Implementing a Planning Search - Heuristic Analysis

Optimal plan examples for the three planning problems in the Air Cargo domain

Problem 1 - the optimal plan has 6 actions, as in the following example:

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

Problem 2 - the optimal plan has 9 actions, as in the following example:

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

Problem 3 - the optimal plan has 12 actions, as in the following example:

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

Non-heuristic search result metrics

In the following metrics obtained for Problem 1, we see that breadth-first search, uniform cost search and greedy best first graph search found an optimal solution.

Method	Plan length	Time (seconds)	Expansions
breadth_first_search	**6**	0.0227	43
depth_first_graph_search	12	0.0056	12
uniform_cost_search	**6**	0.0248	55
greedy_best_first_graph_search	**6**	0.0046	7

In the following metrics obtained for Problem 2, we see that breadth-first search and uniform cost search have found an optimal solution.

Method	Plan length	Time (seconds)	Expansions
breadth_first_search	**9**	11.6552	3343
depth_first_graph_search	575	2.8167	582
uniform_cost_search	**9**	9.1298	4852
greedy_best_first_graph_search	17	1.8443	990

In the following metrics obtained for Problem 3, we see that breadth-first search and uniform cost search have found an optimal solution.

Method	Plan length	Time (seconds)	Expansions
breadth_first_search	**12**	91.5264	14663
depth_first_graph_search	596	2.9209	627
uniform_cost_search	**12**	41.2349	18235
greedy_best_first_graph_search	22	12.7151	5614

Observations

1) Optimality

All of the non-heuristics search techniques found solutions to the problems.

Depth first search does not generally find an optimal solution. It returns the first solution that it encounters. Because the algorithm checks each possible solution successively, reaching the deepest nodes first, it has no knowledge of the optimal solution.

Breadth first search finds the optimal solution, even though it also returns the first encountered solution. However, it checks solutions in parallel, as it goes through all the children of a node at each level, stopping as soon as a node achieves the goal state. This is why the first solution is also optimal.

Uniform cost search also consistently finds the optimal solution, as it checks each child of a node as long as the total path cost up until that child node is the smallest overall cost at that step.

Greedy best first graph search does not find the optimal solution all the time. However, if finds an acceptable solution - generally better than depth first search.

2) Execution time

Depth first search is usually the fastest way to find a solution. Greedy best first search is also fast and returns a better solution than depth first. Uniform cost search performed faster than breadth first in this example. Nonetheless, in other problems it might have a higher execution time than breadth first, because it’s exploration is not a uniform search through a node’s children - it follows the smallest cost path, expanding nodes in any direction that respects this rule.

3) Memory usage (number of node expansions)

Uniform cost search uses the most memory, as it has to keep more node expansions in memory. Depth first search is the most memory efficient.

Heuristic search result metrics

In the following metrics obtained for Problem 1, we see that A* search with both heuristics have found an optimal solution.

Method	Plan length	Time (seconds)	Expansions
astar_search h_ignore_preconditions	**6**	0.0222	41
astar_search h_pg_levelsum	**6**	0.6362	11

In the following metrics obtained for Problem 2, we see that A* search with both heuristics have found an optimal solution.

Method	Plan length	Time (seconds)	Expansions
astar_search h_ignore_preconditions	**9**	2.9429	1450
astar_search h_pg_levelsum	**9**	54.6435	86

In the following metrics obtained for Problem 3, we see that A* search with both heuristics have found an optimal solution.

Method	Plan length	Time (seconds)	Expansions
astar_search h_ignore_preconditions	**12**	11.9753	5040
astar_search h_pg_levelsum	**12**	276.5454	318

Observations

1) Optimality

A* search with both heuristics have found an optimal solution.

2) Execution time

The A* search with the ignore preconditions heuristic is fastest than A* with levelsum. The ignore preconditions heuristic offers a fast way to compute an estimation of the minimum number of actions needed to reach the goal from the current state. The levelsum heuristic is more time consuming, as it calculates the cost of the current level by adding the level costs of each goal.

3) Memory usage (number of node expansions)

The ignore preconditions heuristic makes use of more memory, as it has to store more node expansions.

Conclusions

Both heuristics are optimal. If your focus is speed, the ignore preconditions is best. If memory is an issue, you might consider using the levelsum heuristic.

A* search with the ignore preconditions heuristic was the fastest method to return an optimal solution. It also uses less memory than all the other non-heuristic searches.

Depth first search remains the fastest method to return a non-optimal solution.