Project 3 – Heuristic Analysis

Part 1:

	air_cargo_p1	**air_cargo_p2**	**air_cargo_p3**
breadth_first_search	Command: *python run_search.py -p 1 -s 1* Node expansions: *43* No of goal tests: *56* Time elapsed: *0.03735 sec* Optimality (Plan length): *6*	Command: *python run_search.py -p 2 -s 1* Node expansions: *3343* No of goal tests: *4609* Time elapsed: *15.23275 sec* Optimality (Plan length): *9*	Command: *python run_search.py -p 3 -s 1* Node expansions: *8836* No of goal tests: *11405* Time elapsed: *51.49606 sec* Optimality (Plan length): *12*
depth_first_graph_search	Command: *python run_search.py -p 1 -s 3* Node expansions: *12* No of goal tests: *13* Time elapsed: *0.01098 sec* Optimality (Plan length): *12*	Command: *python run_search.py -p 2 -s 3* Node expansions: *582* No of goal tests: *583* Time elapsed: *3.49760 sec* Optimality (Plan length): *575*	Command: *python run_search.py -p 3 -s 3* Node expansions: *1292* No of goal tests: *1293* Time elapsed: *3.75252 sec* Optimality (Plan length): *875*
uniform_cost_search	Command: *python run_search.py -p 1 -s 5* Node expansions: *55* No of goal tests: *57* Time elapsed: *0.04314 sec* Optimality (Plan length): *6*	Command: *python run_search.py -p 2 -s 5* Node expansions: *4852* No of goal tests: *4854* Time elapsed: *49.75307 sec* Optimality (Plan length): *9*	Command: *python run_search.py -p 3 -s 5* Node expansions: *11484* No of goal tests: *11486* Time elapsed: *205.52410 sec* Optimality (Plan length): *12*

Part 2:

	air_cargo_p1	**air_cargo_p2**	**air_cargo_p3**
astar_search h_1	Command: *python run_search.py -p 1 -s 8* Node expansions: *55* No of goal tests: *57* Time elapsed: *0.06273 sec* Optimality (Plan length): *6*	Command: *python run_search.py -p 2 -s 8* Node expansions: *4852* No of goal tests: *4854* Time elapsed: *52.58176 sec* Optimality (Plan length): *9*	Command: *python run_search.py -p 3 -s 8* Node expansions: *11484* No of goal tests: *11486* Time elapsed: *192.84211 sec* Optimality (Plan length): *12*
astar_search h_ignore_preconditions	Command: *python run_search.py -p 1 -s 9*	Command: *python run_search.py -p 2 -s 9*	Command: *python run_search.py -p 3 -s 9*

	Node expansions: *41* No of goal tests: *43* Time elapsed: *0.04487 sec* Optimality (Plan length): *6*	Node expansions: *1506* No of goal tests: *1508* Time elapsed: *16.96093 sec* Optimality (Plan length): *9*	Node expansions: *2494* No of goal tests: *2496* Time elapsed: *27.55574 sec* Optimality (Plan length): *12*
astar_search h_pg_levelsum	Command: *python run_search.py -p 1 -s 10* Node expansions: *11* No of goal tests: *13* Time elapsed: *3.55790 sec* Optimality (Plan length): *6*	Command: *python run_search.py -p 2 -s 10* Node expansions: *86* No of goal tests: *88* Time elapsed: *510.79722 sec* Optimality (Plan length): *9*	Command: *python run_search.py -p 3 -s 10* It took over 10 minutes and thus I stopped the program.

Part 3:

Optimal plan

air_cargo_p1: BREADTH_FIRST_SEARCH
air_cargo_p2: BREADTH_FIRST_SEARCH
air_cargo_p3: ASTAR_SEARCH H_IGNORE_PRECONDITIONS

Discussion

The best heuristic used is the “ignore preconditions”. But it is not always better than non-heuristic search planning methods for all problems. It only outperforms in air_cargo_p3 problem.

For air_cargo_p1 and air_cargo_p2, the state spaces to be searched are small, i.e. 2^{12} and 2^{27} respectively. It is still efficient enough to go through the whole state space to find optimal path. So using heuristic doesn’t help.

But when it comes to air_cargo_p3, the state space grows to as large as 2^{32}. It is not efficient to go through the whole state space. So heuristic can provide a good direction of how the search goes towards the goal state, without going through the whole state space.

In particular, “ignore preconditions” works much faster than “level-sum” because “level-sum” requires more computation power than “ignore preconditions”. “level-sum” need to go through multiple state levels and thus it may potentially go through many state nodes for each goal state. But “ignore preconditions” only need to go through the current state level, which is finite set of state nodes.