CSE643: Artificial Intelligence

Assignment-2

Divyajeet Singh (2021529)

October 31, 2023

Theory

Problem-1

Part (a)

Yes, the A* algorithm can still find the optimal solution without an explored set provided that the heuristic function is **admissible** (and there are no negative weight cycles). An admissible heuristic helps in exploration of the nodes in a non-decreasing order of their costs. The purpose of the explored set is to avoid exploring already visited nodes. So without it, the algorithm will explore the same node multiple times, but it will still find the optimal solution - as the admissible heuristic will ensure that the goal node is reached with the optimal path.

Part (b)

Yes, if the heuristic is admissible and the state space is finite, then the A* algorithm will still be complete without an explored set. If the graph is finite, the algorithm will eventually end up exploring all nodes (since $f(n) = g(n) + h(n)$ will increase in the cycles), making the algorithm complete. Note that if the state space has infinite nodes or contains negative weight cycles, then the algorithm may not be complete as negative weight cycles invite infinite exploration in the cycle.

Part (c)

No, the A* algorithm will not be any faster without an explored set. The explored set prevents exploration of already visited nodes, and hence reduces the number of nodes that need to be explored. Without it, the algorithm will explore and expand the same node multiple times. Thsi redundancy can easily make the algorithm slower, reducing its efficiency, and slowing it down - i.e. increasing the running time.

Problem-2

A* Search

A* search is an informed searching technique that expands the node with the minimum value of evaluation function $f(n) = g(n) + h(n)$, where $g(n)$ is the path cost from the source node to the current node n, and $h(n)$ is the heuristic value of the nodes. The following table shows the value of $f(n)$ for each node after each iteration.

Iteration	Node	A	B	C	D	E	F	G	H	I	J	K	Frontier
0	-	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$\{S\}$
1	S	12	24	13	∞	∞	∞	∞	∞	∞	∞	∞	$\{A, B, C\}$
2	A	12	18	13	16	∞	∞	∞	∞	∞	∞	∞	$\{B, D, C\}$
3	C	12	18	13	16	34	15	∞	∞	∞	∞	∞	$\{B, D, F, E\}$
4	F	12	18	13	16	34	15	26	∞	∞	∞	∞	$\{B, D, E, G\}$
5	D	12	18	13	16	34	12	26	19	40	∞	∞	$\{B, E, F, G, I, H\}$
6	F	12	18	13	16	34	12	23	19	40	∞	∞	$\{B, E, G, I, H\}$
7	B	12	18	13	16	34	12	23	19	40	∞	∞	$\{E, G, I, H\}$
8	H	12	18	13	16	34	12	23	19	22	25	∞	$\{E, G, I, J\}$
9	I	12	18	13	16	34	12	14	19	22	25	28	$\{E, G, J, K\}$

Clearly, after this, no costs will change and we have reached the goal node G. So, the search ends with the path $S \to A \to D \to H \to I \to G$. The total path cost is $g(G) = 4+5+1+1+3 = 14$.

Best First Search (BFS)

BFS is a greedy approach, where the node with the minimum heuristic value $h(n)$ is chosen to be expanded. The nodes in the Open set are sorted in ascending order of their heuristic values.

Iteration: 1	Open = $\{S\}$	Closed = $\{\}$
Iteration: 2	Open = $\{C, B, A\}$	Closed = $\{S\}$
Iteration: 3	Open = $\{F, E, B, D, A\}$	Closed = $\{S, C\}$
Iteration: 4	Open = $\{G, E, B, D, A\}$	Closed = $\{S, C, F\}$

The node with the minimum heuristic value is selected every time, expanded (add its children to the Open set), and then closed. Since we have reached the goal node G, we stop the search. So, the search path using BFS is $S \to C \to F \to G$. The total path cost is $g(G) = 11+2+13 = 26$.

Dijkstra's Algorithm

Dijkstra's algorithm is an uninformed searching technique that expands the node with the minimum path cost $g(n)$. The following table shows the distance of each node from the source node S after each iteration.

Iteration	Node	A	B	C	D	E	F	G	H	I	J	K	Frontier
0	-	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	$\{S\}$
1	S	4	18	11	∞	∞	∞	∞	∞	∞	∞	∞	$\{A, B, C\}$
2	A	4	12	11	9	∞	∞	∞	∞	∞	∞	∞	$\{B, D, C\}$
3	D	4	12	11	9	∞	10	∞	10	29	∞	∞	$\{B, C, F, I, H\}$
4	F	4	12	11	9	∞	10	23	10	29	∞	∞	$\{B, C, I, H, G\}$
5	H	4	12	11	9	∞	10	23	10	11	12	∞	$\{B, C, I, G, J\}$
6	C	4	12	11	9	∞	10	23	10	11	12	∞	$\{B, I, G, J, E\}$
7	I	4	12	11	9	∞	10	14	10	11	12	24	$\{B, G, J, E, K\}$
8	J	4	12	11	9	∞	10	14	10	11	12	19	$\{B, G, E, K\}$
9	B	4	12	11	9	∞	10	14	10	11	12	19	$\{G, E, K\}$

After iteration 9, we find that the node G has the minimum path cost, implying we have found the path with least cost to the goal node. The search path using Dijkstra's algorithm is $S \to A \to D \to H \to I \to G$. The total path cost is $g(G) = 14$, as shown above. Clearly, in this case, Dijsktra and A* provided the optimal answer but BFS did not.

Computational

The solution to the computational part of the assignment is given in the code in `main.py` in the submission. The code is a menu-driven program that uses the given dataset to find the path between two places entered by the user. The code implements the following algorithms:

1. Uniform Cost Search (UCS) or Dijsktra's Algorithm[1]

2. A* Search (using an admissible heuristic)

3. A* Search (using an unadmissible heuristic)

The code also provides the option to view all available locations. The code assumes that all roads given in the dataset are bidirectional.

Heuristics and Verification

Admissible Heuristic

For an admissible heuristic, I have used the straight line distance between the two locations as the heuristic. The coordinates to the points were obtained using the Indian Cities Dataset. The `geopy` library was used to find the distance between the two locations. Clearly, this makes an admissible heuristic, as the straight line distance (geodesic) on Earth can never be greater than the distance between two locations by road. Clearly, an A* search using this heuristic is optimal, as is verified in the code - the solution given by this matches the optimal solution.

Unadmissible Heuristic

For an unadmissible heuristic, I have used the same geodesic distance between the two locations scaled up by a factor of 5. This heuristic expands fewer nodes since the heuristic value is greater than the actual distance, and hence the nodes with lower heuristic values are expanded first. However, this heuristic is suboptimal, as can be seen in the code.
Clearly, this heuristic is unadmissible because it overshoots the distance estimates by nearly a factor of 5. For reference, the actaul distances between locations are of the order of 1000 km to 4500 km.

Comparison of the Algorithms

The Uniform Cost Search always provides an optimal solution, as it is equivalent to the Dijkstra's Algorithm, which means it explores all nodes and finds the shortest path or the path with the least cost. However, it is not efficient, as it explores all nodes, even if they are not on the optimal path.
In contrast, the A* search with an admissible heuristic is more efficient, as it explores fewer nodes that are more likely to lie on an optimal path. However, the A* search with an unadmissible heuristic provides a sub-optimal solution, but expands even fewer nodes.

Examples where A* (unadmissible) expands fewer nodes

Source	Destination	A* (admissible)	A* (unadmissible)
Agartala	Hubli	9	2
Delhi	Calcutta	8	1
Jullundur	Cochin	3	1

[1]UCS is a special case of A* Search with heuristic $h(n) = 0 \ \forall \ n$; this is implemented in the code.