Heuristic Analysis

Nhat Bui - AIND Project 3

Sypnosis

This report compares and contrasts three uninformed search algorithms: Breadth First Search (BFS), Depth First Search (DFS), Uniform Cost Search (UCS) and two informed search algorithms: A* search with ignore preconditions heuristic and A* search with level sum heuristic to solve a planning problem defined in PDDL (Planning Domain Definition Language). The report goes over each algorithm's optimality and performance based on empirical results and finally offers recommendations on the choice of algorithms best suited for the given problems.

Table of contents

Optimal plans

Air Cargo Problem 1:
Optimal plan length: 6
Possible optimal Plan:

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

Air Cargo Problem 2:
Optimal plan length: 9
Possible optimal 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)
```

Air Cargo Problem 3:
Optimal plan length: 12
Possible optimal Plan:

```
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)
```

Performance comparisons

Air cargo problem 1:

	Optimality	Plan Length	Expansions	Goal Tests	New Nodes	Time
BFS	Yes	6	43	56	180	0.0314
DFS	No	20	21	22	84	0.0158
UCS	Yes	6	55	57	224	0.0392
h_ip	Yes	6	51	53	208	0.0442
h_gp	Yes	6	11	13	50	0.6135

Air cargo problem 2:

	Optimality	Plan Length	Expansions	Goal Tests	New Nodes	Time
BFS	Yes	9	3343	4609	30509	11.9704
DFS	No	619	624	625	5602	2.9651
UCS	Yes	9	4852	4854	44030	10.3349
h_ip	Yes	9	2904	2906	26544	7.4019
h_gp	Yes	9	86	88	841	57.2542

Air cargo problem 3:

	Optimality	Plan Length	Expansions	Goal Tests	New Nodes	Time
BFS	Yes	12	14663	18098	129631	88.5780
DFS	No	392	408	409	3364	1.5510
UCS	Yes	12	18235	18237	159716	45.4150
h_ip	Yes	12	9088	9090	81079	26.7696
h_gp	Yes	12	325	327	3002	297.2497

Uninformed algorithms

Optimality-wise, all uninformed algorithms except depth first search produce optimal plans of length 6, 9 and 12. This is due to the guarantees built into DFS and UCS. DFS found a valid plan for each problem which satisfies the goal but the plans include many redundant steps as the algorithm keeps trying more actions instead of backtracking to a shorter but not yet explored path as with BFS or UCS.

In terms of time elapsed, for problem 1, the difference is negligible thanks to the small problem size. All tested algorithms result in similar run-time of less than 1 second, only slightly noticeable to a human observer. As the problem size increases, search times begin to diverge. Out of three uninformed heuristic, BFS and UCS perform at a similar 11.97 seconds and 10.34 seconds respectively for problem 2, 88.58 seconds and 45.42 seconds for problem 3 (the latter difference comes from optimization details of the data structure used to hold the frontier. It is not inherent to the algorithm itself). DFS finishes the fastest taking only 2.97 seconds for problem 2 and 1.55 seconds for problem 3. The fact that DFS can quickly stumble upon a valid solution demonstrates a high density of possible (optimal and nonoptimal) valid paths in the search space of our air cargo problems.

In terms of node expansions, there is a clear positive correlation between time elapsed and the number of expansions. BFS and UCS need to search through a similar number of expansions to find an optimal solution. But since the search beam is not focused i.e it is not guided by an heuristic function, the number of expansions required quickly grows out of hand as the problem size increases. In fact, BFS and UCS top the chart in terms of expansions for both problem 2 and problem 3 across all 5 algorithms tested including informed A* searches. On the other hand, DFS expanded far fewer nodes than BFS and UCS to reach a solution. However, the solution is far from optimal (392 steps from the optimal 12 steps for problem 3) suggesting a trade-off between optimality and performance in this case can be unacceptably expensive.

Informed algorithms

Ignore preconditions as a relaxed constraint heuristic is always admissible and thus, A* search using ignore preconditions will always return an optimal path. In contrast, level sum requires subgoals independence or only negative interaction between goals to be admissible. These requirements don’t hold in our problem setting. Consider that our goals only include goals where a certain cargo have to be at a certain airport and our initial state always have cargos at the airports and not in any plane, it follows that to satisfy each individual goal, it takes 0 action if the cargo is already at that airport, 3 actions (Load, Fly, Unload) if there’s a plane at the port of departure, 4 (Fly, Load, Fly, Unload) if there’s no plane there. However merged plans can share Fly actions if they share ports of departure and destinations, resulting in the optimal plan being shorter than the sum of the paths that satisfy individual goals, making level sum a pessimistic estimate. Thus, for our problem setting, level sum is not admissible and we lose the guarantee of an optimal solution. However, in practice, level sum does return optimal solutions for all three problems. This suggests that graphplan based level sum can be used with good results despite being inadmissible.

Performance wise, A* with ignore preconditions heuristic cuts down the number of required expansions significantly compared to other optimal searches (BFS and UCS) with increasing savings as problem size increases. (13% fewer expansions than BFS for problem 2 and 38% for problem 3). Search time similarly improves upon BFS and UCS with more visible discrepancy as problem size grows (38% faster than DFS for problem 2 and 70% for problem 3). A* with planning graph level sum heuristic really shines in the number of expansions it can do away with, requiring only 2.6% of what BFS need for problem 2, and 2.2% for problem 3, indicating a very high accuracy in the estimation of the optimal path. However, in exchange for the substantially reduced expansions, level sum suffers heavy penalty from the expensive building of the planning graph, finishing last among all algorithms across all 3 problems. The difference in runtime is so large compared to the more efficient ignore preconditions (7.7 times more time than ignore preconditions for problem 2 and 11.1 times for problem 3), it is impractical to use planning graph unless the problem size is so large other methods become infeasible.

Recommendations

Taking into account all relevant metrics, A* search with ignore preconditions heuristic return optimal results while taking the shortest amount of time except for problem 1 where DFS and UCS takes similar amount of time to finish. Thus, it can be seen that except for very small problem size, generally A* search with a good heuristic should be prioritized over other methods. In our problem setting, ignore preconditions is the better heuristic to choose over graphplan based level sum, as the benefits of reducing expansions has not been shown to outweigh the additional cost of building the planning graph.

Additional discussion

An intriguing result is why DFS plan length does not equal the number of expansions, given one’s impression that DFS should never branch for there is no state in our problem that no action is possible i.e expanding the frontier should always add new nodes. This is not quite true for this implementation of DFS which is depth first graph search specifically, where existing nodes in the frontier and the explored set will prevent identical nodes from being added. Thus if there’s a node whose all possible successors are already in the frontier or the explored set, no new node will be added. Search will continue with the last node added to the frontier, which belongs to the same or shallower depth as the node that has just been popped from the stack, making the plan length differ from the number of expansions by one each time this happens. What’s more surprising is that this is the reason DFS can find a valid plan at all. If the same problem is subjected to the tree version of DFS, no solution will be found as the search keeps taking the first action available to it, which due to the order in which the list of all possible actions is constructed, will always result in a long list of Fly actions to try first before any other type of action. Shuffling the actions mitigates the problem, and both versions of DFS can find a solution but take wildly fluctuating amounts of time.