Heuristic_Analysis for Planning Project

Comparison of Non-heuristic & Heuristic Search Solution

1. `air_cargo_p1`

   Optimal solution found:

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

   Comparison for p1:

Search Name	# of Node Expanded	# of Goal Tests	# New Nodes	Time Elapsed(s)	Plan Length	Optimal?
Breadth_first_search	43	56	180	0.13	6	Y
Breadth_first_tree_search	1458	1459	5960	4.2	6	Y
Depth_first_graph_search	21	22	84	0.06	20	N
Uniform_Cost_search	55	57	224	0.16	6	Y
Depth_Limited_Search	101	271	414	0.3	50	N
Recursive_best_first_graph_search	4229	4230	17023	12.6	6	Y
Greedy_best_first_graph_search	7	9	28	0.02	6	Y
A*_search_h1	55	57	224	0.178	6	Y
A*_search_h_ignore_preconditions	41	43	170	0.14	6	Y
A*_search_h_pg_levelsum	11	13	50	0.57	6	Y

2. `air_cargo_p2`

   Optimal solution found:

```
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)
```

`Breadth_first_tree_search` and `Recursive_best_first_-graph_search` take more than 15 minutes to finish, so they are dropped from the analysis

Comparison for p2:

Search Name	# of Node Expanded	# of Goal Tests	# New Nodes	Time Elapsed(s)	Plan Length	Optimal?
Breadth_first_search	3343	4609	30509	43	9	Y
Breadth_first_tree_search						
Depth_first_graph_search	624	625	5602	8.84	619	N
Uniform_Cost_search	4852	4854	44030	53.8	9	Y
Depth_Limited_Search	222719	2053741	2054119	2678	50	N
Recursive_best_first_graph_search						
Greedy_best_first_graph_search	990	992	8910	10.5	15	N
A*_search_h1	4852	4854	44030	52.8	9	Y
A*_search_h_ignore_preconditions	1450	1452	13303	17	9	Y
A*_search_h_pg_levelsum	86	88	841	50	9	Y

3. `air_cargo_p3`

Optimal solution found:

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

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

Comparison for p3:

Search Name	# of Node Expanded	# of Goal Tests	# New Nodes	Time Elapsed(s)	Plan Length	Optimal ?
Breadth_first_search	14663	18098	129631	280	12	Y
Breadth_first_tree_search						
Depth_first_graph_search	408	409	3364	6.4	392	N
Uniform_Cost_search	18235	18237	159716	250	12	Y
Depth_Limited_Search						
Recursive_best_first_graph_search						
Greedy_best_first_graph_search	5614	5616	49429	75	22	N
A*_search_h1	18235	18237	159716	246	12	Y
A*_search_h_ignore_preconditions	5040	5042	44944	70	12	Y
A*_search_h_pg_levelsum	325	327	3002	242	12	Y

Analysis for Uninformed Search

Of the uninformed search, BFS, BFS_tree and Uniform_Cost_Search all return optimal solutions. DFS doesn't guarantee an optimal solution and its major advantage is a save in space: requiring only O(b*m) space as opposed to BFS's exponential space requirement. However, in these particular cargo problems, the space required isn't much so DFS isn't ideal here.

While BFS_tree found the optimal solution for p1, it was too slow and timed out for p2 and p3. This is in agreement with Peter Norvig and Russel Stuwart's comment that in tree search "redundant paths are unavoidable" and "algorithms that forget their

history are doomed to repeat it". To correct his, they suggest to augment the tree-search algorithm with an explored set that keeps track of expanded node (Norvig and Russel, 2009). This is how the graph search version, BFS, is implemented and the test result also proves that BFS is superior to the BFS_tree since BFS found optimal solutions for all problems without ever timing out.

As for Depth_Limited_Search, it is non-optimal and it timed out for p2 because of the large depth limit and space requirement: O(b^l), where b is the branching factor and l is the depth limit (Norvig and Russel, 2009).

Over all the uninformed searches, BFS and uniform search play out the best. They are both optimal and have similar metrics in terms of # of nodes expanded and # of goal tests. To be precise, uniform cost tends to be slightly faster than BFS in more complex problems (12% faster in p3) even though it expanded more nodes.

Analysis for Heuristic Search

All three versions of A* (h1, h_ignore_precondition, h_pg_levelsum) are optimal. The h1 version expands the most number of nodes and is the slowest. The ignore_precondition version is the fastest even though it doesn't expand the fewest number of nodes. The level_sum version expands the least amount of nodes, an order of magnitude smaller than the 2nd best: ignore_precondition. Still, it is not the fastest due to the need of constructing plan graph and evaluating level sum.

`Greedy_best_first_graph_search` is also studied and it is among the fastest heuristic search methods. However, it's not optimal for p2 and p3. `Recursive_best_first_graph_search` is optimal for p1 but it timed out for p2 and p3.

Among the heuristic search, `A*_h_ignore_precondition` is preferred when it comes to execution time and optimality.

Comparison between Non-heuristic Search and Heuristic Search

Judging by number of nodes expanded and optimality, `A*_search_h_pg_level_sum` is the best and it also performs better than optimal non-heuristic searches: faster & smaller space requirement.

Judging by execution time and optimality, `A*_search_h_ignore_precondition` is the best and it also performs better than optimal non-heuristic searches: faster & smaller space requirement. Intuitively, its use of heuristics guides the search and hence the search is more focused than uninformed search such as DFS or BFS. Meanwhile, by ignoring precondition, the task is relaxed and hence the heuristic can be computed quickly and is admissible, which guarantees optimality.

References

Peter Norvig and Russell Stuwart (2009). Aritificial Intelligence: A Modern Approach.