Extended Essay

School Bus Route Optimisation

To what extent do nearest neighbour, dynamic programming and ant colony optimization algorithms find the shortest possible route for a school bus?

Subject : Mathematics

Session : November 2023

Word Count : 3998

Contents

1.0 Introduction

The Galaxy School provides door-to-door transport to 1,500 students and faculty daily. Students spend 1.5-2 hours on the bus each day. I realized that mathematics could be used to optimize bus routes. By applying mathematics, I could construct the shortest possible route for a bus, which could reduce the time and cost of school bus transportation. This would benefit both the school (by lowering costs) and the students (by reducing transportation time). Hence it leads to my research question :

To what extent do nearest neighbour, dynamic programming and ant colony optimization algorithms find the shortest possible route for a school bus?

2.0 Approach

The bus routing problem is similar to the traveling salesman problem(TSP). In the TSP, a salesman starts in one city and travels to a number of other cities, then returns to the starting city. The goal is to find the shortest possible route that visits each city exactly once. There are three main types of algorithms that can be used to solve the TSP: exact algorithms, heuristics, and approximations.

- Exact algorithms guarantee optimal solutions but have high time complexity[1], making them impractical for large-scale problems.
- Heuristics are faster than exact algorithms and can provide good solutions to large-scale problems. They are not guaranteed to be optimal, but they can often find solutions that are close to optimal.
- Approximation algorithms provide solutions that are guaranteed to be within a certain factor of the optimal solution. They are less accurate than heuristics, but they can handle large-scale data and do not have a high time complexity.

To solve the school bus routing problem exact algorithm and heuristics were chosen. Exact algorithm would be a long process but guarantees a more reliable result. Heuristics too may give reliable result and will be less complex.
The algorithms used in this exploration are :

- Heuristics
  - Nearest Neighbour
  - Ant colony optimisation
- Exact algorithm
  - Dynamic Programming

[1]Time complexity is defined as the amount of time taken by an algorithm to run, as a function of the length of the input.

The procedure followed throughout the exploration is represented in the flowchart shown in figure 1

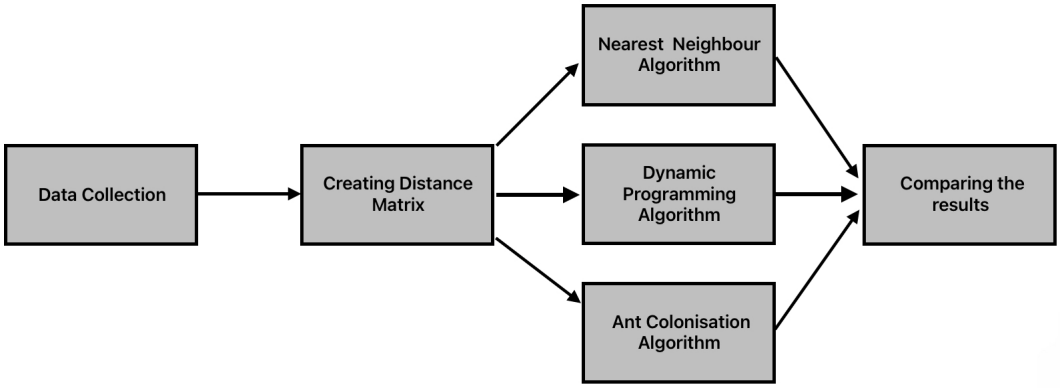


Figure 1: Flow Chart

3.0 Data Collection

The transport department of The Galaxy School provided data in raw form (addresses of the stops) for each school bus. A bus with 14 stops was selected to apply an exact algorithm to find the optimum route. A distance matrix was created to calculate the shortest distance between each stop.

1. Each of the bus stop had an address, which was used to pin their location on the google maps.

2. Each of the stop was numbered in order to label the stops and chart out the routes. The numbers were assigned randomly as shown in figure 2

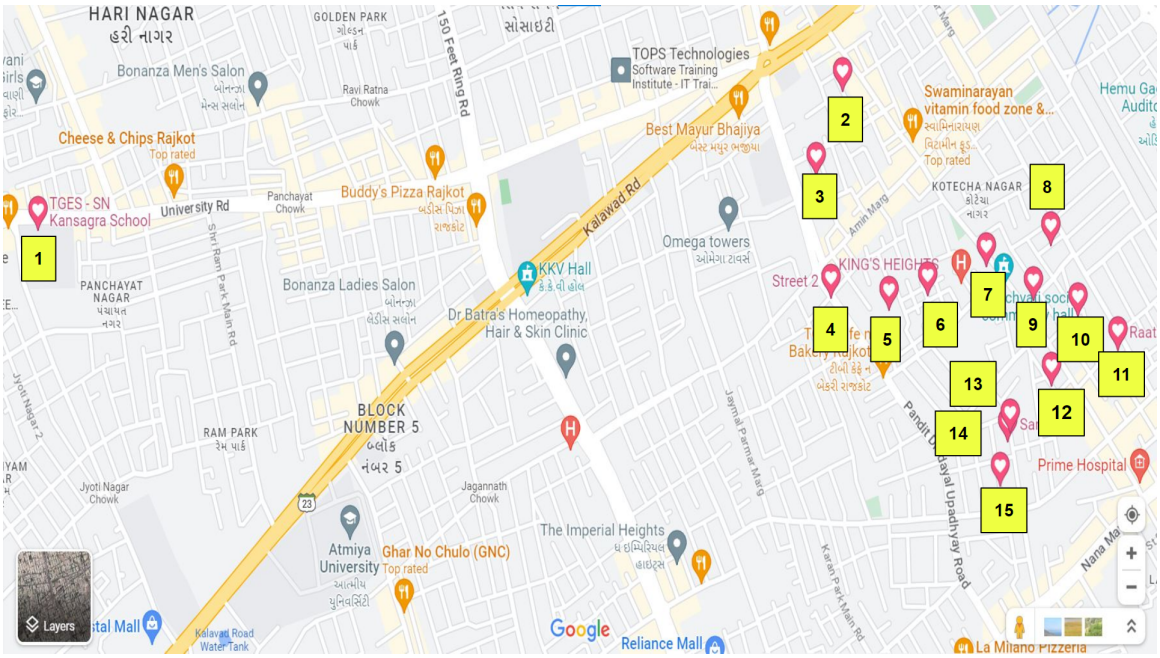


Figure 2: Bus stops on Google Maps

3. Point 1(school) was selected and shortest distance from point 1 to all other points(various stops) was found using the Google Maps. The method of measuring distance from stop 1 to stop 2 is shown in figure 3.

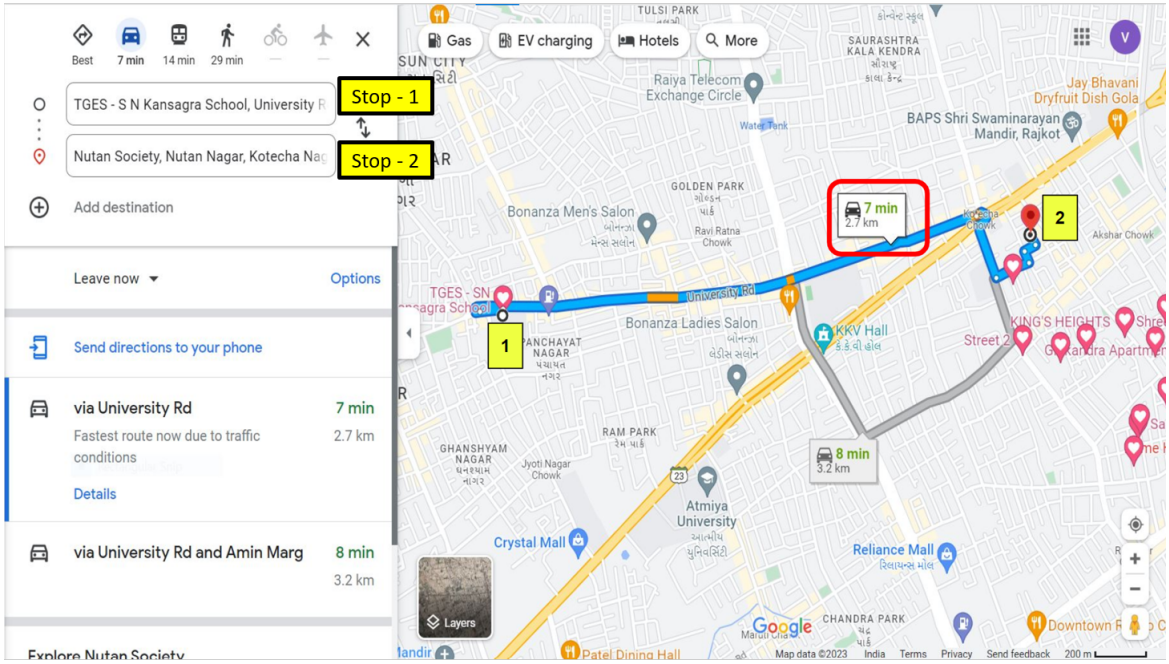


Figure 3: Distance from stop 1 to 2

4. Similarly distances for other stops were calculated to create a final distance matrix. All the distances are given in meters. The distance matrix is shown in the table 1 The route begins and ends at stop 1, which represents the school. This is because all students and teachers are picked up and dropped off at the school.

Stop No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	2700	2500	2700	3200	3200	3200	3600	3400	3500	3800	3600	3400	3400	3500
2	2700	0	290	750	1000	950	550	850	750	900	1100	950	1000	1100	1400
3	2500	290	0	350	750	700	650	850	800	1000	1100	1000	950	1000	1000
4	2700	750	350	0	400	450	500	750	700	850	1000	800	650	660	650
5	3200	1000	750	400	0	85	400	650	450	600	850	600	450	500	500
6	3200	950	700	450	85	0	300	550	400	550	750	550	450	450	550
7	3200	550	650	500	400	300	0	300	150	300	500	350	450	450	600
8	3600	850	850	750	650	550	300	0	170	190	300	400	550	550	700
9	3400	750	800	700	450	400	150	170	0	70	350	240	350	400	500
10	3500	900	1000	850	600	550	300	190	70	0	400	260	400	450	550
11	3800	1100	1100	1000	850	750	500	300	350	400	0	500	700	700	650
12	3600	950	1000	800	600	550	350	400	240	260	500	0	290	270	350
13	3400	1000	950	650	450	450	450	550	350	400	700	290	0	18	150
14	3400	1100	1000	660	500	450	450	550	400	450	700	270	18	0	130
15	3500	1400	1000	650	500	550	600	700	500	550	650	350	150	130	0

Table 1: Distance Matrix

3.10 Original Route

The bus driver's route is shown in figure 4. The results of the algorithms will be compared to this route.

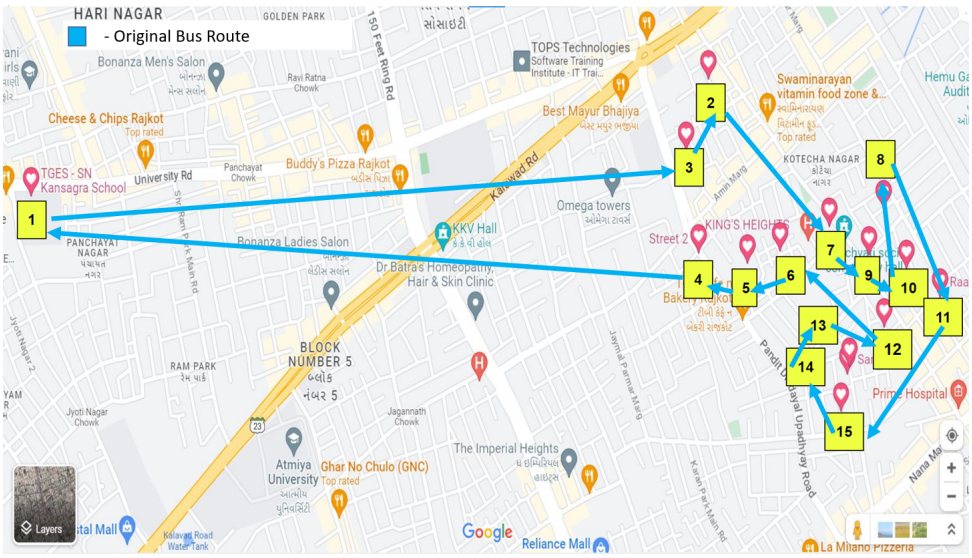


Figure 4: Original Route

The original route is :

$$1 \longrightarrow 3 \longrightarrow 2 \longrightarrow 7 \longrightarrow 9 \longrightarrow 10 \longrightarrow 8 \longrightarrow 11 \longrightarrow 15 \longrightarrow 14 \longrightarrow 13 \longrightarrow 12 \longrightarrow 6 \longrightarrow 5 \longrightarrow 4 \longrightarrow 1$$

The total distance of the original route route is :

$$2500 + 290 + 550 + 150 + 70 + 190 + 300 + 650 + 130 + 18 + 290 + 550 + 85 + 400 + 2700 = 8873\text{meters}$$

4.0 Nearest Neighbour Approach

Nearest neighbor approach starts from a stop and selects the nearest neighbor until all stops are covered without repeating any of the stops. Here is the process for nearest neighbour approach:

1. The starting point selected for the school bus route was school itself which is represented by stop number 1.

2. The nearest stop from stop 1 is stop 3, with a distance of 2500 meters. The cell representing the journey from stop 1 to stop 3 was highlighted in blue, and the cells representing journeys from stop 1 to other stops were highlighted in black. This is to ensure that the route does not pass through stop 1 again.

Stop No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1															
2		0	290	750	1000	950	550	850	750	900	1100	950	1000	1100	1400
3	2500	290	0	350	750	700	650	850	800	1000	1100	1000	950	1000	1000
4		750	350	0	400	450	500	750	700	850	1000	800	650	660	650
5		1000	750	400	0	85	400	650	450	600	850	600	450	500	500
6		950	700	450	85	0	300	550	400	550	750	550	450	450	550
7		550	650	500	400	300	0	300	150	300	500	350	450	450	600
8		850	850	750	650	550	300	0	170	190	300	400	550	550	700
9		750	800	700	450	400	150	170	0	70	350	240	350	400	500
10		900	1000	850	600	550	300	190	70	0	400	260	400	450	550
11		1100	1100	1000	850	750	500	300	350	400	0	500	700	700	650
12		950	1000	800	600	550	350	400	240	260	500	0	290	270	350
13		1000	950	650	450	450	450	550	350	400	700	290	0	18	150
14		1100	1000	660	500	450	450	550	400	450	700	270	18	0	130
15		1400	1000	650	500	550	600	700	500	550	650	350	150	130	0

Table 2: Nearest Neighbour - 1

3. The algorithm started at stop 1 and went to stop 3. The nearest stop from stop 3 is stop 2, so the algorithm highlighted the cell representing the journey from stop 3 to stop 2 in blue. The cells representing journeys from stop 3 to other stops were highlighted in black to prevent the algorithm from repeating stop 3.

Stop No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1															
2		0	290	750	1000	950	550	850	750	900	1100	950	1000	1100	1400
3	2500														
4		750		0	400	450	500	750	700	850	1000	800	650	660	650
5		1000		400	0	85	400	650	450	600	850	600	450	500	500
6		950		450	85	0	300	550	400	550	750	550	450	450	550
7		550		500	400	300	0	300	150	300	500	350	450	450	600
8		850		750	650	550	300	0	170	190	300	400	550	550	700
9		750		700	450	400	150	170	0	70	350	240	350	400	500
10		900		850	600	550	300	190	70	0	400	260	400	450	550
11		1100		1000	850	750	500	300	350	400	0	500	700	700	650
12		950		800	600	550	350	400	240	260	500	0	290	270	350
13		1000		650	450	450	450	550	350	400	700	290	0	18	150
14		1100		660	500	450	450	550	400	450	700	270	18	0	130
15		1400		650	500	550	600	700	500	550	650	350	150	130	0

Table 3: Nearest Neighbour - 2

4. The algorithm charted out the route from stop 1 to stop 3 to stop 2. Using the same method mentioned in the steps above the algorithm was carried out and the final route is represented below in table 4

Stop No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1															
2			290												
3	2500														
4				0	400										
5						85									
6												550			
7		550													
8										190					
9							150								
10									70						
11								300							
12													290		
13														18	
14															130
15											650				

Table 4: Nearest Neighbour - Final

The final route charted out by the nearest neighbour algorithm is :

$$1 \longrightarrow 3 \longrightarrow 2 \longrightarrow 7 \longrightarrow 9 \longrightarrow 10 \longrightarrow 8 \longrightarrow 11 \longrightarrow 15 \longrightarrow 14 \longrightarrow 13 \longrightarrow 12 \longrightarrow 6 \longrightarrow 5 \longrightarrow 4 \longrightarrow 1$$

The total distance of the route given by nearest neighbour algorithm is :

$$2500 + 290 + 550 + 150 + 70 + 190 + 300 + 650 + 130 + 18 + 290 + 550 + 85 + 400 + 2700 = 8873\text{meters}$$

The result given by nearest neighbour approach is represented on the map shown in figure 5

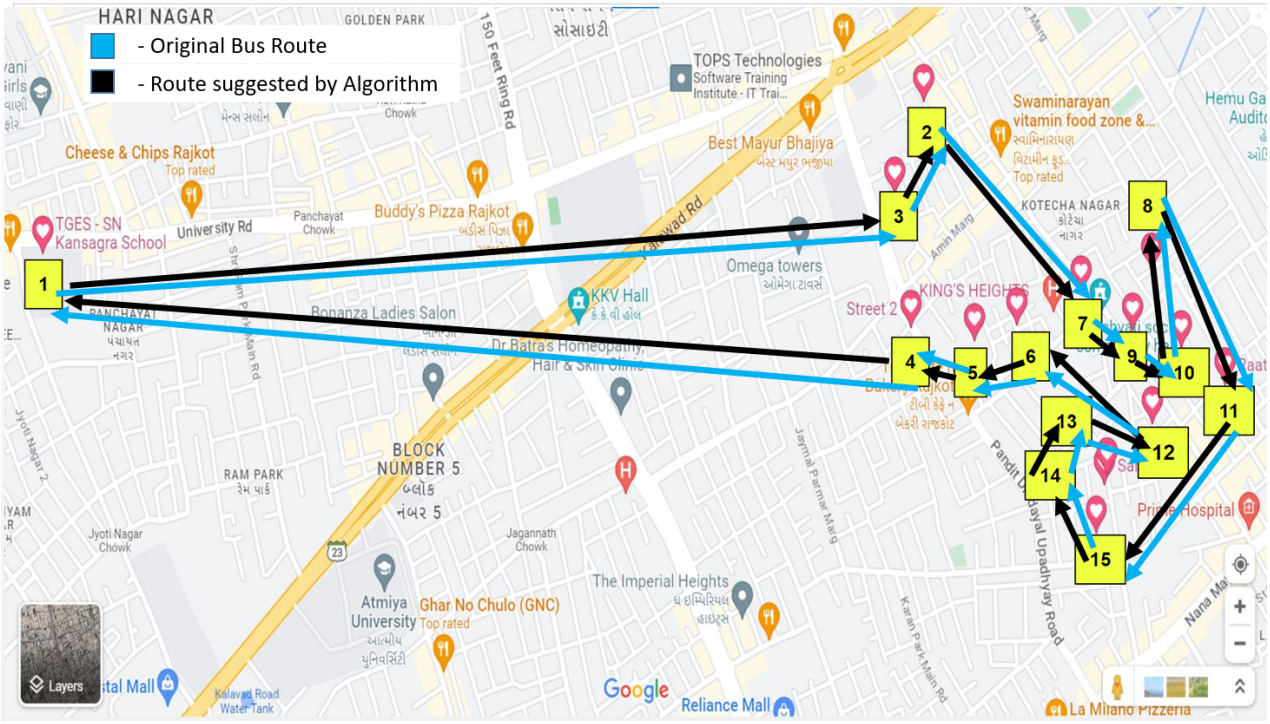


Figure 5: Nearest Neighbour Result

The nearest neighbor algorithm is a simple and practical algorithm for bus routing. It is easy to understand and implement, and it does not require a lot of computational time. The algorithm provides the same route as the original route. Bus driver too uses the same algorithm. This shows the simplicity and practicality of the algorithm. The graphical version of the solution shows that the route does not have any loops, which is credible.

5.0 Dynamic Programming

Dynamic programming is an algorithm that breaks the problem into smaller sub-problems. It saves the result of solving smaller sub-problems for future purposes so that it is not computed again. Dynamic programming optimises the sub-problems to optimise the complex problem. [2].

The steps followed by the dynamic programming are :

1. Breaking down on complex problem into smaller and simpler sub-problems.
    - Grouping all the 15 stops in 3 clusters using the K means algorithm.
2. Finding optimal solution to these sub problems.
    - Using dynamic programming to find the optimum route through each of these clusters.
3. Storing the results of sub-problems.
    - Noting down the results of dynamic programming applied to the clusters.
4. Calculate the result of complex problem.

[2]https://www.javatpoint.com/dynamic-programming: :text=Dynamic%20programming%20is%20a%20technique,known%20as%20optimal%20su

- Combining the results of all the clusters and getting the final result.

Dynamic programming would take a long time to solve the bus routing problem with 15 stops, so the stops were classified into 3 clusters of 5 using the K-means algorithm. This makes it easier to apply dynamic programming to each cluster, reducing the complexity.

5.10 K-means Clustering

5.10.1 K-means clustering algorithm

K-means clustering algorithm, "*It is an iterative algorithm that divides the unlabeled dataset into k different clusters in such a way that each dataset belongs only one group that has similar properties.*"[3] Distance based measurements are used to identify the similarities between the data points.

The K-means algorithm involves defining the number of clusters (K) and randomly selecting K centroids. A centroid is a data point that represents the center of the cluster (the mean), and it might not necessarily be a member of the dataset[4]. Each data point is then associated with the cluster whose centroid is closest to it. This process is repeated until the clusters no longer change. The maximum decimal accuracy provided by Google Maps was used to ensure that the clusters were formed accurately.

5.10.2 Process of K-means algorithm

1. Each stop was given a co-ordinate, this was done by the co-ordinate plotting system of Google Maps.

Stops	**Co-ordinates**	
	X	**Y**
1	22.28822779	70.7602492
2	22.29037206	70.78024775
3	22.28906167	70.77968985
4	22.28691738	70.77999026
5	22.28648058	70.78153521
6	22.28635036	70.78251657
7	22.28703269	70.78407489
8	22.28743179	70.7855219
9	22.28632461	70.78504884
10	22.28598988	70.78617584
11	22.28535904	70.78716371
12	22.28481832	70.78557756
13	22.28395573	70.78460361
14	22.28367249	70.78447839
15	22.28297727	70.78432534

Table 5: Coordinates of stops

[3]https://www.javatpoint.com/K-means-clustering-algorithm-in-machine-learning
[4]https://www.pinecone.io/learn/k-means-clustering/

2. 3 random (no matter what point is selected, final result will be the same) points (C1, C2, C3) were selected to be set as centroids.

C1		C2		C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.28815022	70.78148298	22.28659521	70.78433731	22.28498171	70.78434266

Table 6: Iteration 1

3. Distances of each point from the centroids were measured using the Euclidian distance formula : $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Each point was assigned to the centroid closest to it. x_1 and y_1 represent the co-ordinates of centroids. x_2 and y_2 represent the co-ordinates of the stops.

Stops	**Co-ordinates**		**Distances**			**Cluster**
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0.021233922	0.024143369	0.02431115	C1
2	22.29037206	70.78024775	0.002542119	0.005566785	0.006769352	C1
3	22.28906167	70.77968985	0.002011484	0.0052614	0.006188274	C1
4	22.28691738	70.77999026	0.001936002	0.004358973	0.004763428	C1
5	22.28648058	70.78153521	0.001670457	0.002804441	0.003182513	C1
6	22.28635036	70.78251657	0.002075527	0.001837123	0.002282058	C2
7	22.28703269	70.78407489	0.002822568	0.00051015	0.002068381	C2
8	22.28743179	70.7855219	0.004102325	0.001450223	0.002719095	C2
9	22.28632461	70.78504884	0.004006025	0.000761254	0.001517253	C2
10	22.28598988	70.78617584	0.005166241	0.001935624	0.002092115	C2
11	22.28535904	70.78716371	0.006329405	0.003084908	0.002846167	C3
12	22.28481832	70.78557756	0.005278936	0.002166926	0.001245661	C3
13	22.28395573	70.78460361	0.005228008	0.002652876	0.00105865	C3
14	22.28367249	70.78447839	0.005387254	0.002926118	0.00131624	C3
15	22.28297727	70.78432534	0.005902408	0.003617958	0.002004522	C3

Table 7: K-means iteration - 1

Figure 6 provides a visual representation of the classification of the stops based on their distances relative to each other.

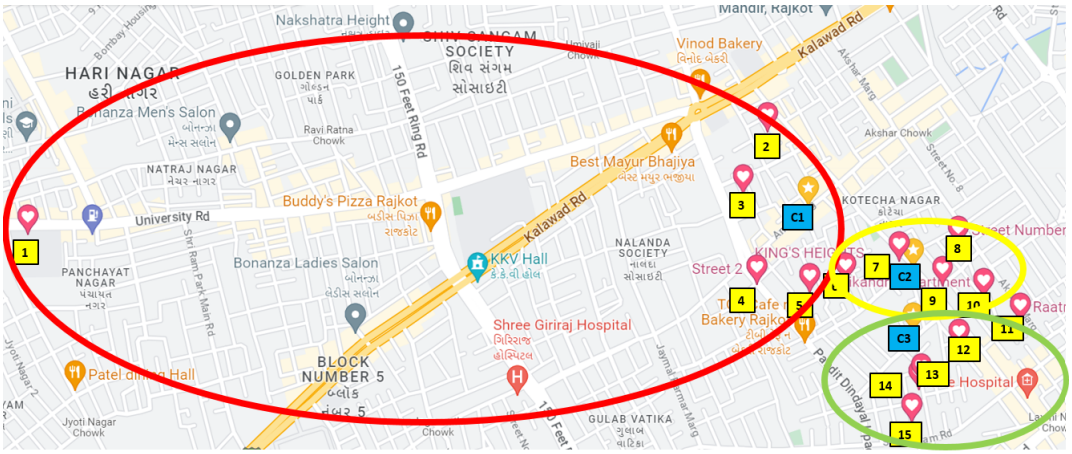


Figure 6: Representation of K-means iteration 1 on maps

4. The new centroids were calculated. The x co-ordinate of new centroid is the average of all the x co-ordinates of stops assigned to that particular centroid. The y co-ordinate of new centroid is the average of all the y co-ordinates of stops assigned to that particular centroid. This was done using 'Average' function in Microsoft Excel spread sheet.

AVG C1		**AVG C2**		**AVG C3**	
X	**Y**	**X**	**Y**	**X**	**Y**
22.2882119	70.77634245	22.28662586	70.78466761	22.28415657	70.78522972

Table 8: Iteration 2

5. Now the distance of each stop from new centroids is calculated using the Euclidean distance formula. Each point is assigned to the centroid closest to it.

Stops	Co-ordinate		Distances			Cluster
	X	Y	C1	C2	C3	
1	22.28822779	70.7602492	0.014800938	0.023951611	0.025310103	C1
2	22.29037206	70.78024775	0.005482695	0.005422987	0.007965699	C2
3	22.28906167	70.77968985	0.004664259	0.005089689	0.00739934	C1
4	22.28691738	70.77999026	0.005238947	0.004167266	0.005922337	C2
5	22.28648058	70.78153521	0.006842215	0.002613142	0.004364681	C2
6	22.28635036	70.78251657	0.007816619	0.001648241	0.003489109	C2
7	22.28703269	70.78407489	0.009173382	0.000436792	0.003099307	C2
8	22.28743179	70.7855219	0.010547616	0.001607327	0.003288228	C2
9	22.28632461	70.78504884	0.010270082	0.000944829	0.002175573	C2
10	22.28598988	70.78617584	0.011443789	0.002120467	0.00206305	C3
11	22.28535904	70.78716371	0.012556938	0.003263954	0.002277332	C3
12	22.28481832	70.78557756	0.011206773	0.002287128	0.000747597	C3
13	22.28395573	70.78460361	0.010647431	0.002685279	0.000657534	C3
14	22.28367249	70.78447839	0.01066423	0.002948009	0.000893774	C3
15	22.28297727	70.78432534	0.010874669	0.00362884	0.001486156	C3

Table 9: K-means iteration - 2

Figure 7 provides a visual representation of the classification of the stops based on their distances relative to each other.

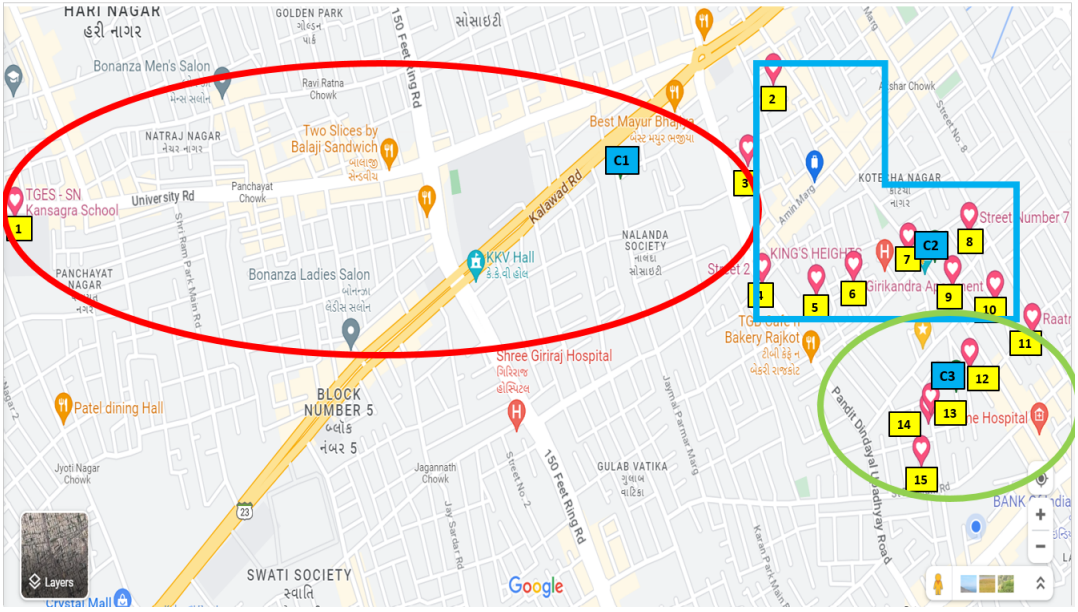


Figure 7: Representation of K-means iteration 2 on maps

6. The new cluster is compared to the old cluster. If there is no change, the cluster is optimum. Otherwise, steps 4 and 5 are repeated.

The calculations of all other iterations are given in appendix. The final solution provided by the K-means algorithms is shown in the table 10

Stops	Co-ordinates		Distances			Cluster
	X	Y	C1	C2	C3	
1	22.28822779	70.7602492	0	0.020550456	0.025142554	C1
2	22.29037206	70.78024775	0.020113177	0.002594227	0.007112953	C2
3	22.28906167	70.77968985	0.019458527	0.001650661	0.006695994	C2
4	22.28691738	70.77999026	0.019784502	0.001222177	0.005477639	C2
5	22.28648058	70.78153521	0.021357598	0.001544286	0.003872961	C2
6	22.28635036	70.78251657	0.02234638	0.002273538	0.002904877	C2
7	22.28703269	70.78407489	0.023855649	0.003376031	0.002089117	C3
8	22.28743179	70.7855219	0.025285239	0.004743268	0.002168419	C3
9	22.28632461	70.78504884	0.024872565	0.004513629	0.001053776	C3
10	22.28598988	70.78617584	0.026023051	0.005687986	0.001188738	C3
11	22.28535904	70.78716371	0.027066964	0.006832716	0.001946231	C3
12	22.28481832	70.78557756	0.025556807	0.005654459	0.000588304	C3
13	22.28395573	70.78460361	0.024726258	0.005436739	0.001464445	C3
14	22.28367249	70.78447839	0.024653688	0.005558663	0.001774091	C3
15	22.28297727	70.78432534	0.024642006	0.006005665	0.002474357	C3

Table 10: k-means iteration - 7

Table 10 gives the final result on K-means algorithm. It has classified all the stops into 3 clusters with stop 1 being in cluster 1. Stops 2,3,4,5 and6 being in cluster 2 and stops 7,8,9,10,11,12,13,14 and 15 being in cluster 3. Dynamic programming will be applied on cluster 2 and 3 and not on cluster 1 since it contains only 1 stop.

5.20 Dynamic programming - Calculation

Dynamic programming was used to find the optimum route for the clusters 2 and 3. For cluster 2, the dynamic programming was performed manually, while for cluster 3, it was performed using a computer program. This was because cluster 3 is larger and calculating the dynamic programming manually would be too tedious.
The general formula[5] for solving a dynamic programming problem is :

$$g(i,s) = min(w(i,j) + g(j,\{s-j\})) \qquad j \in s \tag{1}$$

- $g(i,s)$ is the distance function which is to be minimised since we have to find the shortest distance.
  - i represents the initial vertex, the stop from which the bus starts its journey.

[5]https://youtu.be/hh-uFQ-MGfw

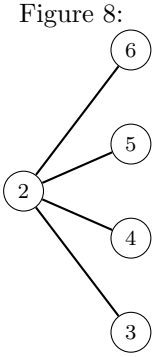
- s represents the set of stops that the bus will visit exactly once and then again return to the initial vertex. The set s will not include the starting stop.

- $'min'$ represents to minimise the function

- $w(i, j)$ represents the weight from starting stop i to some other stop j. Here weight represents the distance between the 2 stops which we can get from the distance matrix (table 1).

- $g(j, \{s - j\})$ represents the distance function of journey from stop j to any other stop in the set $s - j$.

Figure 8:



$$g(2, \{3, 4, 5, 6\}) = \min[w(2, 3) + g(3, \{4, 5, 6\})] \tag{2.3}$$

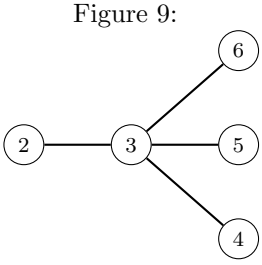
$$= \min[w(2, 4) + g(4, \{3, 5, 6\})] \tag{2.4}$$

$$= \min[w(2, 5) + g(5, \{3, 4, 6\})] \tag{2.5}$$

$$= \min[w(2, 6) + g(6, \{3, 4, 5\})] \tag{2.6}$$

Figure 8 showcases the possibilities of bus travelling from stop 2. It can travel to stops 3,4,5 or 6. The equations representing each of these possibilities are represented towards its right. The aim is to find minimum value out of each of these equations. However each of the equation contains the function g which is the distance function to be minimised and hence it needs to be solved. The figures and equations below represent the further solution of the equations until a numerical value is found for the function g. This process of finding the minimum value for the function $g(2, 3, 4, 5, 6)$ will provide the optimum route as well as the distance of the route for the stops 2 to 6 starting from stop 2.

Figure 9:

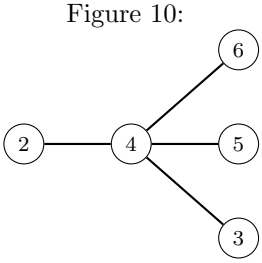


$$g(3, \{4, 5, 6\}) = \min[w(3, 4) + g(4, \{5, 6\})] \quad (2.3.4)$$

$$= \min[w(3, 5) + g(5, \{4, 6\})] \quad (2.3.5)$$

$$= \min[w(3, 6) + g(6, \{4, 5\})] \quad (2.3.6)$$

Figure 10:

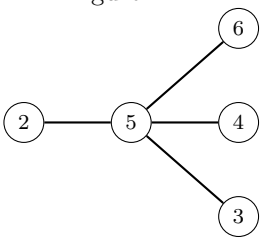


$$g(4, \{3, 5, 6\}) = \min[w(4, 3) + g(3, \{5, 6\})] \quad (2.4.3)$$

$$= \min[w(4, 5) + g(5, \{3, 6\})] \quad (2.4.5)$$

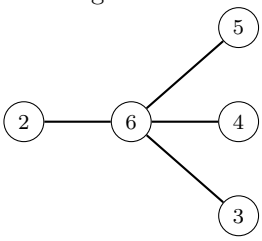
$$= \min[w(4, 6) + g(6, \{3, 5\})] \quad (2.4.6)$$

Figure 11:



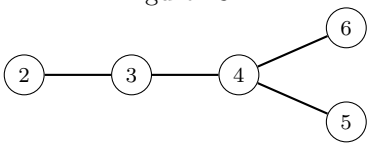
$$g(5,\{3,4,6\}) = \min[w(5,3) + g(3,\{4,6\})] \quad (2.5.3)$$
$$= \min[w(5,4) + g(4,\{3,6\})] \quad (2.5.4)$$
$$= \min[w(5,6) + g(6,\{3,4\})] \quad (2.5.6)$$

Figure 12:



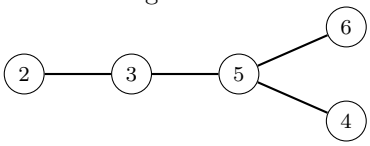
$$g(6,\{3,5,4\}) = \min[w(6,3) + g(3,\{5,4\})] \quad (2.6.3)$$
$$= \min[w(6,4) + g(4,\{3,5\})] \quad (2.6.4)$$
$$= \min[w(6,5) + g(5,\{3,4\})] \quad (2.6.5)$$

Figure 13:



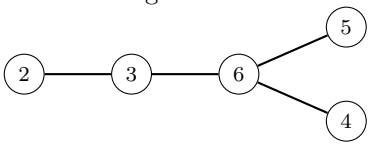
$$g(4,\{5,6\}) = \min[w(4,5) + g(5,\{6\})] \quad (2.3.4.5)$$
$$= \min[w(4,6) + g(6,\{5\})] \quad (2.3.4.6)$$

Figure 14:



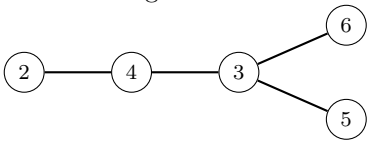
$$g(5,\{4,6\}) = \min[w(5,4) + g(4,\{6\})] \quad (2.3.5.4)$$
$$= \min[w(5,6) + g(6,\{4\})] \quad (2.3.5.6)$$

Figure 15:



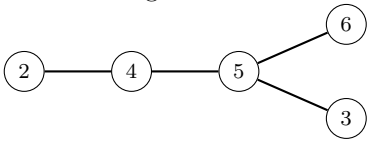
$$g(6,\{4,5\}) = \min[w(6,4) + g(4,\{5\})] \quad (2.3.6.4)$$
$$= \min[w(6,5) + g(5,\{4\})] \quad (2.3.6.5)$$

Figure 16:



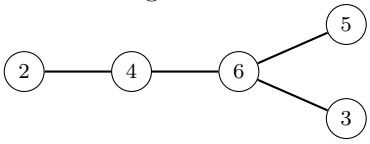
$$g(3,\{5,6\}) = \min[w(3,5) + g(5,\{6\})] \quad (2.4.3.5)$$
$$= \min[w(3,6) + g(6,\{5\})] \quad (2.4.3.6)$$

Figure 17:



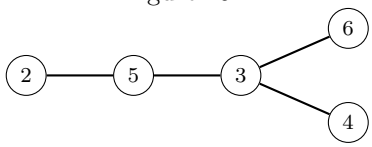
$$g(5,\{3,6\}) = \min[w(5,3) + g(3,\{6\})] \quad (2.4.5.3)$$
$$= \min[w(5,6) + g(6,\{3\})] \quad (2.4.5.6)$$

Figure 18:



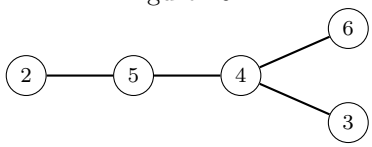
$$g(6,\{3,5\}) = \min[w(6,3) + g(3,\{5\})] \quad (2.4.6.3)$$
$$= \min[w(6,5) + g(5,\{3\})] \quad (2.4.6.5)$$

Figure 19:



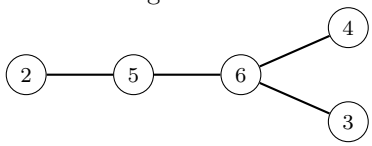
$$g(3,\{4,6\}) = \min[w(3,4) + g(4,\{6\})] \quad (2.5.3.4)$$
$$= \min[w(3,6) + g(6,\{4\})] \quad (2.5.3.6)$$

Figure 20:



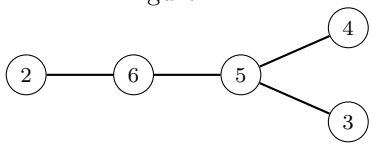
$$g(4,\{3,6\}) = \min[w(4,3) + g(3,\{6\})] \quad (2.5.4.3)$$
$$= \min[w(4,6) + g(6,\{3\})] \quad (2.5.4.6)$$

Figure 21:



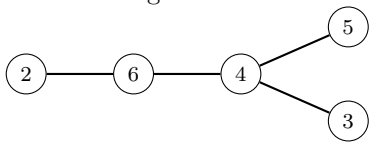
$$g(6,\{3,4\}) = \min[w(6,3) + g(3,\{4\})] \quad (2.5.6.3)$$
$$= \min[w(6,4) + g(4,\{3\})] \quad (2.5.6.4)$$

Figure 22:



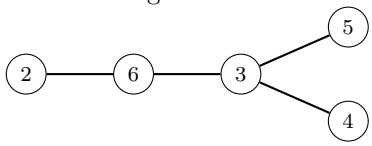
$$g(5,\{3,4\}) = \min[w(5,3) + g(3,\{4\})] \quad (2.6.5.3)$$
$$= \min[w(5,4) + g(4,\{3\})] \quad (2.6.5.4)$$

Figure 23:



$$g(4,\{3,5\}) = \min[w(4,3) + g(3,\{5\})] \quad (2.6.4.3)$$
$$= \min[w(4,5) + g(5,\{3\})] \quad (2.6.4.5)$$

Figure 24:

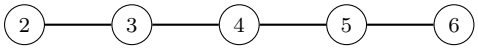


$$g(3,\{4,5\}) = \min[w(3,4) + g(4,\{5\})] \quad (2.6.3.4)$$
$$= \min[w(3,5) + g(5,\{4\})] \quad (2.6.3.5)$$

Equation 2.3.4.5.6 contains the distance function of $g(6,\phi)$, this means that all the stops given in set s are visited exactly once. ϕ represents the null set,indicating that stops left to visit are none. So now from stop 6 the bus moves back to the starting point, stop 2. Hence the value of $g(6,\phi)$ is equal to the distance from stop 6 to stop 1 which is 3200. Same logic is used for all the equation. The general formula for this is :

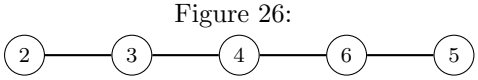
$$g(j,\phi) = w(j,i) \quad (2)$$

Figure 25:



$$g(5,\{6\}) = \min[w(5,6) + g(6,\phi)] \quad (2.3.4.5.6)$$

Figure 26:



$$g(6,\{5\}) = \min[w(6,5) + g(5,\phi)] \quad (2.3.4.6.5)$$

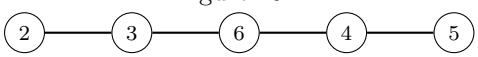
Figure 27:

$$g(4,\{6\}) = \min[w(4,6) + g(6,\phi)] \qquad (2.3.5.4.6)$$

Figure 28:

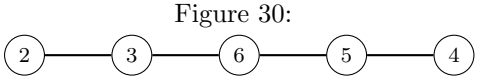
$$g(6,\{4\}) = \min[w(6,4) + g(4,\phi)] \qquad (2.3.5.6.4)$$

Figure 29:



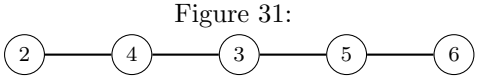
$$g(4,\{5\}) = \min[w(4,5) + g(5,\phi)] \qquad (2.3.6.4.5)$$

Figure 30:



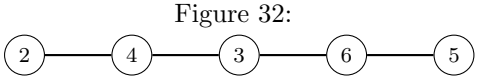
$$g(5,\{4\}) = \min[w(5,4) + g(4,\phi)] \qquad (2.3.6.5.4)$$

Figure 31:



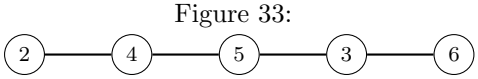
$$g(5,\{6\}) = \min[w(5,6) + g(6,\phi)] \qquad (2.4.3.5.6)$$

Figure 32:



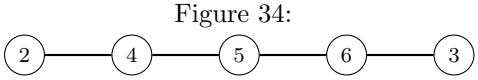
$$g(6,\{5\}) = \min[w(6,5) + g(5,\phi)] \qquad (2.4.3.6.5)$$

Figure 33:



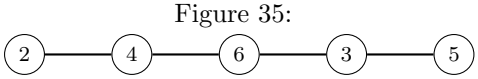
$$g(3,\{6\}) = \min[w(3,6) + g(6,\phi)] \qquad (2.4.5.3.6)$$

Figure 34:



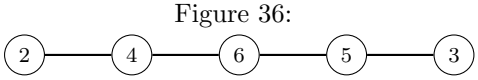
$$g(6,\{3\}) = \min[w(6,3) + g(3,\phi)] \qquad (2.4.5.6.3)$$

Figure 35:



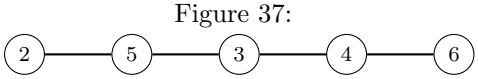
$$g(3,\{5\}) = \min[w(3,5) + g(5,\phi)] \qquad (2.4.6.3.5)$$

Figure 36:



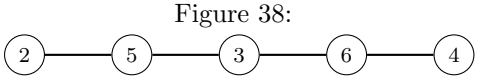
$$g(5,\{3\}) = \min[w(5,3) + g(3,\phi)] \qquad (2.4.6.5.3)$$

Figure 37:



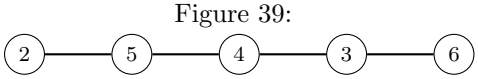
$$g(4,\{6\}) = \min[w(4,6) + g(6,\phi)] \qquad (2.5.3.4.6)$$

Figure 38:



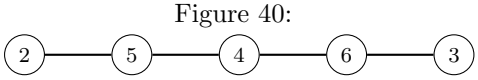
$$g(6,\{4\}) = \min[w(6,4) + g(4,\phi)] \qquad (2.5.3.6.4)$$

Figure 39:



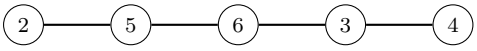
$$g(3,\{6\}) = \min[w(3,6) + g(6,\phi)] \qquad (2.5.4.3.6)$$

Figure 40:



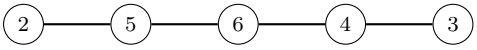
$$g(6,\{3\}) = \min[w(6,3) + g(3,\phi)] \qquad (2.5.4.6.3)$$

Figure 41:



$$g(3,\{4\}) = \min[w(3,4) + g(4,\phi)] \qquad (2.5.6.3.4)$$

Figure 42:



$$g(4,\{3\}) = \min[w(4,3) + g(3,\phi)] \qquad (2.5.6.4.3)$$

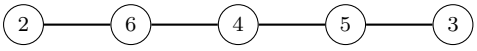
Figure 43:

$$g(3,\{4\}) = \min[w(3,4) + g(4,\phi)] \qquad (2.6.5.3.4)$$

Figure 44:

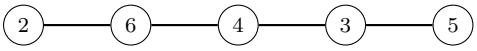
$$g(4,\{3\}) = \min[w(4,3) + g(3,\phi)] \qquad (2.6.5.4.3)$$

Figure 45:



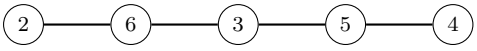
$$g(5,\{3\}) = \min[w(5,3) + g(3,\phi)] \qquad (2.6.4.5.3)$$

Figure 46:



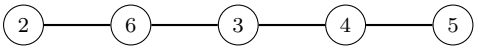
$$g(3,\{5\}) = \min[w(3,5) + g(5,\phi)] \qquad (2.6.4.3.5)$$

Figure 47:



$$g(5,\{4\}) = \min[w(5,4) + g(4,\phi)] \qquad (2.6.3.5.4)$$

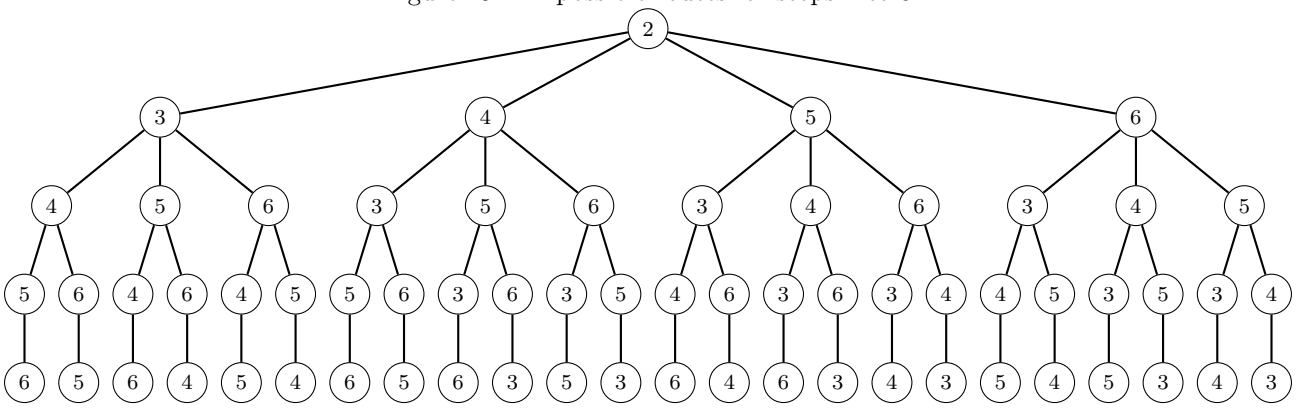
Figure 48:



$$g(4,\{5\}) = \min[w(4,5) + g(5,\phi)] \qquad (2.6.3.4.5)$$

All the possible routes in the bus can take between the nodes 2 to 6, starting from 2, are represented in the figure 49

Figure 49: All possible routes for stops 2 to 6

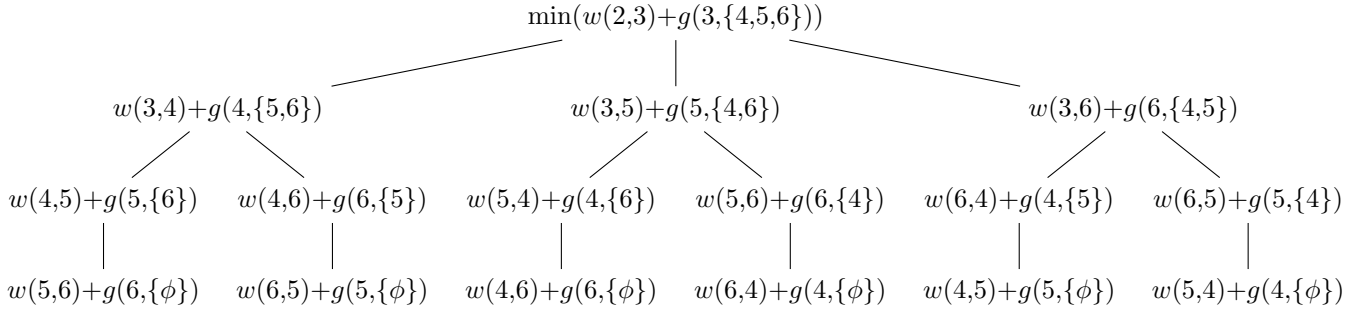


All the possible routes have been converted into equations. The next step is to reduce those equations down to a numerical value which allows the algorithm to compare the values and minimise the function.

5.30 Solving equation 2.3

The objective of solving equation 2.3 is to find the shortest route possible when the first 2 stops covered by the bus are fixed at stop 2 and stop 3 respectively. In order to do this, dynamic programming is performed by solving the equations derived in section 5.20 as shown in figure 50

Figure 50: Expansion of equation 2.3



Solving equation 2.3.4

$$w(5,6) + g(6,\phi) = 85 + 950 = 1035 \tag{3}$$

$$w(6,5) + g(5,\phi) = 85 + 1000 = 1085 \tag{4}$$

$$w(4,5) + g(5,6) = 400 + 1035 = 1435 \tag{5}$$

$$w(4,6) + g(6,5) = 450 + 1085 = 1535 \tag{6}$$

Value of g(4,5,6) is given by the equations 5 and 6. Since objective is to minimise, the one with lowest value is chosen, which is equation 5

$$w(3,4) + g(4,5,6) = 350 + 1435 = 1785 \tag{7}$$

Solving equation 2.3.5

$$w(4,6) + g(6,\phi) = 450 + 950 = 1400 \tag{8}$$

$$w(6,4) + g(4,\phi) = 450 + 750 = 1200 \tag{9}$$

$$w(5,4) + g(4,6) = 400 + 1400 = 1800 \tag{10}$$

$$w(5,6) + g(6,4) = 85 + 1200 = 1285 \tag{11}$$

Value of g(5,4,6) is given by the equations 10 and 11. Since objective is to minimise, the one with lowest value is chosen, which is equation 11.

$$w(3,5) + g(5,4,6) = 4750 + 1285 = 2035 \tag{12}$$

Solving equation 2.3.6

$$w(4,5) + g(5,\phi) = 400 + 1000 = 1400 \tag{13}$$

$$w(5,4) + g(4,\phi) = 400 + 750 = 1150 \tag{14}$$

$$w(6,4) + g(4,5) = 450 + 1400 = 1850 \tag{15}$$

$$w(6,5) + g(5,4) = 85 + 1150 = 1235 \tag{16}$$

Value of g(6,5,4) is given by the equations 15 and 16. Since objective is to minimise, the one with lowest value is chosen, which is equation 16

$$w(3,6) + g(6,4,5) = 700 + 1235 = 1935 \tag{17}$$

Value of g(3,{4,5,6}) is given by the equations 7, 12 and 17. Since objective is to minimise, the one with lowest value is chosen, which is equation 7.

$$w(2,3) + g(3,4,5,6) = 290 + 1785 = 2075 \tag{18}$$

5.40 Solving Equation 2.4

The objective of solving equation 2.4 is to find the shortest route possible when the first 2 stops covered by the bus are fixed at stop 2 and stop 4 respectively. In order to do this, dynamic programming is performed by solving the equations derived in section 5.20 as shown in figure 51

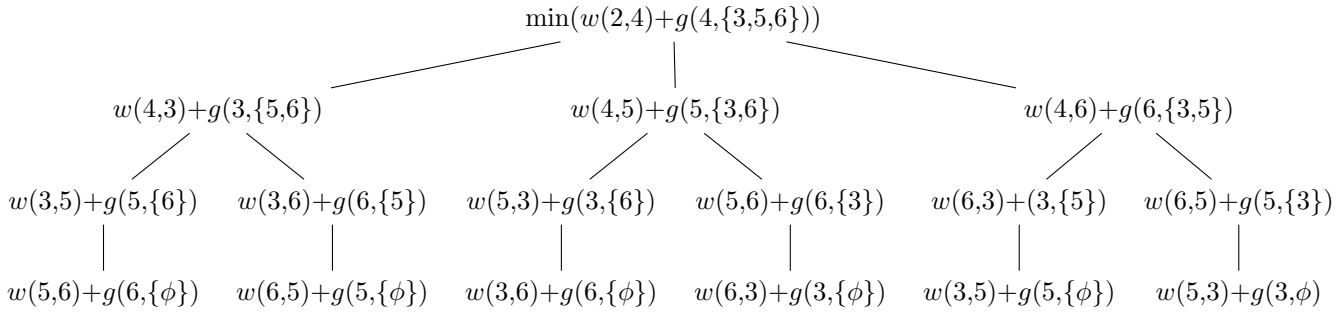


Figure 51: Expansion of Equation 2.4

Solving equation 2.4.3

$$w(5,6) + g(6,\phi) = 85 + 950 = 1035 \tag{19}$$

$$w(6,5) + g(5,\phi) = 85 + 1000 = 1085 \tag{20}$$

$$w(3,5) + g(5,6) = 750 + 1035 = 1785 \tag{21}$$

$$w(3,6) + g(6,5) = 700 + 1085 = 1785 \tag{22}$$

Value of g(6,5,4) is given by the equations 21 and 22. Since objective is to minimise, the one with lowest value is chosen, since both of the equations have same value, both the equations are applicable.

$$w(4,3) + g(3,5,6) = 350 + 1785 = 2135 \tag{23}$$

Solving equation 2.4.5

$$w(3,6) + g(6,\phi) = 700 + 950 = 1650 \tag{24}$$

$$w(6,3) + g(3,\phi) = 700 + 290 = 990 \tag{25}$$

$$w(5,3) + g(3,6) = 750 + 1650 = 2400 \tag{26}$$

$$w(5,6) + g(6,3) = 85 + 990 = 1075 \tag{27}$$

Value of g(5,3,6) is given by the equations 26 and 27. Since objective is to minimise, the one with lowest value is chosen,
which is equation 27.

$$w(4,5) + g(5,3,6) = 400 + 1075 = 1475 \tag{28}$$

Solving equation 2.4.6

$$w(3,5) + g(5,\phi) = 750 + 1000 = 1750 \tag{29}$$

$$w(5,3) + g(3,\phi) = 750 + 290 = 1040 \tag{30}$$

$$w(6,3) + (3,5) = 700 + 1750 = 2450 \tag{31}$$

$$w(6,5) + g(5,3) = 85 + 1040 = 1125 \tag{32}$$

Value of g(6,3,5) is given by the equations 31 and 32. Since objective is to minimise, the one with lowest value is chosen, which is equation 32.

$$w(4,6) + g(6,3,5) = 450 + 1125 = 1575 \tag{33}$$

Value of g(4,{3,5,6}) is given by the equations 23, 28 and 33. Since objective is to minimise, the one with lowest value is chosen, which is equation 28.

$$w(2,4) + g(4,3,5,6) = 750 + 1475 = 2225 \tag{34}$$

5.50 Solving equation 2.5

The objective of solving equation 2.5 is to find the shortest route possible when the first 2 stops covered by the bus are fixed at stop 2 and stop 5 respectively. In order to do this, dynamic programming is performed by solving the equations derived in section 5.20 as shown in figure 52

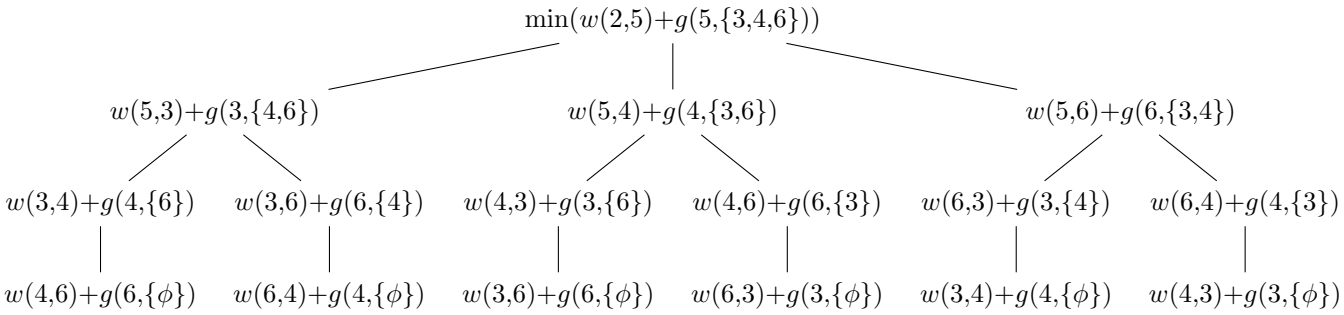


Figure 52: Expansion of equation 2.5

Solving equation 2.5.3

$$w(4,6) + g(6,\{\phi\}) = 450 + 950 = 1400 \tag{35}$$

$$w(6,4) + g(4,\{\phi\}) = 450 + 750 = 1200 \tag{36}$$

$$w(3,4) + g(4,\{6\}) = 350 + 1400 = 1750 \tag{37}$$

$$w(3,6) + g(6,\{4\}) = 700 + 1200 = 1900 \tag{38}$$

Value of g(3,4,6) is given by the equations 38 and 37. Since objective is to minimise, the one with lowest value is chosen, which is equation 37.

$$w(5,3) + g(3,\{4,6\}) = 750 + 1750 = 2500 \tag{39}$$

Solving equation 2.5.4

$$w(3,6) + g(6,\{\phi\}) = 700 + 950 = 1650 \tag{40}$$

$$w(6,3) + g(3,\{\phi\}) = 700 + 290 = 990 \tag{41}$$

$$w(4,3) + g(3,\{6\}) = 350 + 1650 = 2000 \tag{42}$$

$$w(4,6) + g(6,\{3\}) = 450 + 990 = 1440 \tag{43}$$

Value of g(4,3,6) is given by the equations 42 and 43. Since objective is to minimise, the one with lowest value is chosen, which is equation 43.

$$w(5,4) + g(4,3,6) = 400 + 1440 = 1840 \tag{44}$$

Solving equation2.5.6

$$w(3,4) + g(4,\{\phi\}) = 350 + 750 = 1100 \tag{45}$$

$$w(4,3) + g(3,\{\phi\}) = 350 + 290 = 640 \tag{46}$$

$$w(6,3) + g(3,\{4\}) = 700 + 1100 = 1800 \tag{47}$$

$$w(6,4) + g(4,\{3\}) = 450 + 640 = 1090 \tag{48}$$

Value of g(6,3,4) is given by the equations 47 and 48. Since objective is to minimise, the one with lowest value is chosen, which is equation 48.

$$w(5,6) + g(6,3,4) = 85 + 1090 = 1175 \tag{49}$$

Value of g(5,{3,4,6}) is given by the equations 39, 44 and 49. Since objective is to minimise, the one with lowest value is chosen, which is equation 49.

$$w(2,5) + g(5,3,4,6) = 1000 + 1175 = 2175 \tag{50}$$

5.60 Solving equation 2.6

The objective of solving equation 2.6 is to find the shortest route possible when the first 2 stops covered by the bus are fixed at stop 2 and stop 6 respectively. In order to do this, dynamic programming is performed by solving the equations derived in section 5.20 as shown in figure 53

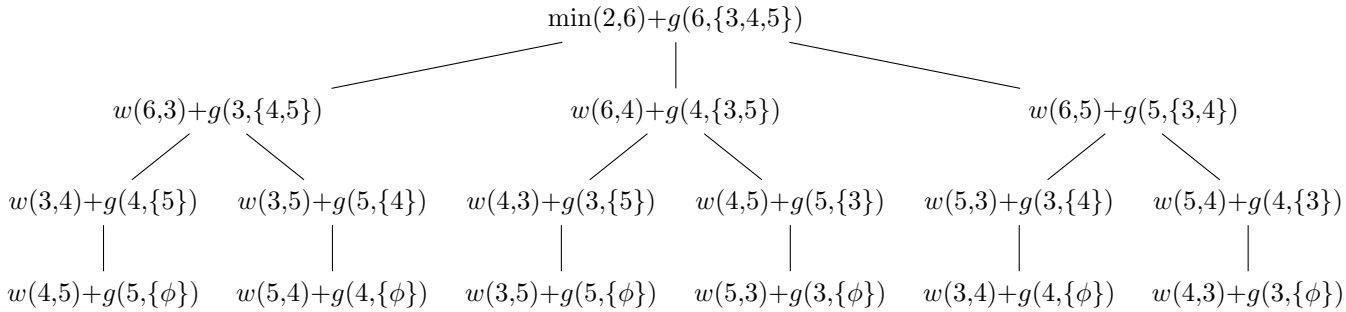


Figure 53: Expansion of equation 2.6

Solving equation 2.6.3

$$w(4,5) + g(5,\phi) = 400 + 1000 = 1400 \tag{51}$$

$$w(5,4) + g(4,\phi) = 400 + 750 = 1150 \tag{52}$$

$$w(3,4) + g(4,5) = 350 + 1400 = 1750 \tag{53}$$

$$w(3,5) + g(3,4,5) = 750 + 1150 = 1900 \tag{54}$$

Value of g(3,4,5) is given by the equations 53 and 54. Since objective is to minimise, the one with lowest value is chosen, which is equation 53.

$$w(6,3) + g(3,4,5) = 700 + 1750 = 2450 \tag{55}$$

Solving equation 2.6.4

$$w(3,5) + g(5,\{\phi\}) = 750 + 1000 = 1750 \tag{56}$$

$$w(5,3) + g(3,\{\phi\}) = 750 + 290 = 1040 \tag{57}$$

$$w(4,3) + g(3,\{5\}) = 350 + 1750 = 2100 \tag{58}$$

$$w(4,5) + g(5,\{3\}) = 400 + 1040 = 1440 \tag{59}$$

Value of g(4,3,5) is given by the equations 58 and 59. Since objective is to minimise, the one with lowest value is chosen, which is equation 59.

$$w(6,4) + g(4,\{3,5\}) = 450 + 1440 = 1890 \tag{60}$$

Solving equation 2.6.5

$$w(3,4) + g(4,\{\phi\}) = 350 + 750 = 1100 \tag{61}$$

$$w(4,3) + g(3,\{\phi\}) = 350 + 290 = 640 \tag{62}$$

$$w(5,3) + g(3,\{4\}) = 750 + 1100 = 1850 \tag{63}$$

$$w(5,4) + g(4,\{3\}) = 400 + 640 = 1040 \tag{64}$$

Value of g(5,3,4) is given by the equations 63 and 64. Since objective is to minimise, the one with lowest value is chosen, which is equation 64.

$$w(6,5) + g(5,3,4) = 85 + 1040 = 1125 \tag{65}$$

Value of g(6,{3,4,5}) is given by the equations 55, 60 and 65. Since objective is to minimise, the one with lowest value is chosen, which is equation 49.

$$w(2,6) + g(6,3,4,5) = 950 + 1125 = 2075 \tag{66}$$

5.70 Solving the problem

Value of g(2,{3,4,5,6}) is given by the equations 18, 34, 50 and 66. Since objective is to minimise, the one with lowest value is chosen, which are equation 18 and 66.

Since 18 and 66 both have the lowest and the same value, the dynamic programming gives us 2 optimum routes which are :

$$2 \longrightarrow 3 \longrightarrow 4 \longrightarrow 5 \longrightarrow 6 \longrightarrow 2$$

$$2 \longrightarrow 6 \longrightarrow 5 \longrightarrow 4 \longrightarrow 3 \longrightarrow 2$$

In both the cases the net distance travelled by the bus is **2075m**.

To find the optimum route for the cluster 3 which are stops 7 to 15, a code[6] was created which used dynamic programming algorithm to find the optimum route.

The optimum route given by it is

$$7 \longrightarrow 12 \longrightarrow 14 \longrightarrow 13 \longrightarrow 15 \longrightarrow 11 \longrightarrow 8 \longrightarrow 10 \longrightarrow 9 \longrightarrow 7$$

[6]Code was created by Preet Sheth. Computer Engineering Student at DA-IICT University in Gujarat, India

Now if we combine both the optimum routes and add stop 1 at the start and the end of the route we get the route :

$$1 \longrightarrow 2 \longrightarrow 3 \longrightarrow 4 \longrightarrow 5 \longrightarrow 6 \longrightarrow 7 \longrightarrow 12 \longrightarrow 14 \longrightarrow 13 \longrightarrow 15 \longrightarrow 11 \longrightarrow 8 \longrightarrow 10 \longrightarrow 9 \longrightarrow 1$$

$$2700 + 290 + 350 + 400 + 85 + 300 + 350 + 270 + 18 + 150 + 650 + 300 + 190 + 70 + 3400 = \mathbf{9523m}$$

The distance bus needs to travel during this route is : 9523m. The solution given be dynamic programming is represented on the map as shown in figure 54

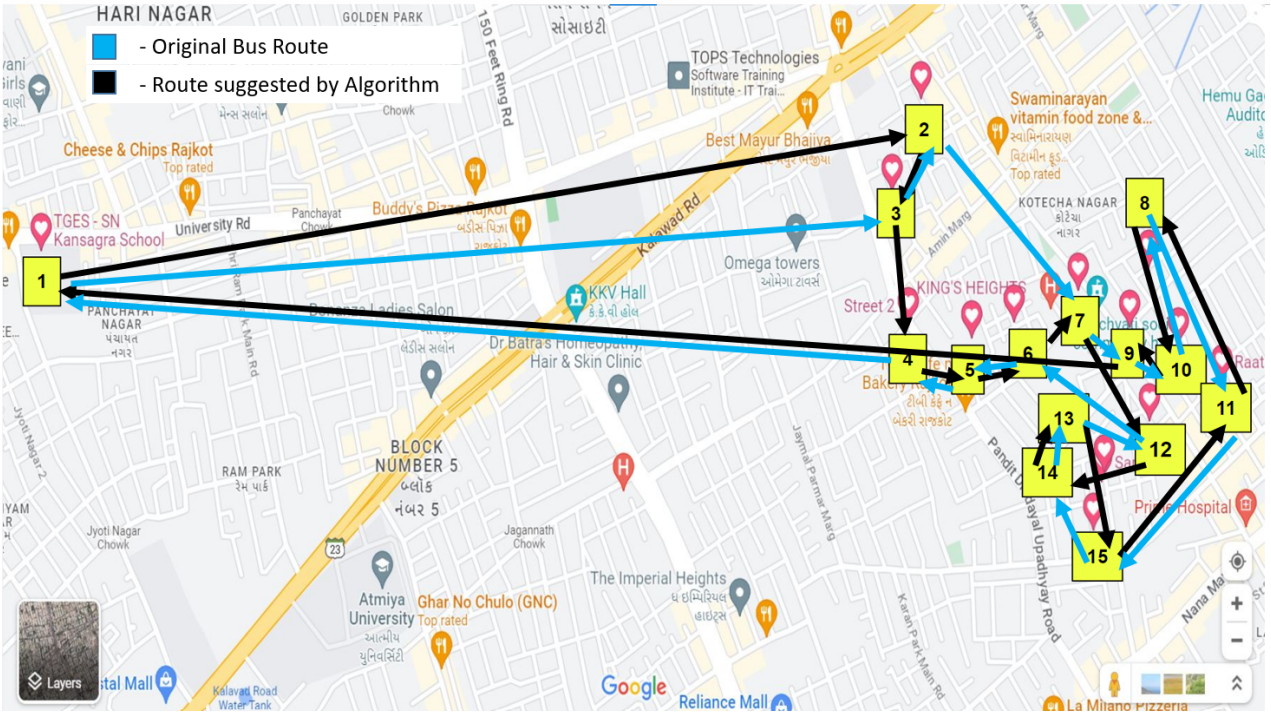


Figure 54: Dynamic programming result

The dynamic programming method to solve the problem is extremely tedious and requires a large amount of calculations. The algorithm is quite difficult to understand and has high time complexity, that is it requires large amount of time to be computed. However this was made easier by breaking down the problem into clusters. Moreover in figure 54 it is seen that the route is overlapping many stops too and hence causing the route to have a higher distance than the original route which had the distance of 8873 m.

6.0 Ant Colony Optimisation

Ant colony optimization is a heuristic method that does not guarantee an optimal solution, but it can find good solutions quickly. Ants leave pheromones on the ground as they walk[7] , and other ants follow these pheromone trails. When an ant has to choose between two paths, it is more likely to choose the path with more pheromones.

[7]https://www.nytimes.com/2019/01/22/science/ants-navigate-scent.html

This is how ants find the shortest route between a set of stops.

Ants leave pheromones on the ground as they walk. The more ants that walk on a path, the more pheromones are deposited. This means that the shortest path will have the most pheromones, and ants will be more likely to choose it. Over time, this will lead to all of the ants using the shortest path.

There are some conditions[8] which the ants follow while moving around the route :

- Ants have memory, they do not visit any of the stops twice in their journey
- Ants choose the closest stop when pheromone levels are the same.
- If the distance of 2 paths are same, the ants choose the path with greater pheromone level

The mathematical model used to represent the pheromone level on a path is given by the formula :

$$\Delta\tau_{i,j} = \left\{ \begin{array}{ll} \dfrac{1}{L_{i,j}}, & \text{k ant travels on the edge i to j} \\ 0, & \text{Otherwise} \end{array} \right\} \tag{67}$$

- $\tau^k_{i,j}$ shows the amount of pheromone deposited by ants on path from stop i to stop j
- Pheromone level is equal to 0 if ant does not travel on the edge.
- If ant travels on the path i to j then the pheromone level is $\dfrac{1}{L_k}$ where L_k is the length of path from i to j. Hence the shorter the path greater the amount of pheromone deposited.

After calculating the pheromone level on each path we need to calculate the probability of ant moving onto a path. The formula for the probability P of ant moving onto a path from stop i to stop j is given by the equation[9]:

$$P_{i,j} = \frac{(\tau_{i,j})^\alpha \cdot (\eta_{i,j})^\beta}{\Sigma((\tau_{i,j})^\alpha \cdot (\eta_{i,j})^\beta)}, \text{ Where } \eta_{i,j} = \frac{1}{L_{i,j}} \tag{68}$$

- $\tau_{i,j}$ represents the pheromone level on the path from stop i to j.
- $\eta_{i,j}$ represents the quality of path, it describes desire of the ant moving from stop i to stop j.
- The denominator has the sum of the pheromone level of all edges times the quality of all edges.
- The parameters α and β can be used to increase or decrease the impact of or on the decision-making process. For example, if we want to add a factor of road quality, we can regulate the pheromone level on a path based on the road quality using α and β. However, in this exploration, we only consider the distance, so α and β are both set to 1.

Equation 67 gives the level of pheromone deposited on each possible path. The pheromone level is inversely proportional to the distance, meaning that the shorter the distance, the higher the pheromone level. This is because more ants will travel through a shorter path in a given time, depositing more pheromone.

[8] https://youtu.be/783ZtAF4j5g

[9] https://mat.uab.cat/ alseda/MasterOpt/$ACO_Intro.pdf$

The formula of probability of ant moving on a particular node after the pheromone is deposited on all the possible nodes is given by the equation 68. This formula is a ratio of pheromone level on a particular node (i to j)to sum of total pheromone level on all the paths starting from i. Starting from stop 1, which next stop has the greatest probability can be checked from and that stop will be the next stop after stop 1. This process can be repeated until all the stops are covered to get the optimised route.

6.10 Applying ant colony optimisation

1. To simplify the calculations, the distances measured in meters are converted to Kilometers. Here is the new distance matrix.

Stop No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0.00	2.70	2.50	2.70	3.20	3.20	3.20	3.60	3.40	3.50	3.80	3.60	3.40	3.40	3.50
2	2.70	0.00	0.29	0.75	1.00	0.95	0.55	0.85	0.75	0.90	1.10	0.95	1.00	1.10	1.40
3	2.50	0.29	0.00	0.35	0.75	0.70	0.65	0.85	0.80	1.00	1.10	1.00	0.95	1.00	1.00
4	2.70	0.75	0.35	0.00	0.40	0.45	0.50	0.75	0.70	0.85	1.00	0.80	0.65	0.66	0.65
5	3.20	1.00	0.75	0.40	0.00	0.09	0.40	0.65	0.45	0.60	0.85	0.60	0.45	0.50	0.50
6	3.20	0.95	0.70	0.45	0.09	0.00	0.30	0.55	0.40	0.55	0.75	0.55	0.45	0.45	0.55
7	3.20	0.55	0.65	0.50	0.40	0.30	0.00	0.30	0.15	0.30	0.50	0.35	0.45	0.45	0.60
8	3.60	0.85	0.85	0.75	0.65	0.55	0.30	0.00	0.17	0.19	0.30	0.40	0.55	0.55	0.70
9	3.40	0.75	0.80	0.70	0.45	0.40	0.15	0.17	0.00	0.07	0.35	0.24	0.35	0.40	0.50
10	3.50	0.90	1.00	0.85	0.60	0.55	0.30	0.19	0.07	0.00	0.40	0.26	0.40	0.45	0.55
11	3.80	1.10	1.10	1.00	0.85	0.75	0.50	0.30	0.35	0.40	0.00	0.50	0.70	0.70	0.65
12	3.60	0.95	1.00	0.80	0.60	0.55	0.35	0.40	0.24	0.26	0.50	0.00	0.29	0.27	0.35
13	3.40	1.00	0.95	0.65	0.45	0.45	0.45	0.55	0.35	0.40	0.70	0.29	0.00	0.02	0.15
14	3.40	1.10	1.00	0.66	0.50	0.45	0.45	0.55	0.40	0.45	0.70	0.27	0.02	0.00	0.13
15	3.50	1.40	1.00	0.65	0.50	0.55	0.60	0.70	0.50	0.55	0.65	0.35	0.15	0.13	0.00

Table 11: Distance Matrix (Km)

2. Calculating the pheromone level on each path. Using equation 67.

   Sample calculation for journey from stop 2 to stop 3.

$$\begin{aligned}\tau_{2,3}^{k} &= \frac{1}{L_k} \\ &= \frac{1}{0.29} \\ &\approx 3.45\end{aligned}$$

Stop No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0.00	0.37	0.40	0.37	0.31	0.31	0.31	0.28	0.29	0.29	0.26	0.28	0.29	0.29	0.29
2	0.37	0.00	3.45	1.33	1.00	1.05	1.82	1.18	1.33	1.11	0.91	1.05	1.00	0.91	0.71
3	0.40	3.45	0.00	2.86	1.33	1.43	1.54	1.18	1.25	1.00	0.91	1.00	1.05	1.00	1.00
4	0.37	1.33	2.86	0.00	2.50	2.22	2.00	1.33	1.43	1.18	1.00	1.25	1.54	1.52	1.54
5	0.31	1.00	1.33	2.50	0.00	11.76	2.50	1.54	2.22	1.67	1.18	1.67	2.22	2.00	2.00
6	0.31	1.05	1.43	2.22	11.76	0.00	3.33	1.82	2.50	1.82	1.33	1.82	2.22	2.22	1.82
7	0.31	1.82	1.54	2.00	2.50	3.33	0.00	3.33	6.67	3.33	2.00	2.86	2.22	2.22	1.67
8	0.28	1.18	1.18	1.33	1.54	1.82	3.33	0.00	5.88	5.26	3.33	2.50	1.82	1.82	1.43
9	0.29	1.33	1.25	1.43	2.22	2.50	6.67	5.88	0.00	14.29	2.86	4.17	2.86	2.50	2.00
10	0.29	1.11	1.00	1.18	1.67	1.82	3.33	5.26	14.29	0.00	2.50	3.85	2.50	2.22	1.82
11	0.26	0.91	0.91	1.00	1.18	1.33	2.00	3.33	2.86	2.50	0.00	2.00	1.43	1.43	1.54
12	0.28	1.05	1.00	1.25	1.67	1.82	2.86	2.50	4.17	3.85	2.00	0.00	3.45	3.70	2.86
13	0.29	1.00	1.05	1.54	2.22	2.22	2.22	1.82	2.86	2.50	1.43	3.45	0.00	55.56	6.67
14	0.29	0.91	1.00	1.52	2.00	2.22	2.22	1.82	2.50	2.22	1.43	3.70	55.56	0.00	7.69
15	0.29	0.71	1.00	1.54	2.00	1.82	1.67	1.43	2.00	1.82	1.54	2.86	6.67	7.69	0.00

Table 12: Pheromone Level Matrix

3. Calculating $\tau_{i,j} \times \eta_{i,j}$. Below is the matrix for the same.

   Sample calculation for journey from stop 2 to stop 3.

$$(\tau_{2,3})^{\alpha} \cdot (\eta_{2,3})^{\beta} = (3.45)^{1} \cdot \left(\frac{1}{0.29}\right)^{1}$$

$$\approx 11.89$$

Stop No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0.00	0.14	0.16	0.14	0.10	0.10	0.10	0.08	0.09	0.08	0.07	0.08	0.09	0.09	0.08
2	0.14	0.00	11.89	1.78	1.00	1.11	3.31	1.38	1.78	1.23	0.83	1.11	1.00	0.83	0.51
3	0.16	11.89	0.00	8.16	1.78	2.04	2.37	1.38	1.56	1.00	0.83	1.00	1.11	1.00	1.00
4	0.14	1.78	8.16	0.00	6.25	4.94	4.00	1.78	2.04	1.38	1.00	1.56	2.37	2.30	2.37
5	0.10	1.00	1.78	6.25	0.00	138.41	6.25	2.37	4.94	2.78	1.38	2.78	4.94	4.00	4.00
6	0.10	1.11	2.04	4.94	138.41	0.00	11.11	3.31	6.25	3.31	1.78	3.31	4.94	4.94	3.31
7	0.10	3.31	2.37	4.00	6.25	11.11	0.00	11.11	44.44	11.11	4.00	8.16	4.94	4.94	2.78
8	0.08	1.38	1.38	1.78	2.37	3.31	11.11	0.00	34.60	27.70	11.11	6.25	3.31	3.31	2.04
9	0.09	1.78	1.56	2.04	4.94	6.25	44.44	34.60	0.00	204.08	8.16	17.36	8.16	6.25	4.00
10	0.08	1.23	1.00	1.38	2.78	3.31	11.11	27.70	204.08	0.00	6.25	14.79	6.25	4.94	3.31
11	0.07	0.83	0.83	1.00	1.38	1.78	4.00	11.11	8.16	6.25	0.00	4.00	2.04	2.04	2.37
12	0.08	1.11	1.00	1.56	2.78	3.31	8.16	6.25	17.36	14.79	4.00	0.00	11.89	13.72	8.16
13	0.09	1.00	1.11	2.37	4.94	4.94	4.94	3.31	8.16	6.25	2.04	11.89	0.00	3086.42	44.44
14	0.09	0.83	1.00	2.30	4.00	4.94	4.94	3.31	6.25	4.94	2.04	13.72	3086.42	0.00	59.17
15	0.08	0.51	1.00	2.37	4.00	3.31	2.78	2.04	4.00	3.31	2.37	8.16	44.44	59.17	0.00

Table 13: $\tau_{i,j} \cdot \eta_{i,j}$

4. Calculating the probability percentage of each path.

   Sample calculation is shown for path from stop 2 to 3.

$$
\begin{aligned}
P_{2,3} &= \frac{(\tau_{2,3})^{\alpha} \cdot (\eta_{2,3})^{\beta}}{\Sigma((\tau_{2,j})^{\alpha}\eta_{2,j})^{\beta}} \times 100 \\
&= \frac{11.89}{0.14 + 0 + 11.89 + 1.78 + ... + 0.08} \times 100 \\
&\approx 42.64
\end{aligned}
$$

   Here is the probability percentage matrix :

Stop No.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
1	0.00	9.99	11.65	9.99	7.11	7.11	7.11	5.62	6.30	5.94	5.04	5.62	6.30	6.30	5.94
2	0.49	0.00	42.64	6.37	3.59	3.97	11.85	4.96	6.37	4.43	2.96	3.97	3.59	2.96	1.83
3	0.45	33.70	0.00	23.14	5.04	5.78	6.71	3.92	4.43	2.83	2.34	2.83	3.14	2.83	2.83
4	0.34	4.44	20.38	0.00	15.60	12.33	9.98	4.44	5.09	3.45	2.50	3.90	5.91	5.73	5.91
5	0.05	0.55	0.98	3.45	0.00	76.48	3.45	1.31	2.73	1.53	0.76	1.53	2.73	2.21	2.21
6	0.05	0.59	1.08	2.62	73.30	0.00	5.88	1.75	3.31	1.75	0.94	1.75	2.62	2.62	1.75
7	0.08	2.79	2.00	3.37	5.27	9.37	0.00	9.37	37.47	9.37	3.37	6.88	4.16	4.16	2.34
8	0.07	1.26	1.26	1.62	2.16	3.01	10.13	0.00	31.54	25.25	10.13	5.70	3.01	3.01	1.86
9	0.03	0.52	0.45	0.59	1.44	1.82	12.93	10.07	0.00	59.37	2.37	5.05	2.37	1.82	1.16
10	0.29	4.43	3.59	4.96	9.96	11.85	39.84	99.33	731.82	0.00	22.41	53.05	22.41	17.71	11.85
11	0.02	0.29	0.29	0.35	0.48	0.62	1.39	3.86	2.83	2.17	0.00	1.39	0.71	0.71	0.82
12	0.17	2.42	2.18	3.41	6.06	7.21	17.80	13.63	37.86	32.26	8.72	0.00	25.93	29.91	17.80
13	0.09	1.06	1.18	2.51	5.24	5.24	5.24	3.51	8.67	6.64	2.17	12.63	0.00	3277.50	47.20
14	0.00	0.03	0.03	0.07	0.13	0.16	0.16	0.10	0.20	0.16	0.06	0.43	97.00	0.00	1.86
15	0.00	0.02	0.03	0.07	0.13	0.10	0.09	0.06	0.13	0.10	0.07	0.26	1.39	1.85	0.00

Table 14: Probability Percentage Matrix

5. Start at stop 1 and move to the stop with the highest probability. Then, move to the stop with the highest probability from that stop, and so on. This process will eventually lead to the optimum route. The optimum route obtained is shown in the table given :

Stop No.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1			11.65												
2							11.85								
3		33.70													
4				0.00											
5						76.48									
6				2.62											
7									37.47						
8											10.13				
9										59.37					
10								99.33							
11												1.39			
12														29.91	
13															47.20
14													97.00		
15					0.13										

Table 15: Ant Colonisation - Optimum Route

Therefore the optimum route given by ant colonisation method is :

$$1 \longrightarrow 3 \longrightarrow 2 \longrightarrow 7 \longrightarrow 9 \longrightarrow 10 \longrightarrow 8 \longrightarrow 11 \longrightarrow 12 \longrightarrow 14 \longrightarrow 13 \longrightarrow 15 \longrightarrow 5 \longrightarrow 6 \longrightarrow 4 \longrightarrow 1$$

$$2500 + 290 + 550 + 150 + 70 + 190 + 300 + 500 + 270 + 18 + 150 + 500 + 85 + 450 + 2700 = 8723m$$

The total distance of the journey is 8723m.

The result given by ant colonisation approach is represented on the map shown in figure 55

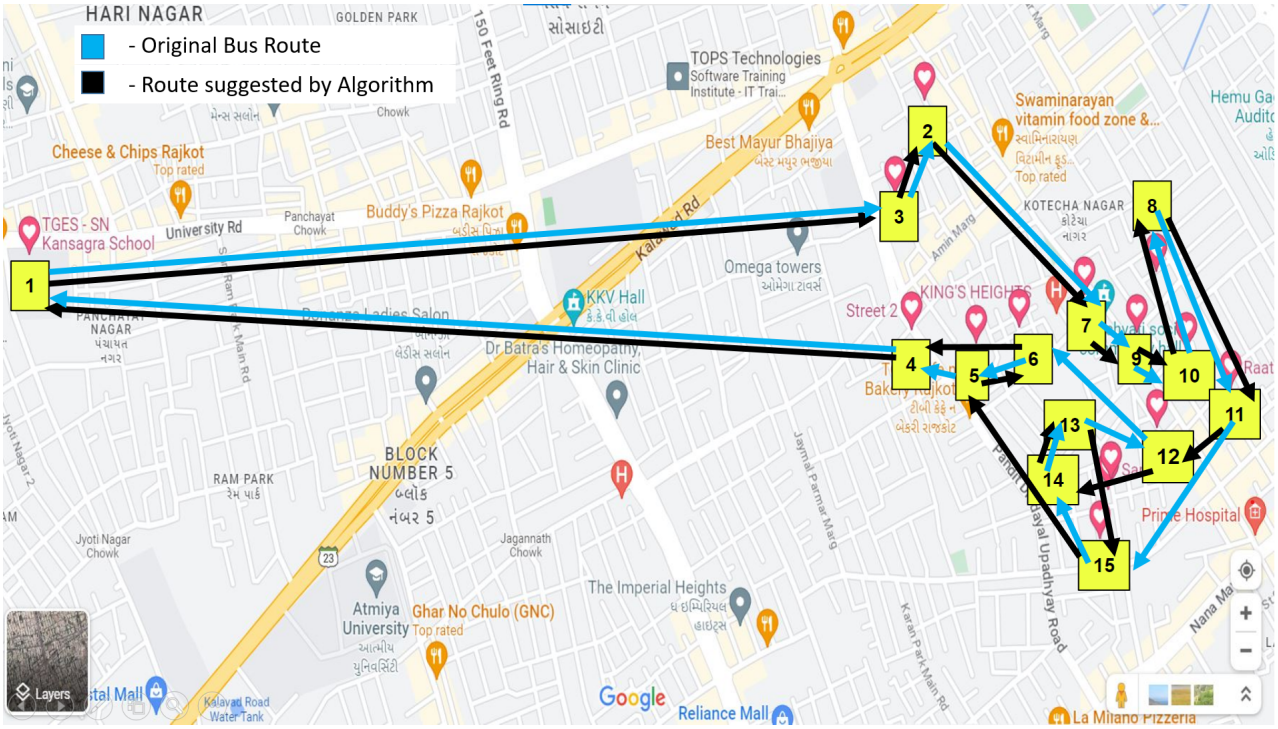


Figure 55: Ant Colonisation Result

The ant colony optimization algorithm is moderately complicated and involves a moderate amount of calculations. Ant colony optimisation is an heuristic method which provides a relatively 'good' result with reasonable level of calculations and complications. The route provided by the ant colony optimization is same as the original route up to stop 11, after it both the routes are different. The algorithm has provided a route which is shorter than the original route by 154m, the difference is not very significant, but the objective of minimisation is met using the algorithm. Hence the route provided by the ant colony optimization has the least distance out of all the 3 methods.

7.0 Evaluation

Limitations	Effect	Possible solution
Algorithms applied to only one bus	Not possible to justify the efficiency of an algorithm just based on one route.	Need to apply algorithms of other buses with different routes to decide which algorithm is the best
Dynamic programming applied to small clusters and not the route as a whole.	Reduced the power of dynamic programming. No algorithm used in connecting the clusters.	Use computers with higher processing power which can apply dynamic programming on routes with higher number of stops.
All the algorithms were applied separately	Each algorithm had their own limitations. Merging them could lead to suppress the limitations and exploit advantages of both.	A possible solution could be apply dynamic programming in small clusters and connect the clusters using nearest neighbour or ant colonisation

Table 16: Limitations and Possible Solutions

8.0 Conclusion

Ant colony optimization provides the best route at 8723 meters, followed by nearest neighbor at 8872 meters, and dynamic programming at 9523 meters.

The exploration aimed to optimize the school bus route by finding the shortest possible route for a school bus covering all the bus stops. Nearest neighbor algorithm and ant colony optimization were successfully able to provide relatively optimum routes, while dynamic programming gave relatively long routes. The potential of dynamic programming has not been reached, possibly due to the large number of stops in the problem. Further testing of all 3 algorithms on numerous bus routes is needed to determine the best method, however the objective of the exploration was to find a shortest route, which is achieved to some extent as it has found a shorter route that the original route of the bus.

9.0 Further Extension

The exploration focused on minimizing the distance of the route. However, further research can be conducted to minimize the time taken. Other factors that may affect the time taken include road quality, traffic level, time of day, and waiting time per stop. Optimizing the route by road quality can be done using adapted ant colony optimization[10], where ants deposit pheromone trails on edges based on distance and road quality factors. Genetic algorithms[11] include the factor of traffic level.

[10] https://ieeexplore.ieee.org/document/7231524/

[11] https://in.mathworks.com/help/gads/what-is-the-genetic-algorithm.html

10.0 Bibliography

- "Travelling Salesman Problem Using Dynamic Programming." GeeksforGeeks, 19 Apr. 2023, www.geeksforgeeks.org/travelling-salesman-problem-using-dynamic-programming/.
- YouTube, 4 Apr. 2021, https://youtu.be/3QiSyc7KyC4. Accessed 15 Mar. 2023.
- YouTube, 4 June 2020, https://youtu.be/FllcPjvztTI. Accessed 15 Mar. 2023.
- Yue, Longwang, and Hanning Chen. "Unmanned Vehicle Path Planning Using a Novel Ant Colony Algorithm - Eurasip Journal on Wireless Communications and Networking." SpringerOpen, 28 May 2019, jwcn-eurasipjournals.springeropen.com/articles/10.1186/s13638-019-1474-5.
- Tables Generator, www.tablesgenerator.com/. Accessed summer 2023.
- YouTube, 4 Oct. 2018, https://youtu.be/783ZtAF4j5g. Accessed 24 Apr. 2023.
- YouTube, 2 Apr. 2018, https://youtu.be/Q4zHb-Swzro. Accessed 14 Mar. 2023.
- Ant Colonies for the Traveling Salesman Problem - SUPSI, people.idsia.ch/ luca/acs-bio97.pdf. Accessed 15 June 2023.
- "Clustering Algorithms - k-Means Algorithm." Tutorialspoint, www.tutorialspoint.com/machine_learning_with_python/clustering_algorithms_k_means_algorithm.htm. Accessed 3 May 2023.
- Lippman, David. "Mathematics for the Liberal Arts." Lumen, courses.lumenlearning.com/waymakermath4libarts/chapter/shortest-path/. Accessed 1 May. 2023.
- University, Haifeng Huang Beijing Jiaotong, et al. "Data-Driven Prediction of One-Way Bus Running Time: An Integrated Model: Proceedings of the 5th International Conference on Management Engineering, Software Engineering and Service Sciences." ACM Other Conferences, 1 Jan. 2021, dl.acm.org/doi/abs/10.1145/3459012.3459037.
-
- YouTube, 13 Mar. 2019, https://youtu.be/hh-uFQ-MGfw. Accessed 14 Mar. 2023.
- Regression Analysis for Transport Trip Generation Evaluation - Researchgate, www.researchgate.net/publication/272261087_Regression_Analysis_for_Transport_Trip_Generation_Evaluation. Accessed 1 Aug. 2023.
- Wagner, Harvey M. Principles of Operations Research: With Applications to Managerial Decisions. Prentice/Hall International, 1972.
- Google Maps, maps.google.com/. Accessed summer 2023.
- "Bus Routes —— Graph Theory." YouTube, 16 July 2021, youtu.be/WhuiqhMXhxM.

- Regression Analysis for Transport Trip Generation Evaluation - Researchgate, www.researchgate.net /publication/272261087_Regression_Analysis_for_Transport_Trip_Generation_Evaluation. Accessed 3 Aug. 2023. Google Sheets: Online Spreadsheet Editor — Google Workspace, www.google.com/sheets /about/. Accessed summer 2023.

A Appendix

1.10 Iterations for k-means clustering

All the iterations done during the calculation of k-means clustering are presented here.

C1		C2		C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.28815022	70.78148298	22.28659521	70.78433731	22.28498171	70.78434266

Table 17: Iteration 1

Stops	Co-ordinates		Distances			Cluster
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0.021233922	0.024143369	0.02431115	C1
2	22.29037206	70.78024775	0.002542119	0.005566785	0.006769352	C1
3	22.28906167	70.77968985	0.002011484	0.0052614	0.006188274	C1
4	22.28691738	70.77999026	0.001936002	0.004358973	0.004763428	C1
5	22.28648058	70.78153521	0.001670457	0.002804441	0.003182513	C1
6	22.28635036	70.78251657	0.002075527	0.001837123	0.002282058	C2
7	22.28703269	70.78407489	0.002822568	0.00051015	0.002068381	C2
8	22.28743179	70.7855219	0.004102325	0.001450223	0.002719095	C2
9	22.28632461	70.78504884	0.004006025	0.000761254	0.001517253	C2
10	22.28598988	70.78617584	0.005166241	0.001935624	0.002092115	C2
11	22.28535904	70.78716371	0.006329405	0.003084908	0.002846167	C3
12	22.28481832	70.78557756	0.005278936	0.002166926	0.001245661	C3
13	22.28395573	70.78460361	0.005228008	0.002652876	0.00105865	C3
14	22.28367249	70.78447839	0.005387254	0.002926118	0.00131624	C3
15	22.28297727	70.78432534	0.005902408	0.003617958	0.002004522	C3

Table 18: k-means iteration - 1

AVG C1		AVG C2		AVG C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.2882119	70.77634245	22.28662586	70.78466761	22.28415657	70.78522972

Table 19: Iteration - 2

Stops	**Co-ordinates**		**Distances**			**Cluster**
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0.014800938	0.023951611	0.025310103	C1
2	22.29037206	70.78024775	0.005482695	0.005422987	0.007965699	C2
3	22.28906167	70.77968985	0.004664259	0.005089689	0.00739934	C1
4	22.28691738	70.77999026	0.005238947	0.004167266	0.005922337	C2
5	22.28648058	70.78153521	0.006842215	0.002613142	0.004364681	C2
6	22.28635036	70.78251657	0.007816619	0.001648241	0.003489109	C2
7	22.28703269	70.78407489	0.009173382	0.000436792	0.003099307	C2
8	22.28743179	70.7855219	0.010547616	0.001607327	0.003288228	C2
9	22.28632461	70.78504884	0.010270082	0.000944829	0.002175573	C2
10	22.28598988	70.78617584	0.011443789	0.002120467	0.00206305	C3
11	22.28535904	70.78716371	0.012556938	0.003263954	0.002277332	C3
12	22.28481832	70.78557756	0.011206773	0.002287128	0.000747597	C3
13	22.28395573	70.78460361	0.010647431	0.002685279	0.000657534	C3
14	22.28367249	70.78447839	0.01066423	0.002948009	0.000893774	C3
15	22.28297727	70.78432534	0.010874669	0.00362884	0.001486156	C3

Table 20: k-means iteration - 2

AVG C1		AVG C2		AVG C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.28864472	70.77504426	22.28660165	70.78414554	22.28415657	70.78522972

Table 21: Iteration - 3

Stops	**Co-ordinates**		**Distances**			**Cluster**
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0.014800938	0.023951611	0.025310103	C1
2	22.29037206	70.78024775	0.005482695	0.005422987	0.007965699	C2
3	22.28906167	70.77968985	0.004664259	0.005089689	0.00739934	C1
4	22.28691738	70.77999026	0.005238947	0.004167266	0.005922337	C2
5	22.28648058	70.78153521	0.006842215	0.002613142	0.004364681	C2
6	22.28635036	70.78251657	0.007816619	0.001648241	0.003489109	C2
7	22.28703269	70.78407489	0.009173382	0.000436792	0.003099307	C2
8	22.28743179	70.7855219	0.010547616	0.001607327	0.003288228	C2
9	22.28632461	70.78504884	0.010270082	0.000944829	0.002175573	C2
10	22.28598988	70.78617584	0.011443789	0.002120467	0.00206305	C3
11	22.28535904	70.78716371	0.012556938	0.003263954	0.002277332	C3
12	22.28481832	70.78557756	0.011206773	0.002287128	0.000747597	C3
13	22.28395573	70.78460361	0.010647431	0.002685279	0.000657534	C3
14	22.28367249	70.78447839	0.01066423	0.002948009	0.000893774	C3
15	22.28297727	70.78432534	0.010874669	0.00362884	0.001486156	C3

Table 22: k-means iteration - 3

AVG C1		AVG C2		AVG C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.28864473	70.76996952	22.28727278	70.78270506	22.28446212	70.78538741

Table 23: Iteration - 4

Stops	**Co-ordinates**		**Distances**			**Cluster**
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0.009729263	0.022476161	0.02541869	C1
2	22.29037206	70.78024775	0.010422359	0.003955238	0.007832207	C2
3	22.28906167	70.77968985	0.009729263	0.003505944	0.007322436	C2
4	22.28691738	70.77999026	0.010168521	0.002737969	0.005929381	C2
5	22.28648058	70.78153521	0.011766419	0.001412848	0.004348978	C2
6	22.28635036	70.78251657	0.0127551	0.000941484	0.003436151	C2
7	22.28703269	70.78407489	0.014197187	0.001390713	0.002886265	C2
8	22.28743179	70.7855219	0.015599609	0.002821329	0.002972713	C2
9	22.28632461	70.78504884	0.015256764	0.002528309	0.001893011	C3
10	22.28598988	70.78617584	0.016422334	0.003700293	0.001719208	C3
11	22.28535904	70.78716371	0.017505306	0.004852004	0.001989901	C3
12	22.28481832	70.78557756	0.016070229	0.003778313	0.000403773	C3
13	22.28395573	70.78460361	0.015366951	0.003821952	0.000933151	C3
14	22.28367249	70.78447839	0.015337219	0.004013324	0.00120409	C3
15	22.28297727	70.78432534	0.015434038	0.00459094	0.00182559	C3

Table 24: k-means iteration - 4

AVG C1		AVG C2		AVG C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.28822779	70.7602492	22.28766379	70.78193949	22.28472819	70.78533904

Table 25: Iteration - 5

Stops	**Co-ordinates**		**Distances**			**Cluster**
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0	0.021697624	0.025332734	C1
2	22.29037206	70.78024775	0.020113177	0.003193228	0.007600955	C2
3	22.28906167	70.77968985	0.019458527	0.002648578	0.007119864	C2
4	22.28691738	70.77999026	0.019784502	0.002087256	0.005779452	C2
5	22.28648058	70.78153521	0.021357598	0.001250374	0.004188079	C2
6	22.28635036	70.78251657	0.02234638	0.001434618	0.00325542	C2
7	22.28703269	70.78407489	0.023855649	0.002226709	0.002628458	C2
8	22.28743179	70.7855219	0.025285239	0.003589919	0.002709777	C3
9	22.28632461	70.78504884	0.024872565	0.003385481	0.001622581	C3
10	22.28598988	70.78617584	0.026023051	0.00455507	0.001513967	C3
11	22.28535904	70.78716371	0.027066964	0.00571002	0.001930642	C3
12	22.28481832	70.78557756	0.025556807	0.004618687	0.000254978	C3
13	22.28395573	70.78460361	0.024726258	0.004565878	0.001066562	C3
14	22.28367249	70.78447839	0.024653688	0.004730377	0.001362067	C3
15	22.28297727	70.78432534	0.024642006	0.005258874	0.002023197	C3

Table 26: k-means iteration - 5

AVG C1		AVG C2		AVG C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.28822779	70.7602492	22.28770246	70.78134242	22.28506614	70.7853619

Table 27: Iteration - 6

Stops	**Co-ordinates**		**Distances**			**Cluster**
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0	0.021099764	0.025310942	C1
2	22.29037206	70.78024775	0.020113177	0.002885322	0.007369349	C2
3	22.28906167	70.77968985	0.019458527	0.002139735	0.006938044	C2
4	22.28691738	70.77999026	0.019784502	0.001563551	0.005681694	C2
5	22.28648058	70.78153521	0.021357598	0.001236994	0.00407973	C2
6	22.28635036	70.78251657	0.02234638	0.001790755	0.003121714	C2
7	22.28703269	70.78407489	0.023855649	0.002813359	0.002350257	C3
8	22.28743179	70.7855219	0.025285239	0.004188239	0.002371054	C3
9	22.28632461	70.78504884	0.024872565	0.003954242	0.001296823	C3
10	22.28598988	70.78617584	0.026023051	0.005127855	0.001231178	C3
11	22.28535904	70.78716371	0.027066964	0.006275268	0.00182546	C3
12	22.28481832	70.78557756	0.025556807	0.005123931	0.000328522	C3
13	22.28395573	70.78460361	0.024726258	0.004967224	0.001344624	C3
14	22.28367249	70.78447839	0.024653688	0.005106359	0.001650106	C3
15	22.28297727	70.78432534	0.024642006	0.005587952	0.002331919	C3

Table 28: k-means iteration - 6

AVG C1		AVG C2		AVG C3	
X	**Y**	**X**	**Y**	**X**	**Y**
22.28822779	70.7602492	22.28783641	70.78079593	22.28528465	70.7852189

Table 29: Iteration - 7

Stops	**Co-ordinates**		**Distances**			**Cluster**
	X	**Y**	**C1**	**C2**	**C3**	
1	22.28822779	70.7602492	0	0.020550456	0.025142554	C1
2	22.29037206	70.78024775	0.020113177	0.002594227	0.007112953	C2
3	22.28906167	70.77968985	0.019458527	0.001650661	0.006695994	C2
4	22.28691738	70.77999026	0.019784502	0.001222177	0.005477639	C2
5	22.28648058	70.78153521	0.021357598	0.001544286	0.003872961	C2
6	22.28635036	70.78251657	0.02234638	0.002273538	0.002904877	C2
7	22.28703269	70.78407489	0.023855649	0.003376031	0.002089117	C3
8	22.28743179	70.7855219	0.025285239	0.004743268	0.002168419	C3
9	22.28632461	70.78504884	0.024872565	0.004513629	0.001053776	C3
10	22.28598988	70.78617584	0.026023051	0.005687986	0.001188738	C3
11	22.28535904	70.78716371	0.027066964	0.006832716	0.001946231	C3
12	22.28481832	70.78557756	0.025556807	0.005654459	0.000588304	C3
13	22.28395573	70.78460361	0.024726258	0.005436739	0.001464445	C3
14	22.28367249	70.78447839	0.024653688	0.005558663	0.001774091	C3
15	22.28297727	70.78432534	0.024642006	0.006005665	0.002474357	C3

Table 30: k-means iteration - 7

1.20 Code used for Dynamic Programming

```
#include <bits/stdc++.h>
using namespace std;

typedef long long int ll;
#define f(i_itr,a,n) for(ll i_itr=a; i_itr<n; i_itr++)



const ll max_nodes = 18;
const ll inf=1e18;

ll dp[max_nodes][1 ll << max_nodes];
ll parent[max_nodes][1 ll << max_nodes];

const ll source_node=0;

ll n;

vector<vector<ll>> adjacency_matrix = {
        {0, 10, 15, 20},
        {10, 0, 25, 25},
        {15, 25, 0, 30},
        {20, 25, 30, 0},
};

int main()
{
        n=adjacency_matrix.size();

        ///initializing all values
        f(mask,0,1 ll<<n) f(i,0,n) dp[i][mask]=inf,parent[i][mask]=-1;
        dp[0][0]=0;

        f(mask,1,1 ll<<n)
{
f(i,0,n)
{
```

```
if((mask>>i) & 1ll)
{
ll remaining_mask = mask^(1ll<<i);
f(j,0,n)
{
if(dp[i][mask] > (dp[j] [remaining_mask] + adjacency_matrix[j][i]))
{
dp[i][mask] = (dp[j][remaining_mask] + adjacency_matrix[j][i]);
parent[i][mask]=j;
}
}
}
//                        cout << dp[i][mask] << " ";
}
//                cout << "\n";
}

vector<ll> path;
path.push_back(0);
ll current_mask=(1ll<<n) -1;
ll counter=0;
while(current_mask)
{
ll next_=parent[path.back()][current_mask];
current_mask^=(1ll<<path.back());
path.push_back(next_);
}

reverse(path.begin(),path.end());

cout << "Optimal Path Length : " << dp[0][(1ll<<n) - 1] << "\n";

cout << "Optimal Path : ";
for(ll node : path )
cout << node << ' ';
cout << "\n";
```

```
return 0;
}

}
```