AIND Project Planning – Heuristic Analysis

- Abstract
  - In these 3 problems, if we would like to get an optimal plan, A* search with ignored_precondition is better than using non-heurisitc search like Breadth-first search and uniform-cost search. Moreover, if the search space is an issue, A* search with level sum heuristic would be the best choice because this heuristic estimates the path cost well than other heuristics.
- Metrics for non-heuristic planning solution searches
  - Air Cargo Problem 1
    - Initial & Goal State:

```
Init(At(C1, SFO) ∧ At(C2, JFK)
       ∧ At(P1, SFO) ∧ At(P2, JFK)
       ∧ Cargo(C1) ∧ Cargo(C2)
       ∧ Plane(P1) ∧ Plane(P2)
       ∧ Airport(JFK) ∧ Airport(SFO))
Goal(At(C1, JFK) ∧ At(C2, SFO))
```

- Search Strategies
  - Search Results Overview (sorted by Time elapsed)

	n of nodes expanded	n of goal tests	Time elapsed	Plan Length
Depth first graph search	21	22	0.01441	20
Breadth first search	43	56	0.03664	6
Uniform-cost search	55	57	0.03923	6
Depth-limited search	101	271	0.09571	50
Breadth first tree search	1458	1459	0.97802	6

Priya: Very neat comparison of all the different search results on problems 1, 2 and 3.

- In Depth first graph search, the actions are shown below:

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

  - In terms of the time elapsed, depth first graph search spent less least time to reach goal state. As we see the plan, however, it seems that there are some redundant (not effective) actions (total 20 actions), which would indicate that the optimality of this search isn't well.

Priya: Suggestion: The link below has an interesting comparison of the BFS and DFS methods on when to chose
one vs the other : http://stackoverflow.com/questions/3332947/when-is-it-practical-to-use-dfs-vs-bfs

- In Breadth first search, the actions are shown below:

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

- With the breadth-first search, there are only 6 actions in the plan, which seems to be the optimal plan for this problem. Using breath-first search, we'll expand the nodes in the same level until we reach the goal. It means we would spend more time to reach goal than depth-first search, but it will return a more optimal plan than by using depth-first graph search. That is, the optimality for breadth-first search is better than depth-first graph search.

- In Uniform cost search, the actions are shown below:

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

Priya: Well done

- With this search, it will choose a state whose path cost is the lowest in the frontier. As a result, it will certainly return a plan that is optimal for this problem, which also indicates that it wouldn't over-estimate the cost.

➢ Air Cargo Problem 2

✧ Initial & Goal State:

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL)
        ∧ At(P1, SFO) ∧ At(P2, JFK) ∧ At(P3, ATL)
        ∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3)
        ∧ Plane(P1) ∧ Plane(P2) ∧ Plane(P3)
        ∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL))
Goal(At(C1, JFK) ∧ At(C2, SFO) ∧ At(C3, SFO))
```

✧ Search Strategies

- Overview

	n of nodes expanded	n of goal tests	Time elapsed	Plan Length
Depth first graph search	624	625	3.68168	619
Breadth first search	3343	4609	15.15613	9
Uniform-cost search	4853	4855	12.46180	9
Depth-limited search	--	--	> 10 minutes	--
Breadth first tree search	--	--	> 10 minutes	--

- In Depth-first graph search, it can be seen that the Plan Length is still larger than the other two (Breadth-first search & uniform-cost search), which indicating that this search isn't optimal.
- In breadth-first search, it will expand the states at the same level and test

which one reach the goal state, suggesting that it will return an optimal plan.

- Here is the 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)
```

- Like breadth-first search, uniform-cost search keep calculating the cost for a set of paths and expand the path whose cost is the lowest. Thus, it will get an optimal plan as well.

➢ Air Cargo Problem 3

✧ Initial & Goal State

```
Init(At(C1, SFO) ∧ At(C2, JFK) ∧ At(C3, ATL) ∧ At(C4, ORD)
	∧ At(P1, SFO) ∧ At(P2, JFK)
	∧ Cargo(C1) ∧ Cargo(C2) ∧ Cargo(C3) ∧ Cargo(C4)
	∧ Plane(P1) ∧ Plane(P2)
	∧ Airport(JFK) ∧ Airport(SFO) ∧ Airport(ATL) ∧ Airport(ORD))
Goal(At(C1, JFK) ∧ At(C3, JFK) ∧ At(C2, SFO) ∧ At(C4, SFO))
```

✧ Search Strategies

- Overview

	n of nodes expanded	n of goal tests	Time elapsed	Plan Length
Depth first graph search	408	409	2.53954	392
Breadth first search	14663	18098	111.04126	12
Uniform-cost search	18235	18237	60.72423	12
Depth-limited search	--	--	> 10 minutes	--
Breadth first tree search	--	--	> 10 minutes	--

- Similar to problem 1 & 2, the depth first graph search isn't an optimal search in terms of the Plan Length in problem 3
- Regardless of the nodes expanded, Breadth-first search and Uniform-cost search are optimal in this problem because they return the neat plan.

```
Load(C1, P1, SFO)      Load(C1, P1, SFO)
Load(C2, P2, JFK)      Load(C2, P2, JFK)
Fly(P1, SFO, ATL)      Fly(P2, JFK, ORD)
Load(C3, P1, ATL)      Load(C4, P2, ORD)
Fly(P2, JFK, ORD)      Fly(P1, SFO, ATL)
Load(C4, P2, ORD)      Load(C3, P1, ATL)
Fly(P2, ORD, SFO)      Fly(P1, ATL, JFK)
Fly(P1, ATL, JFK)      Unload(C1, P1, JFK)
Unload(C4, P2, SFO)    Unload(C3, P1, JFK)
Unload(C3, P1, JFK)    Fly(P2, ORD, SFO)
Unload(C2, P2, SFO)    Unload(C2, P2, SFO)
Unload(C1, P1, JFK)    Unload(C4, P2, SFO)
```

Priya: Awesome: Good work! You have identified the optimal no. of steps for each of the 3 problems.

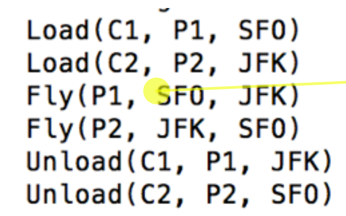
- Metrics of A* searches
  - Air Cargo Problem 1, 2 & 3

- Metrics overview of problem 1

heuristics	n of nodes expanded	n of goal tests	Time elapsed	Plan Length
h_1	55	57	0.0456	6
h_ignore_preconditions	41	43	0.0483	6
h_pg_levelsum	11	13	0.5853	6

Priya: Well done! All algorithms have been implemented properly

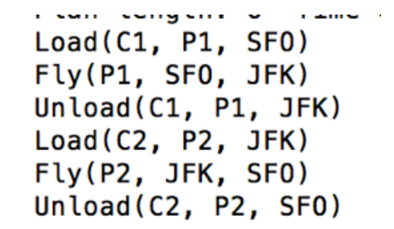
- Plan with h_1



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

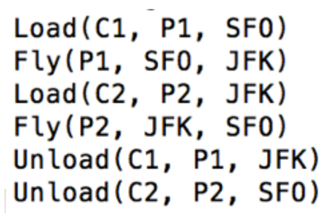
Priya: Suggestion: You don't need to list the optimal sequence of actions for all algorithms

- Plan with ignore_preconditions



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

- Plan with level sum

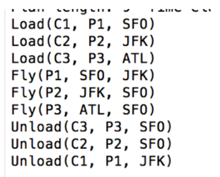


```
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C1, P1, JFK)
Unload(C2, P2, SFO)
```

- Metrics overview of problem 2

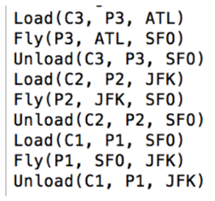
heuristics	n of nodes expanded	n of goal tests	Time elapsed	Plan Length
h_1	4853	4855	13.8454	9
h_ignore_preconditions	1450	1452	5.1872	9
h_pg_levelsum	86	88	51.4103	9

- Plan with h_1



```
Load(C1, P1, SFO)
Load(C2, P2, JFK)
Load(C3, P3, ATL)
Fly(P1, SFO, JFK)
Fly(P2, JFK, SFO)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Unload(C2, P2, SFO)
Unload(C1, P1, JFK)
```

- Plan with ignore_preconditions



```
Load(C3, P3, ATL)
Fly(P3, ATL, SFO)
Unload(C3, P3, SFO)
Load(C2, P2, JFK)
Fly(P2, JFK, SFO)
Unload(C2, P2, SFO)
Load(C1, P1, SFO)
Fly(P1, SFO, JFK)
Unload(C1, P1, JFK)
```

- Plan with level sum

    ```
    Load(C1, P1, SFO)
    Fly(P1, SFO, JFK)
    Load(C2, P2, JFK)
    Fly(P2, JFK, SFO)
    Load(C3, P3, ATL)
    Fly(P3, ATL, SFO)
    Unload(C3, P3, SFO)
    Unload(C2, P2, SFO)
    Unload(C1, P1, JFK)
    ```

- Metrics overview of problem 3

heuristics	n of nodes expanded	n of goal tests	Time elapsed	Plan Length
h_1	18235	18237	61.2400	12
h_ignore_preconditions	5040	5042	20.2001	12
h_pg_levelsum	316	318	269.8408	12

- Plan with h_1

    ```
    Load(C1, P1, SFO)
    Load(C2, P2, JFK)
    Fly(P1, SFO, ATL)
    Load(C3, P1, ATL)
    Fly(P2, JFK, ORD)
    Load(C4, P2, ORD)
    Fly(P2, ORD, SFO)
    Fly(P1, ATL, JFK)
    Unload(C4, P2, SFO)
    Unload(C3, P1, JFK)
    Unload(C2, P2, SFO)
    Unload(C1, P1, JFK)
    ```

- Plan with ignore_preconditions

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

- Plan with level sum

    ```
    Load(C2, P2, JFK)
    Fly(P2, JFK, ORD)
    Load(C4, P2, ORD)
    Fly(P2, ORD, SFO)
    Load(C1, P1, SFO)
    Fly(P1, SFO, ATL)
    Load(C3, P1, ATL)
    Fly(P1, ATL, JFK)
    Unload(C4, P2, SFO)
    Unload(C3, P1, JFK)
    Unload(C2, P2, SFO)
    Unload(C1, P1, JFK)
    ```

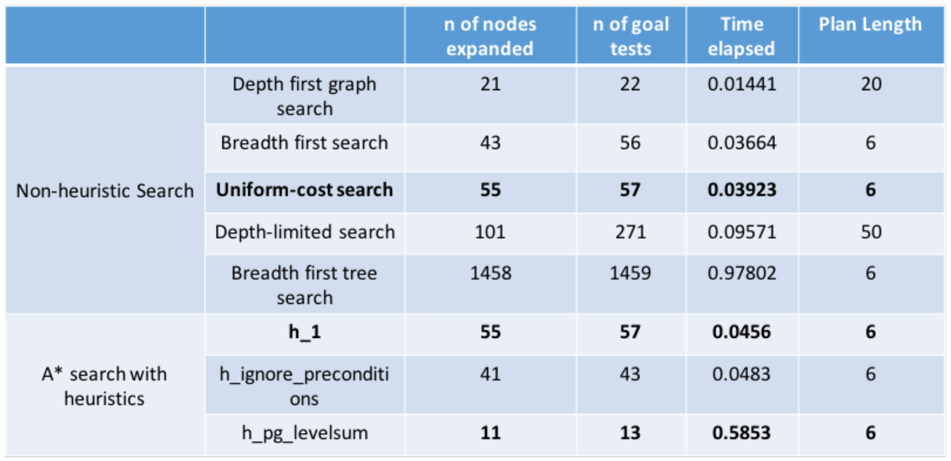
- A* Searches with heuristics
    - From these 3 overview, the A* search with level sum heuristic expanded the least nodes to get the optimal plan regardless of the time it spent, which means level

sum heuristic is better than the other two in optimality.

- In terms of the elapsed time, the A* search with level sum heuristic need to calculate the cost through the plan graph, indicating that the heuristic is complicated than the other two which leads to more time on finding the solution. The A* search with h_1 is simply like the uniform cost search, which consider every node has the same cost, and the A* search with ignored_preconditions just calculates the number of goal literals in the node's state. These simple heuristic made them spending less time on the search.

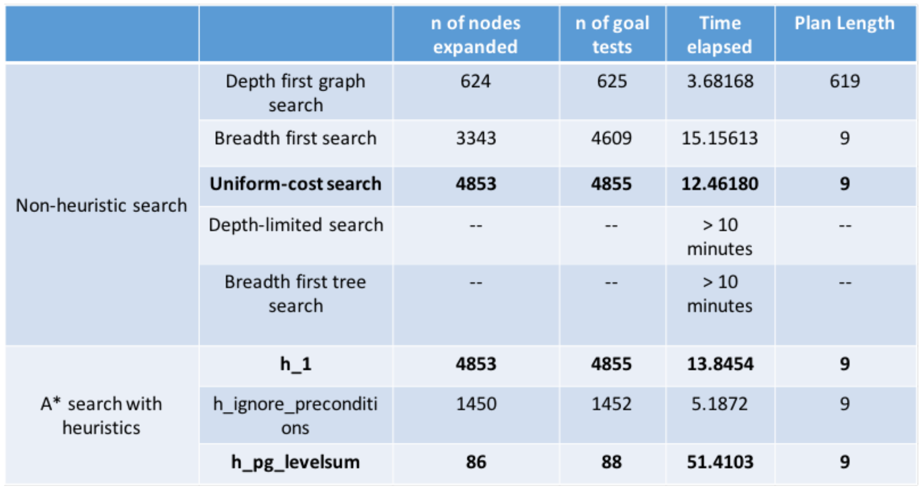
Priya: Awesome: This is a great point. Better heuristics like level sum may be more efficient in reducing the no. of expansions but costly in terms of taking more time.

- Overall Analysis
  - Overview of the metrics for problem 1



		n of nodes expanded	n of goal tests	Time elapsed	Plan Length
Non-heuristic Search	Depth first graph search	21	22	0.01441	20
	Breadth first search	43	56	0.03664	6
	Uniform-cost search	**55**	**57**	**0.03923**	**6**
	Depth-limited search	101	271	0.09571	50
	Breadth first tree search	1458	1459	0.97802	6
A* search with heuristics	**h_1**	**55**	**57**	**0.0456**	**6**
	h_ignore_preconditions	41	43	0.0483	6
	h_pg_levelsum	**11**	**13**	**0.5853**	**6**

  - Overview of the metrics for problem 2



		n of nodes expanded	n of goal tests	Time elapsed	Plan Length
Non-heuristic search	Depth first graph search	624	625	3.68168	619
	Breadth first search	3343	4609	15.15613	9
	Uniform-cost search	**4853**	**4855**	**12.46180**	**9**
	Depth-limited search	--	--	> 10 minutes	--
	Breadth first tree search	--	--	> 10 minutes	--
A* search with heuristics	**h_1**	**4853**	**4855**	**13.8454**	**9**
	h_ignore_preconditions	1450	1452	5.1872	9
	h_pg_levelsum	**86**	**88**	**51.4103**	**9**

  - Overview of the metrics for problem 3

		n of nodes expanded	n of goal tests	Time elapsed	Plan Length
Non-heuristic search	Depth first graph search	408	409	2.53954	392
	Breadth first search	14663	18098	111.04126	12
	Uniform-cost search	**18235**	**18237**	**60.72423**	**12**
	Depth-limited search	--	--	> 10 minutes	--
	Breadth first tree search	--	--	> 10 minutes	--
A* search with heuristics	**h_1**	**18235**	**18237**	**61.2400**	**12**
	h_ignore_preconditions	5040	5042	20.2001	12
	h_pg_levelsum	**316**	**318**	**269.8408**	**12**

- Analysis
  - Comparing to A* search with ignore_precondition heuristic, A* search with level sum heuristic expanded the less node to get the similarly optimal plan. Although it spent much time on the search, it could say that the optimality of A* search with level sum is better and the space of A* search with level sum is lesser than the A* search with ignore_preconditions
  - Comparing to uninformed searches, especially Breadth-first search and uniform-cost search, A* with ignored_precondition heuristic performs better enough to get the optimal plan with less time and expanded nodes as the problem gets complicated. It achieves that by considering the minimum number of actions that must be carried out from the current state in order to satisfy all the goal literals. That is, with A* search, it would choose the expand the node whose state literals matched most of the goal literals, resulting in lesser nodes to expand to find an optimal solution than non-heuristic search.
  - On the other hand, if there's some space issue, the A* search with level sum heuristic might be better than A* search with ignored_preconditions because of the lesser nodes it expanded during search. During the A* search with level sum heuristic, it will create a plan graph for every current state, and calculate the level sum for the current state. From this heuristic, we could know that it also expands the node which is closer to achieve the goal like heuristic ignored_preconditions. From the results above, it could be said the heuristic with level sum is a better estimate of the cost for A*search or for this problem.

Priya: I agree with your conclusion