N. Veiga (nohe.veiga@gmail.com)
Udacity's Artificial Intelligence Nanodegree
Sep 2017

Heuristics analysis report

Measurement of the optimal

In this Cargo problems, we assume that no action can satisfy more than one goal, and since goals are satisfied by different actions, so if there are three goal states, there will be necessary at least three actions to satisfy the goal state. It is also assumed that all step nodes have the same cost.

All these search algorithms will return a sequence of actions to complete the goal, not always an optimal one. From the AIMA book[1] it is known that in order to evaluate performance these characteristics should be considered:

- **Completeness:** Is the algorithm guaranteed to find a solution when there is one?
- **Optimality:** Does the strategy find the optimal solution?
- **Time complexity:** How long does it take to find a solution?
- **Space complexity:** How much memory is needed to perform the search? The more search space, the more space complexity, therefore more memory to perform the operation.

Our optimal is maybe the less number of operations to complete the goal, therefore 'plan length' parameter, but as not all the search functions used in this project are optimal, it has to be considered the nature itself of the algorithm. Node expansions and time elapsed gives an idea of the amount of resources consumed.

Comparison of search algorithms overview

Uninformed search

Among Breadth-first and Breadth tree algorithms, the main difference is an explore set implementation to prevent expanding already explored paths. Breadth-first tree search algorithm is able to find an optimal solution with more expansions, which is extremely inefficient, as probably some of these additional expansions are loopy paths. However, these both algorithms are optimal since they expand the shallowest nodes, therefore they both guarantee that no deeper nodes in the search space are going to be considered.

Both Depth-first and Depth-limited algorithms are aimed to find the longest path in a non-optimal way. The main difference is the branching factor due to the imposition of

[1] AIMA book p80 3.3.2 Measuring problem-solving performance

exploring the search space 50 times, producing several redundant paths on the search tree. Depth algorithms are not optimal as they might overlook the goal. Depth-limited has some advantages over Depth-first, since forcing the algorithm to search stop at some level, completeness can be imposed preventing those cases where the branching factor is infinite. However, for these problems complexity, the depth-limited search algorithm has a worse performance as more redundancy is being added.

Regarding the Uniform Cost search, it expands nodes according to their optimal path cost. Compared to Breadth-first search, uniform-cost search will keep expanding nodes, trying to evaluate if there is any node goal-satisfying with lower cost meanwhile the breadth-first search algorithm will stop early once it finds a goal state.

Informed (Heuristic search)

With the heuristics algorithms, we take advantage of knowing where the can we find a goal state. An admissible heuristics is one that never overestimates the cost to reach the goal, however, all the nodes have the same cost in these problems. That makes that the apparently informed searches will act as uninformed as the advantage of knowing the possible goal state is not being used. However, strictly speaking, Recursive Best First search, Greedy Best First Graph search and A* informed searches.

The greedy algorithm is going to try to get as close as it can to the goal state, overlooking other paths with lesser cost. A* is also a path cost-based algorithm, likewise the uninformed-cost search algorithm, it estimates the cost of reaching a node, but combined to the cost from the node to the goal. In these three cargo problems, A* is used with different planning heuristics:

- A* + h_1 which means no heuristics and becomes A* to an uninformed search returning the same results as an Uninformed-cost search algorithm.

- A* + Ignore-preconditions which drops all preconditions from actions and thus, every action is applicable in every state and the goal can be achieved in only one step. With this heuristics, we get the minimum number of actions that reach the goal state.

- A* + level-sum is a planning graph heuristics that treat the problem as independent subgoals, returning the sum of the level costs of the goals

Recursive Best First is a memory-bounded search algorithm, similar to recursive depth-first search, but rather than continuing indefinitely down the current path, it uses a limit variable. If the current node exceeds this limit, the recursion unwinds back to the alternative path. This algorithm can provide a solution, in this case, using a memoization technique, in those cases when the search space is so complex, that A* runs out of memory.

N. Veiga (nohe.veiga@gmail.com)
Udacity's Artificial Intelligence Nanodegree
Sep 2017

Observations and conclusions

In general, considering the search space for the either of the cargo problems and taking into account that all the paths have the same search cost, it seems reasonable to choose the breadth-first search, as it performs a reasonable number of actions needed with the minimum depth in the search space. It must be noticed that depth-first graph search is able to find a goal state on the way to find the deepest node with fewer node expansions and thus, expending less time. However, it cannot be guaranteed that always would work this way independently of the problem. Certainly, the metrics thrown by depth-first are fair enough for all these problems, though *-please see the next section 'Data' in order to explore metrics-.*

Uniform-cost algorithm for P2 and P3, has the better performance despite the number of nodes expanded, due to the implementation of the PriorityQueue, is more expensive to use, than the FIFO Queue[2]. Anyway, as the all the node steps have the same cost, in this case, it could be seen the Breadth-first search as the cheapest algorithm from the uninformed list.

Regarding the 'heuristics' algorithms Recursive Best First Search, Greedy Best First Graph Search with h_1 and A* search with h_1, they are formally informed algorithms, but in this case, since this heuristics returns the same value for all nodes, they act like uniformed algorithms.

Level-sum heuristics is only admissible if the problem is subgoal independent, so it can be considered for P1 and P2. However, for P3 as we have two planes and four cargos, the subgoals are not independent, therefore an optimal solution cannot be guaranteed by this heuristic. The best metrics are provided by the A* search with h_ignore preconditions, since the problem is relaxed, completely ignoring all preconditions whereas level-sum heuristics, incurs in more loops.

Regarding which algorithm to use, it depends on the problem. Some non-optimal algorithms have a fairly good metrics due to the behavior of the data structures used in the implementation of this problem, producing some 'lucky' optimizations. On the other hand turning back to the origin of this project, 'Build a domain-independent planner', the proposed algorithm should be able to solve the problem considering the logic nature of the same, therefore, BFS algorithm seems to be a good option for these problems as we are not discussing the cost of every node path. The best metrics are provided by A* with ignoring preconditions heuristics since the problem becomes relaxed and it can reach a goal state with fewer caveats.

---

[2] https://discussions.udacity.com/t/time-elapsed-bfs-vs-ucs/311680/7, https://discussions.udacity.com/t/uniform-cost-search-faster-than-breadth-first/324045/3

N. Veiga (nohe.veiga@gmail.com)
Udacity's Artificial Intelligence Nanodegree
Sep 2017

Data

	search algorithm	type of algorithm	optimal ?	vers	P1			P2			P3		
					exp	pl	te	exp	pl	te	exp	pl	te
1	breadth first search	uninformed	yes	graph	43	6	0.066	3343	9	28.163	14663	12	188.221
2	breadth first tree search	uninformed	yes	tree	1458	6	1.944	--	--	--	--	--	--
3	depth first graph search	uninformed	no	graph	12	12	0.022	582	575	6.120	627	596	6.232
4	depth limited search	uninformed	no	tree	101	50	0.196	--	--	--	--	--	--
5	uniform cost search	uninformed	yes	graph	55	6	0.091	4852	9	25.419	18235	12	143.849
6	recursive best first search	theoretically informed, formally uninformed	yes	tree	4229	6	5.841	--	--	--	--	--	--
7	greedy best first graph search with h_1	theoretically informed, formally uninformed	no	graph	7	6	0.012	990	17	5.399	5614	22	31.632
8	astar search with h_1	theoretically informed, formally uninformed	yes	graph	55	6	0.084	4852	9	26.580	18235	12	111.969
9	astar search with h ignore preconditions	informed (heuristics)	yes	graph	41	6	0.086	1450	9	8.691	5040	12	37.818
10	astar search h pg levelsum	informed (heuristics)	yes-but	graph	55	6	4.583	4852	9	4180.141	--	--	--

exp: node expansions
pl: plan length
te: time elapsed
--: time > 10 min

Optimal plans

Air Cargo Problem 1
Example of optimal sequence of operations proposed by breadth-first search:

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

Air Cargo Problem 2
Example of optimal sequence of operations proposed by breadth-first search algorithm.

Load(C2, P2, JFK)
Load(C1, P1, SFO)
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
Example of optimal sequence of operations proposed by breadth-first search algorithm.

Load(C2, P2, JFK)
Load(C1, P1, SFO)
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)