *greedy_best_first_graph_search h_unmet_goals* would be the best.

Question 3:

Which algorithm(s) would be most appropriate for planning problems where it is important to find only optimal plans?