ECS759P Artificial Intelligence: Coursework 1

Agenda Based Search

Agenda based search algorithms like Depth First Search, Breadth First Search and Uniform Cost Search are implemented in this section to find the path between two stations using London tube service. The data containing the information about the time taken to travel between stations, the tube lines, and the zones are loaded from tubedata.csv.

Once the data has been successfully imported, Graph class of Python library NetworkX is used to model the data.create_station_graph function is used to for this purpose.

Implement BFS, DFS, UCS

Class Station() has been created to represent the state of station and it has all the information needed to represent relationships between two stations such as the name of the station, the name of the previously visited station, the tube line that connects the two stations and the average time taken to travel from one station to the other one.

Function called cost_of_path has been defined to find the time taken from the starting station to the goal station and the route that was taken by the algorithm. This function is used by all the algorithms.

Compare BFS, DFS and UCS

Different routes have been tried for evaluating BFS, DFS, and UCS. Table 1 summarises the the cost of the path from starting station to the goal station and the number of stations that has been expanded by each algorithm for different routes that has been tried.

	DFS		BFS		UCS	
	Path Cost	No of stations expanded	Path Cost	No of stations expanded	Path Cost	No of stations expanded
Canada Water to Stratford	15	6	15	40	14	52
New Cross Gate to Stepney Green	27	32	14	26	14	18
Ealing Broadway to South Kensington	57	179	20	50	20	53
Baker Street to Wembley Park	13	3	13	16	13	70

Table 1 : Summary of DFS, BFS, UCS in terms of path cost and the number of stations expanded

It is noticeable from the Table 1 that Depth First Search algorithm does not always gives the shortest path from the start station to the goal station. DFS finds the shortest path only for Baker Street to Wembley Park. Stack data structure is used to find the path in case of DFS. DFS takes the station that was added to the frontier lastly and expands it until the goal station is found. The order in which different stations are added to the frontier is very important in the case of Depth First Search. If by chance the station that was removed from the frontier is on the way of the optimal path then DFS will be able to find the best route with the least number of expanded stations. The route found by DFS for Baker Street to Wembley Park illustrates this. But if the station that was removed from the frontier is the worst case possible then the algorithm might expand stations that are not leading to the goal state. When it expands all those stations and does not find the goal station then it might go back and expand the stations from the start. This is illustrated by the path found by DFS from Earling Broadway to South Kensington. The cost of the path and the number of stations expanded for this route is much higher than the other two algorithms. Therefore DFS is computationally expensive in this case.

Best First Search algorithm manages to find better paths from stating station to goal station as compared to Depth First Search Algorithm. But the DFS fails to find better path than UCS. In the case of the route from Canada Water to Stratford UCS finds better path than BFS. Queue data structure is used to find the path in case of BFS. BFS expands all the stations at a certain level before expanding stations at the next level. Order in which the stations are expanded matters in the case of BFS too.

Uniform Cost Search finds the best path for all the routes that was experimented upon. Uniform cost search does not depend upon the order in which different stations was added to the frontier unlike depth first search and best first search. Uniform cost search sort the

frontier with respect to the time that was taken to travel from the starting station to the goal station and pick the one which has the least time to expand. Although UCS finds the path with the least cost, it is more computationally expensive than DFS, BFS and UCS. In case of the route from Baker Street to Wembley Street, BFS was able to find the path by expanding just 16 stations, and
DFS was able to find the path by expanding just 3 stations while UCS had to expand
70 stations to find the best path. In the case of this route UCS was highly computationally expensive than the other two algorithms.

Although Uniform Cost Search is computationally expensive than Breadth First Search and Depth First Search, it always manages to find the best path between initial station and the goal station. The number of expanded stations for UCS can be reduced by making improvements to the cost function.

The path cost and the number of explored stations has been have been observed for all routes for two different order of processing the explored stations at each level. First the direct order of processing the explored stations was observed using reverse = False in the implementation of DFS, BFS and UCS. After that reverse order of explored stations at each level was observed using reverse = True in the implementation of each algorithm. The path cost and the number of explored stations for both order of exploring stations at each levels for all algorithms is illustrated in Table 2.

	DFS				BFS				UCS			
	Direct order		Reverse order		Direct order		Reverse order		Direct order		Reverse order	
	Path Cost	No of stations expanded	Path Cost	No of stations expanded	Path Cost	No of stations expanded	Path Cost	No of stations expanded	Path Cost	No of stations expanded	Path Cost	No of stations expanded
Canada Water to Stratford	15	6	20	227	15	40	15	25	14	52	14	47
New Cross Gate to Stepney Green	27	32	14	267	14	26	14	40	14	18	14	18
Ealing Broadway to South Kensington	57	179	51	157	20	50	20	47	20	53	20	52
Baker Street to Wembley Park	13	3	89	191	13	16	13	9	13	70	13	70

Table 2 : Summary of Path Cost and No of stations expanded for all algorithms with respect to different order of explored stations at each level

DFS takes the station that was lastly added to the frontier and tries to expand it in depth until it finds the goal state. As a result of this the order in which each station is added to the frontier at each level makes a significant impact on the cost of the path and the number of nodes expanded. In the case of the route between New Cross Gate to Stepney Green, the direct order gives a path with cost as 27 and the number of stations expanded is 32. While the reverse order gives path with cost as 14 and the number of expanded stations as 267.

BFS first expands all the stations at depth d before expanding the stations at depth d + 1. So the order of the order of processing the stations at each level does not affect the cost of the path found. But the number of expanded stations changes according to the order of the explored stations. In the case of the route between New Cross Gate to Stepney Green the cost of the path for the BFS is 14 for both orders while the number of explored stations is 26 in case of direct order and 40 in case of reverse order.

As mentioned earlier, UCS picks the station that has the least cost from the starting station to expand, so irrespective of the order in which different station are explored it will always find the path with the least cost. The number of expanded stations are different for different order as order in which nodes are processed matters in UCS when there are two or more nodes with the same cost.

Extending the cost function

Previously, we have been considering the cost it takes between to travel between two stations to decide the the route to take between two stations. It is also important to take into consideration the time taken to travel from one platform to another to change the line. In some cases it might not be worth to change the tube line as walking between different platforms to change line might take longer.

The extended cost function takes this into account and adds an extra time of 2 minutes for the time needed to walk to change the tube lines. Uniform cost search sorts the stations in the frontier according to the time needed to reach the stations. The time added to change the line between two stations will affect the order of the stations in the frontier and as such the station expanded first to search for the goal station also changes.

The UCS with the new extended cost function is compared with the UCS with the simple cost function and the cost of the path and the number of expanded stations are summarised in Table 3.

	UCS		UCS with extended cost function	
	Path Cost	No of stations expanded	Path Cost	No of stations expanded
Canada Water to Stratford	14	52	15	74
New Cross Gate to Stepney Green	14	18	14	25
Ealing Broadway to South Kensington	20	53	20	58
Baker Street to Wembley Park	13	70	13	89

Table 3 : Summary of the path cost and number of stations expanded for UCS and UCS with expanded cost function

As illustrated in the table 3, the new path cost tends to be comparable to the path cost of simple UCS. But, the number of expanded stations tends to be higher as the time taken to change lines is also taken into consideration now.

Although the path of the cost between starting station to goal station is same for most of the routes as can be seen from the table, for the route between Canada Water to Stratford, the time taken by the UCS with extended cost function is higher than the time taken with simple UCS. This is because the time taken to change the tube lines has not been taken into consideration while calculating the cost of the path from the start station to the end station. The path taken by Uniform cost search with extended cost function is Canada Water → Canary Wharf → North Greenwich → Canning Town → West Ham → Stratford while the path taken by simple Uniform Cost search is Canada Water → Rotherhithe → Wapping → Shadwell → Whitechapel → Stepney Green → Mile End → Stratford. A close analysis of the two different routes shows us that the route given by Simple UCS changes the tube two times while the route given by UCS with improved cost function does not change tube at all. This proves that although the path cost is 14 in reality the time taken by Simple UCS is much higher than the the improved UCS.

In conclusion, although the number of nodes expanded by UCS with improved cost function is higher, it gives a better path than simple UCS.

Heuristic Search

Best first search algorithm tries to explore stations that are closest to the goal stations before exploring stations that are far away to try the path between start station to the goal station. The algorithm evaluates each station by finding the finding the heuristics of that station and the station with least heuristics value is expanded first. The heuristics for each station station is found using the knowledge of the zone of the current station. Class StationDetails have been created to represent the details of the stations including the zone details.

As it can be seen from the London tube map, the zone number increases as we move away from the centre of the map. There will be three important cases in terms of zones when travelling between stations having different zones : Case 1 : Start station having lower zone number than end station

Case 2 : Start station having higher zone number than the end station

Case 3 : Start station and end station has same zone number

The heuristic value associated with a station is explained by taking the example of start station having zone 2 and end station having zone 5(Case 1). If the path that is found by best first station taken it to a lower station 2, it will have a higher heuristics(4 minutes) associated with it. If the path takes it to a higher zone than 5 that is the algorithm is taking it away from the goal state then also it will have a higher heuristics of 4 minutes. If the best first search takes the path to a zone lower than the zone of end station and higher than the

zone of the start station it means that the algorithm is travelling in the right direction and the heuristics that will be associated will be the lowest(2 minutes). If the best first search algorithm stays in the same zone while expanding the stations then a heuristic value that is between the best and the worst cases will be associated with it. Here we take a heuristic value of 3 minutes. Since the stations with the lowest heuristics value are expanded first, the best first search algorithm will first try to expand stations that are in the least cost before expanding other stations.

Similar heuristics values are associated with each station in the second case as well. It is further explained by taking an example of start station having zone 8 and end station having zone 4. If the path found by the algorithm takes it to a station higher than 8 then a higher heuristics of 4 is associated with it. If the path takes it to a station having same zone then a heuristics of 3 is taken. If the path takes it a zone lower than end station but higher than the start station the a lower heuristics value of 2 is associated with that station. If the path takes it to a station lower than four then the path followed will be similar to the previously explained example.

A summary of the path of the cost and the number of expanded stations for Best First Search and Improved Uniform Cost Search is given in Table 4.

	UCS with extended cost function		Best First Search	
	Path Cost	No of stations expanded	Path Cost	No of stations expanded
Canada Water to Stratford	15	74	15	20
New Cross Gate to Stepney Green	14	25	14	14
Ealing Broadway to South Kensington	20	58	20	36
Baker Street to Wembley Park	13	89	13	8

Table 5 : Summary of the path cost and number of stations expanded for UCS and UCS with expanded cost function

As it can be seen from the table that the Best first search algorithm finds the better path with less number of expanded stations as compared to UCS with expanded cost function. Because the heuristics of the stations that move in the correct path is lower, the stations that move in the correct direction are removed and expanded from the frontier first. As a consequence of this best first search algorithm is able to find the best path without expanding many stations.

Genetic Algorithm

Genetic Algorithm is a search heuristic in which the fittest individuals are selected for reproduction in order to produce individuals for the next generation

Implement a Genetic Algorithm

The secret phrase that was obtained by the genetic algorithm is ---. The closeness function was called with student username ---.

Elements of the Algorithm

The secret phrase and the fitness is represented by a class called Chromosome.
Chromosome class stores the secret phrase and the fitness of the secret phrase obtained from the closeness function.
Population of each generation is represented as a list of objects of chromosome class. Each secret phrase of initial population is generated randomly from the allowed characters(A-Z, 0-9,_) using python random module.

Tournament selection strategy has been used to pick the fittest individual from the current population and pass it on to the next generation. From the list of chromosomes in the population, 3(Size of the tournament) chromosomes have been picked at random and sorted in the decreasing order of their fitness value. Two chromosome having highest fitness are selected as parents for the next generation.

N point crossover technique is used to perform crossover between two fittest parents. N points are selected randomly within the phrase length and the crossover between two parents in performed at these randomly selected points. The two resulting children are then added to the next generation with their fitness as None.

Each chromosome in the population is mutated with a probability of 0.04. Each chromosome in the population is mutated by replacing replacing a character at a randomly generated index by randomly generated character from the gene set.

The number of reproductions

Number of reproductions = (Population Size / 2) * (Number of generation - 1)

One is decreased from the number of generations as the initial generation is not taken while counting the number of reproductions.

The average and variance obtained for the number of generations and number of reproductions after running the code multiple times are recorded in the Table 6. On an average it takes 194 generations and 9675 reproductions to converge to the secret phrase.

	Number of generation	Number of reproductions
1	397	19800
2	285	14200
3	201	10000
4	230	11450
5	151	7500
6	216	10750
7	54	2650
8	77	3800
9	176	8750
10	158	7850
Average	194.5	9675
Variance	8807.45	22018625

Table 6 : The number of generations and number of reproductions obtained by running the genetic algorithm multiple times

Hyperparameters

The difference hyper-parameters in the genetic algorithm are Mutation probability, Population size, the number of crossover points and tournament size used for the selection.

The effect the population have on the number of generations needed to converge to the secret phrase is illustrated in table 7. The effect of population size is studied by keeping other hyper-parameters same and by seeding the random variable. A seed of 10 is used for this study. The values of other hyper-parameters are as follows mutation probability = 0.04, tournament size = 4, crossover points = 3. The experiment showed that as the population size increased, the number of generations that was required to converge to the secret phrase decreased. It was also observed that after a certain population size the decrease in the number of generations needed to converge to secret phrase did not show any significant reduction. It was also observed that the time required to run the genetic algorithm also increased with the size of the population. Optimal population size without a higher run time was found to be 1000.

	Population size	Number of generations
1	5	17214
2	10	2216
3	100	212
4	1000	14
5	10000	12
6	100000	11

Table 7 : Summary of the change in the number of generations when the population size is changed in genetic algorithm