Artificial Intelligence Nanodegree

Planning

Overview

The goal of this project is to create a deterministic planning and scheduling search agent for an Air Cargo transport system. Multiple algorithms are implemented to find an optimal plan for a set of planning problems.

Optimal Solutions

Each problem is solved using a diversity of deterministic greedy and non-greedy algorithms:

- uninformed search
  - breadth first search (BFS)
  - depth first search (DSF)
  - uniform cost search (UCS)
- heuristic search with unmet goals, level sum, max level and set level
  - greedy best first graph search (GBFG)
  - A*

Analyze the search complexity

On the uninformed search we can draw some observations on the different approaches. In theory the three approaches have the same complexity in the limit, a carefully crafted problem can force all three algorithm to exhaust all branches of the tree before getting to a solution.

BFS prioritizes the branch with the smallest depth, i.e. it will grow all branches by one node at a time. BFS is optimal in finding the shortest path to a solution, it does not guarantee optimal solution in terms of cost.

DFS prioritizes the branch with the highest depth, i.e. it will explore one branch down to its leaf before exploring another branch. DFS is neither optimal in terms of cost nor in terms of finding the shortest path.

UCS (aka Dijkstra) prioritizes branches with the minimum cumulative cost. UCS is optimal in finding

the lowest cost solution and given a cost of 1 on each vertex will behave similarly to BFS and find the shortest path.

Search complexity in expansions

The table below presents the number of expansions for each algorithm used against the four problems, where problem 1 has the smallest action space and problem 4 the largest.

		Number of Actions (problem 1,2,3,4)			
		20	72	88	104
Expansions	BFS	43	3343	14663	99736
	DFS	21	624	408	25174
	UCS	60	5154	18510	113339
	GBFG – unmet	7	17	25	29
	GBFG – levelsum	6	9	14	17
	GBFG – maxlevel	6	27		
	GBFG – setlevel	6	9		
	Astar – unmet	50	2467	7388	34330
	Astar – levelsum	28	357	369	1208
	Astar – maxlevel	43	2887		
	Astar – setlevel	33	1037		

Unsurprisingly BFS creates a larger set of expansions than DFS. BFS will exhaustively explore the search tree level by level before reaching a leaf (i.e. a solution). DFS on the other hand will explore one branch down to its leaf and as such is able to early stop if a branch offers a solution. Because UCS is not looking for the shortest path but rather the optimal cost solution, it is not surprising to see that it has the largest number of expansions of the uninformed search.

The heuristic based algorithm display a striking difference between the greedy BFG and A* which is non-greedy. BFG not unlike DFS focus on finding a solution, whereas A* (subject to its heuristic) is focused on finding an optimal solution.

Search complexity in time

The table below presents the time taken by each algorithm against the four problems

		Number of Actions (problem 1,2,3,4)			
		20	72	88	104
Time in sec	BFS	0.0023	0.8138	3.9530	37.6600
	DFS	0.0012	1.1452	0.4530	1,407.1460
	UCS	0.0038	1.4554	6.0322	50.9945
	GBFG – unmet	0.0006	0.0075	0.0136	0.0254
	GBFG – levelsum	0.2308	5.8090	12.6010	22.9503
	GBFG – maxlevel	0.1721	11.2325		
	GBFG – setlevel	0.6879	14.1626		
	Astar – unmet	0.0040	0.9033	3.5330	22.0996
	Astar – levelsum	0.5921	151.9912	227.9900	1,299.4800
	Astar – maxlevel	0.6178	844.1260		
	Astar – setlevel	1.6815	1,104.0552		

The time complexity here is a little biased by the implementation of the different heuristics (i.e. maxelevel, setlevel etc..). We can however draw some observations.

Irregardless of their nature, uniformed or heuristic based, the algorithms time complexity is somewhat exponentially correlated to size of the action domain. There is an exception however, GBFG with unmet goal heuristic seem to rather be correlated in a logarithmic scale.

Search results

The table below presents the resulting plan length of each algorithm against the four problems.

	Number of Actions (problem 1,2,3,4)			
	20	72	88	104
Plan Length BFS	6	9	12	14
DFS	20	619	392	24132
UCS	6	9	12	14
GBFG – unmet	6	9	15	18
GBFG – levelsum	6	9	14	17
GBFG – maxlevel	6	9		
GBFG – setlevel	6	9		
Astar – unmet	6	9	12	14
Astar – levelsum	6	9	12	15
Astar – maxlevel	6	9		
Astar – setlevel	6	9		

As mentioned prior BFS and UCS are respectively optimal in terms of length and cost, DFS on the other hand is not optimal and this is clearly shown here.

The heuristic based algorithm are rather consistent in the solutions provided, but it now becomes more clear that greedy BFG provides sub-optimal solutions despite providing a rather balanced optimal efficiency and performance ratio. A* see however to be providing optimal solutions, subject to its heuristic.

Questions

Which algorithm or algorithms would be most appropriate for planning in a very restricted domain (i.e., one that has only a few actions) and needs to operate in real time?

> In a restricted domain GBFG with the unmet goal policy is the fastest of the heuristic search but does not provide the optimal solution unlike A* with unmet goal policy.
> Among the uninformed algorithm BFS and UCS albeit slower than DFS may be more convenient due to the optimal solution guarantee.

Which algorithm or algorithms would be most appropriate for planning in very large domains (e.g., planning delivery routes for all UPS drivers in the U.S. on a given day)

> For a large domain GBGF with the unmet goal policy is once again the fastest, but in this scenario time is not as much of a constraint as optimality of the solution. A* with the same

policy is considerably slower but offers a good compromise on optimal efficiency compared to uninformed searches.

Which algorithm or algorithms would be most appropriate for planning problems where it is important to find only optimal plans?

According to some studies[1], A* is an optimally efficient algorithm given an admissible and consistent heuristic, if the heuristic is not consistent in that case Dijkstra/UCS will outperform A*.