AIND-Planning Heuristic Analysis

1. Results

The results of problems 1, 2, and 3 using uninformed planning searches and A* planning searches displays as following tables.

Uniformed planning searches (Non-heuristic)

Search Type	Expansions	Goal Tests	New Nodes	Time (s)	Length	Optimal
breadth_first_search	43	56	180	0.0363	6	True
depth_first_graph_search	21	22	84	0.0143	20	False
uniform_cost_search	55	57	224	0.0375	6	True

air_cargo_p1 results

Search Type	Expansions	Goal Tests	New Nodes	Time (s)	Length	Optimal
breadth_first_search	3343	4609	30509	12.5926	9	True
depth_first_graph_search	624	625	5602	3.9228	619	False
uniform_cost_search	4853	4855	44041	18.2347	9	True

air_cargo_p2 results

Search Type	Expansions	Goal Tests	New Nodes	Time (s)	Length	Optimal
breadth_first_search	14663	18098	129631	71.0049	12	True
depth_first_graph_search	408	409	3364	2.4437	392	False
uniform_cost_search	18235	18237	159716	91.8150	12	True

air_cargo_p3 results

A* planning searches (Heuristic)

Search Type	Expansions	Goal Tests	New Nodes	Time (s)	Length	Optimal
astar_search h_1	55	57	224	0.0401	6	True
astar_search h_ignore_preconditions	41	43	170	0.0402	6	True
astar_search h_pg_levelsum	11	13	50	0.8650	6	True

air_cargo_p1 results

Search Type	Expansions	Goal Tests	New Nodes	Time (s)	Length	Optimal
astar_search h_1	4853	4855	44041	17.8084	9	True
astar_search h_ignore_preconditions	1450	1452	13303	6.5433	9	True
astar_search h_pg_levelsum	86	88	841	146.8179	9	True

air_cargo_p2 results

Search Type	Expansions	Goal Tests	New Nodes	Time (s)	Length	Optimal
astar_search h_1	18234	18237	159716	91.9590	12	True
astar_search h_ignore_preconditions	5040	5042	44944	27.4414	12	True
astar_search h_pg_levelsum	318	320	2934	849.9812	12	True

air_cargo_p3 results

2.Analysis

Optimal plan

Optimal plan is the plan which achives goal with the least actions possible. So from above results tables we can find that:

- **Problem1**: **six** length plan is optimal. Two of non-heuristic searches are found a six length optimal plan(**breadth_first_search** and **uniform cost search**). All of the heuristic searches found a six length optimal plan(**astar_search h_1**, **astar_search h_ignore_preconditions** and **astar_search h_pg_levelsum**). For instance, **breadth_first_search** has optimal sequence of actions:

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

- **Problem2**: **nine** length plan is optimal. Two of non-heuristic searches are found a six length optimal plan(**breadth_first_search** and **uniform cost search**). All of the heuristic searches found a six length optimal plan(**astar_search h_1**, **astar_search h_ignore_preconditions** and **astar_search h_pg_levelsum**). For instance, **breadth_first_search** has optimal sequence of actions:

  ```
  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)
  ```

- **Problem3**: **twelve** length plan is optimal. Two of non-heuristic searches are found a six length optimal plan(**breadth_first_search** and **uniform cost search**). All of the heuristic searches found a six length optimal plan(**astar_search h_1**, **astar_search h_ignore_preconditions** and **astar_search h_pg_levelsum**). For instance, **astar_search h_ignore_preconditions** has optimal sequence of actions:

  ```
  Load(C2, P2, JFK)
  Fly(P2, JFK, ORD)
  Load(C4, P2, ORD)
  Fly(P2, ORD, SFO)
  Unload(C4, 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)
  Unload(C2, P2, SFO)
  ```

The **depth_first_graph_search** always could not reach an optimal plan and the reason we will discuss further.

The best non-heuristic search and heuristic search

Based on the optimal length, the best search of non-heuristic and heuristic must be considered **time elapsed**. By checking the seaches tables we can pick the **breadth_first_search** (optimal and the least time elapsed in all problems) is the best non-heuristic search and the **astar_search h_ignore_preconditions** (optimal and the least time elapsed in most of problems) is the best non-heuristic search.

Other discussions

- Compare and contrast non-heuristic search (**breadth_first_search(BFS)** ,**depth_first_graph_search(DFGS)** and **uniform_cost_search(UCS)** result metrics (optimality, time elapsed, number of node expansions) for Problems 1,2, and 3：
    - **optimality**: **BFS** and **UCS** is optimal in all the problems since both searches wise and return the optimal (first goal-state encounter) value for even an infinite state space. But **DFGS** is not in the task. It returns a plan but it could be not the optimal because it does not consider if a node is better than another, it simply explores the nodes that take it as deep as possible even if the goal is not optimal.
    - **time elapsed**: **BFS** finds goal faster than **UCS** and the **DFGS** is the fastest of the three.
    - **node expansions**: **UCS** expands more nodes than **BFS** and the **DFGS** is the least of the three. **UCS** takes longer and expands more nodes than **BFS** since even after finding a path to the goal state it continues searching to try and find a cheaper path that also reaches the goal state. **DFGS** is the fastest and the least nod expansion because **DFGS** only uses a little storage spaces when not tracking the explored set and not comparing length and cost.
- Compare and contrast heuristic search result metrics using A* with the "ignore preconditions" and "level-sum" heuristics (**A* H1**, **A*HIP** and **A* HPGLS**)for Problems 1, 2, and 3:
    - **optimality**: (**A* H1**, **A*HIP** and **A* HPGLS** have found an optimal plan for the three problems. That
      happens because they are admissible, each h function doesn’t overestimate the distance to goal.
    - **time elapsed**: **A*HIP** finds goal faster than **A* H1**(besides, in problem 1 they spent similar time) and the **A* HPGLS** is the **slowest** of the three. The **A* HPGLS** heuristic uses a Planning Graph but just estimates the sum of all actions that must be carried out from the current state to satisfy each individual goal condition so that it is time consuming most likely due to the heuristic does tons of computation. But the **A*HIP** heuristic fnds the goal faster since it ignores preconditions required for an action to be executed to make the problem easier than **A* H1**.
    - **node expansions**: **A* H1** expands more nodes than **A*HIP** and the **A* HPGLS** is the least of the three. **A*HIP** that minimises the set of actions to remove more redundant actions from consideration than **A* H1** does. The **A* HPGLS** heuristic uses a Planning Graph but just estimates the sum of all actions that must be carried out from the current state to satisfy each individual goal condition so that the strict conditions makes the satisfied nodes is less than the other two.
- 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?
    - From above analysis, we can see that for problem 1 the best search planning method (cosider the optimal and time elapsed) is the **breadth_first_search (non-heuristic)** and for problem 2 and 3 the best search planing method (cosider the optimal and time elapsed) is the **astar_search h_ignore_preconditions (heuristic)**.
    - So when the problem is simple and the searching state space is small the heuristic is not better than the non-heuristic searching planning methods. The reason is that when the problem is

  simple, the search planning method can quickly find the solution from the small state space using the limited actions (such as greedily searching). If using the heuristic search planning methods, it adds checking procedure (guess how many actions are needed to achieve all the goals) and make the efficiency reduced.
  - As the complexity of the problems increase, it might be worth to consider a heuristic based approach such as “A* Search with ‘h_ignore_preconditions’” can outperform breadth first search and thus be used instead. The reason is that when the problem is complex, the heuristic can remove redundant actions and let the enormous searching space decreased to making the search is efficient.

3. Reference

Stuart J. Russell, Peter Norvig (2010), Artificial Intelligence: A Modern Approach (3rd Edition).