Pure and Applied Mathematics Journal
2015; 4(1): 9-23
Published online January 29, 2015 (http://www.sciencepublishinggroup.com/j/pamj)
doi: 10.11648/j.pamj.20150401.12
ISSN: 2326-9790 (Print); ISSN: 2326-9812 (Online)

Applying minimum travel cost approach on 43–nodes travelling salesman problem

Mohamed Eleiche

Geo Tiba Systems, Cairo, Egypt

Email address:
mohamed.eleiche@gmail.com

To cite this article:
Mohamed Eleiche. Applying Minimum Travel Cost Approach on 43–Nodes Travelling Salesman Problem. *Pure and Applied Mathematics Journal.* Vol. 4, No. 1, 2015, pp. 9-23. doi: 10.11648/j.pamj.20150401.12

Abstract: The minimum travel cost is a new approach to solve the Travelling Salesman Problem (TSP). The TSP library website (TSPLIB) provides several TSP problems with their best knownsolutions as a means to test any proposed algorithm. The present paper successfully applies the minimum travel cost algorithmto the 43 nodes P43problem which has the value of 5620 for its best knownsolution.This paper provides the details of the solution for value of 5621.

Keywords: Traveling Salesman Problem, TSP, Minimum Travel Cost Approach, TSPLIB, TSP 43-nodes

1. Introduction

The Travelling Salesman Problem (TSP) is defined as a set of nodes that represent a number N of cities, where the distance (cost) between each two nodes is known, and it is required the tour with least cost that starts from one node and visits all other nodes and returns back to the start node, in condition that each node is visited only once. TSP is mathematically presented as a full graph with number of nodes N. TSP is a prototype of hard combinatorial optimization problem where the possible solutions are (N-1)! and is considered NP-hard and NP-complete. The new approach of minimum travel cost provides convergent solution for the TSP problem (Eleiche and Markus 2010).

The TSPLIB website (https://www.tsp.gatech.edu/problem/index.html) provides sample TSP problems with known solutions in order to test the validity of proposed solutions for this interesting problem. This paper addresses the P43 problem which is a symmetrical graph that has a value of 5620 for the least cost tour visiting all nodes. The P43 problem was selected to be solved using the new algorithm as it is the smallest problem in size after the previously solved problem with 17 nodes.

2. Solution for P43Based on Minimum Travel Cost

The P43 problem is composed from 43 nodes, and it is an asymmetrical network where $C_{ij} \neq C_{ji}$. The input data of the problem as downloaded from the website is shown in Table 1.

Table 1. *Cost of edges (Origin-Destination Matrix) (diagonal valuedv=99999)*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	dv	26	26	26	10	60	60	60	68	68	68	68	92	92
2	36	dv	0	0	36	6	6	6	10	10	10	10	12	12
3	36	0	dv	0	36	6	6	6	10	10	10	10	12	12
4	36	0	0	dv	36	6	6	6	10	10	10	10	12	12
5	10	26	26	26	dv	26	26	26	30	30	30	30	32	32
6	74	6	6	6	36	dv	0	0	4	4	4	4	6	6
7	74	6	6	6	36	0	dv	0	4	4	4	4	6	6
8	74	6	6	6	36	0	0	dv	4	4	4	4	6	6
9	82	10	10	10	40	4	4	4	dv	0	0	0	12	12
10	82	10	10	10	40	4	4	4	0	dv	0	0	12	12
11	82	10	10	10	40	4	4	4	0	0	dv	0	12	12
12	82	10	10	10	40	4	4	4	0	0	0	dv	12	12

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
13	106	12	12	12	42	6	6	6	6	6	6	6	dv	0
14	106	12	12	12	42	6	6	6	6	6	6	6	0	dv
15	106	12	12	12	42	6	6	6	6	6	6	6	0	0
16	130	16	16	16	46	12	12	12	16	16	16	16	12	12
17	130	16	16	16	46	12	12	12	16	16	16	16	12	12
18	130	16	16	16	46	12	12	12	16	16	16	16	12	12
19	178	96	96	96	126	24	24	24	24	24	24	24	24	24
20	178	96	96	96	126	24	24	24	24	24	24	24	24	24
21	178	96	96	96	126	24	24	24	24	24	24	24	24	24
22	70	46	46	46	76	56	56	56	56	56	56	56	76	76
23	46	22	22	22	56	56	56	56	64	64	64	64	88	88
24	46	22	22	22	56	56	56	56	64	64	64	64	88	88
25	52	16	16	16	52	22	22	22	26	26	26	26	28	28
26	56	22	22	22	46	22	22	22	26	26	26	26	28	28
27	134	30	30	30	60	26	26	26	30	30	30	30	24	24
28	37	1	1	1	37	7	7	7	11	11	11	11	13	13
29	37	1	1	1	37	7	7	7	11	11	11	11	13	13
30	75	7	7	7	37	1	1	1	5	5	5	5	7	7
31	75	7	7	7	37	1	1	1	5	5	5	5	7	7
32	83	11	11	11	41	5	5	5	1	1	1	1	13	13
33	83	11	11	11	41	5	5	5	1	1	1	1	13	13
34	107	13	13	13	43	7	7	7	7	7	7	7	1	1
35	107	13	13	13	43	7	7	7	7	7	7	7	1	1
36	31	7	7	7	41	41	41	41	49	49	49	49	73	73
37	37	1	1	1	37	7	7	7	11	11	11	11	13	13
38	41	7	7	7	31	7	7	7	11	11	11	11	13	13
39	5048	5014	5014	5014	5038	5014	5014	5014	5018	5018	5018	5018	5020	5020
40	5048	5014	5014	5014	5038	5014	5014	5014	5018	5018	5018	5018	5020	5020
41	5126	5022	5022	5022	5052	5018	5018	5018	5022	5022	5022	5022	5016	5016
42	5126	5022	5022	5022	5052	5018	5018	5018	5022	5022	5022	5022	5016	5016
43	5126	5022	5022	5022	5052	5018	5018	5018	5022	5022	5022	5022	5016	5016

Table 1. *Continued*

0	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	92	116	116	116	164	164	164	84	36	36	42	46	120	27
2	12	16	16	16	96	96	96	64	22	22	16	22	30	1
3	12	16	16	16	96	96	96	64	22	22	16	22	30	1
4	12	16	16	16	96	96	96	64	22	22	16	22	30	1
5	32	36	36	36	116	116	116	84	46	46	42	36	50	27
6	6	12	12	12	24	24	24	112	60	60	22	22	26	7
7	6	12	12	12	24	24	24	112	60	60	22	22	26	7
8	6	12	12	12	24	24	24	112	60	60	22	22	26	7
9	12	16	16	16	24	24	24	112	68	68	26	26	30	11
10	12	16	16	16	24	24	24	112	68	68	26	26	30	11
11	12	16	16	16	24	24	24	112	68	68	26	26	30	11
12	12	16	16	16	24	24	24	112	68	68	26	26	30	11
13	0	12	12	12	24	24	24	124	92	92	28	28	24	13
14	0	12	12	12	24	24	24	124	92	92	28	28	24	13
15	dv	12	12	12	24	24	24	124	92	92	28	28	24	13
16	12	dv	0	0	12	12	12	232	116	116	32	32	28	17
17	12	0	dv	0	12	12	12	232	116	116	32	32	28	17
18	12	0	0	dv	12	12	12	232	116	116	32	32	28	17
19	24	12	12	12	dv	0	0	352	164	164	112	112	40	97
20	24	12	12	12	0	dv	0	352	164	164	112	112	40	97
21	24	12	12	12	0	0	dv	352	164	164	112	112	40	97
22	76	160	160	160	232	232	232	dv	24	24	30	30	96	46
23	88	112	112	112	160	160	160	48	dv	0	6	10	84	22
24	88	112	112	112	160	160	160	48	0	dv	6	10	84	22
25	28	32	32	32	112	112	112	48	6	6	dv	6	14	16
26	28	32	32	32	112	112	112	48	10	10	6	dv	14	22
27	24	22	22	22	40	40	40	144	88	88	14	14	dv	30
28	13	17	17	17	97	97	97	64	22	22	16	22	30	dv
29	13	17	17	17	97	97	97	64	22	22	16	22	30	0
30	7	13	13	13	25	25	25	112	60	60	22	22	26	6
31	7	13	13	13	25	25	25	112	60	60	22	22	26	6
32	13	17	17	17	25	25	25	112	68	68	26	26	30	10

0	15	16	17	18	19	20	21	22	23	24	25	26	27	28
33	13	17	17	17	25	25	25	112	68	68	26	26	30	10
34	1	13	13	13	25	25	25	124	92	92	28	28	24	12
35	1	13	13	13	25	25	25	124	92	92	28	28	24	12
36	73	97	97	97	145	145	145	64	16	16	22	26	100	14
37	13	17	17	17	97	97	97	64	22	22	16	22	30	8
38	13	17	17	17	97	97	97	64	26	26	22	16	30	14
39	5020	5024	5024	5024	5104	5104	5104	5064	5026	5026	5022	5016	5030	5014
40	5020	5024	5024	5024	5104	5104	5104	5064	5026	5026	5022	5016	5030	5014
41	5016	5014	5014	5014	5032	5032	5032	5160	5104	5104	5030	5030	5016	5022
42	5016	5014	5014	5014	5032	5032	5032	5160	5104	5104	5030	5030	5016	5022
43	5016	5014	5014	5014	5032	5032	5032	5160	5104	5104	5030	5030	5016	5022

Table 1. *Continued*

0	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
1	27	61	61	69	69	93	93	21	27	31	372	372	446	446	446
2	1	7	7	11	11	13	13	7	1	7	348	348	356	356	356
3	1	7	7	11	11	13	13	7	1	7	348	348	356	356	356
4	1	7	7	11	11	13	13	7	1	7	348	348	356	356	356
5	27	27	27	31	31	33	33	31	27	21	362	362	376	376	376
6	7	1	1	5	5	7	7	45	7	7	348	348	352	352	352
7	7	1	1	5	5	7	7	45	7	7	348	348	352	352	352
8	7	1	1	5	5	7	7	45	7	7	348	348	352	352	352
9	11	5	5	1	1	13	13	53	11	11	352	352	356	356	356
10	11	5	5	1	1	13	13	53	11	11	352	352	356	356	356
11	11	5	5	1	1	13	13	53	11	11	352	352	356	356	356
12	11	5	5	1	1	13	13	53	11	11	352	352	356	356	356
13	13	7	7	7	7	1	1	77	13	13	354	354	350	350	350
14	13	7	7	7	7	1	1	77	13	13	354	354	350	350	350
15	13	7	7	7	7	1	1	77	13	13	354	354	350	350	350
16	17	13	13	17	17	13	13	101	17	17	358	358	354	354	354
17	17	13	13	17	17	13	13	101	17	17	358	358	354	354	354
18	17	13	13	17	17	13	13	101	17	17	358	358	354	354	354
19	97	25	25	25	25	25	25	149	97	97	438	438	366	366	366
20	97	25	25	25	25	25	25	149	97	97	438	438	366	366	366
21	97	25	25	25	25	25	25	149	97	97	438	438	366	366	366
22	46	56	56	56	56	76	76	40	46	46	380	380	446	446	446
23	22	56	56	64	64	88	88	16	22	26	360	360	434	434	434
24	22	56	56	64	64	88	88	16	22	26	360	360	434	434	434
25	16	22	22	26	26	28	28	22	16	22	356	356	364	364	364
26	22	22	22	26	26	28	28	26	22	16	350	350	364	364	364
27	30	26	26	30	30	24	24	104	30	30	364	364	350	350	350
28	0	6	6	10	10	12	12	14	8	14	348	348	356	356	356
29	dv	6	6	10	10	12	12	14	8	14	348	348	356	356	356
30	6	dv	0	4	4	6	6	52	14	14	348	348	352	352	352
31	6	0	dv	4	4	6	6	52	14	14	348	348	352	352	352
32	10	4	4	dv	0	12	12	60	18	18	352	352	356	356	356
33	10	4	4	0	dv	12	12	60	18	18	352	352	356	356	356
34	12	6	6	6	6	dv	0	84	20	20	354	354	350	350	350
35	12	6	6	6	6	0	dv	84	20	20	354	354	350	350	350
36	14	48	48	56	56	80	80	dv	6	10	352	352	426	426	426
37	8	14	14	18	18	20	20	6	dv	6	348	348	356	356	356
38	14	14	14	18	18	20	20	10	6	dv	342	342	356	356	356
39	5014	5014	5014	5018	5018	5020	5020	5018	5014	5008	dv	0	14	14	14
40	5014	5014	5014	5018	5018	5020	5020	5018	5014	5008	0	dv	14	14	14
41	5022	5018	5018	5022	5022	5016	5016	5096	5022	5022	14	14	dv	0	0
42	5022	5018	5018	5022	5022	5016	5016	5096	5022	5022	14	14	0	dv	0
43	5022	5018	5018	5022	5022	5016	5016	5096	5022	5022	14	14	0	0	dv

By analyzing Table 1, there are identical nodes such as nodes 2,3,4. These identical nodes have the same cost values to other nodes in the network, while their inner-cost equals to zero. This means that only one node of them can be kept for solution, and then the other nodes are inserted left or right the remaining one, as it will be shown later. Similarly, nodes {6,7,8}, {9,10,11,12}, {13,14,15}, {16,17,18}, {19,20,21}, {28,29}, {30,31}, {32,33}, {34,35}, {39,40} and {41,42,43}

are identical nodes. Table 2 list the identical nodes in the network, and the selected node to keep. Any of the identical nodes can be kept for the solution, however the node with smaller ID will be kept.

Table 2. *List of identical nodes*

#	List of identical nodes	Selected node to keep	Nodes to be removed from network
1	2, 3, 4	2	3, 4
2	6, 7, 8	6	7, 8
3	9, 10, 11, 12	9	10, 11, 12
4	13, 14, 15	13	14, 15
5	16, 17, 18	16	17, 18
6	19, 20, 21	19	20, 21
7	23, 24	23	24
8	28, 29	28	29
9	30, 31	30	31
10	32, 33	32	33
11	34, 35	34	35
12	39, 40	39	40
13	41, 42, 43	41	42, 43

From the Table 2, there are thirteen groups of identical nodes. From each group, only one node will be kept within network and the other nodes will be removed, and a new reduced cost matrix will be generated, and shown in Table 3.

The minimum cycle will go through the identical nodes in any order and are alternate for minimum cycle. This means that the minimum cycle include nodes (2,3,4) in this order, or in any other order such that (4,3,2) or (3,4,2).

The reduced network in Table 3will have same solution as original network in Table 1. The reduced network will have 22 nodes, while 21 nodes will be removed.

Table 3. *Reduced cost of edges (Origin-Destination Matrix)(diagonal valued v=99999)*

	1	2	5	6	9	13	16	19	22	23	25	26	27	28	30	32	34	36	37	38	39	41
1	dv	26	10	60	68	92	116	164	84	36	42	46	120	27	61	69	93	21	27	31	372	446
2	36	dv	36	6	10	12	16	96	64	22	16	22	30	1	7	11	13	7	1	7	348	356
5	10	26	dv	26	30	32	36	116	84	46	42	36	50	27	27	31	33	31	27	21	362	376
6	74	6	36	dv	4	6	12	24	112	60	22	22	26	7	1	5	7	45	7	7	348	352
9	82	10	40	4	dv	12	16	24	112	68	26	26	30	11	5	1	13	53	11	11	352	356
13	106	12	42	6	6	dv	12	24	124	92	28	28	24	13	7	7	1	77	13	13	354	350
16	130	16	46	12	16	12	dv	12	232	116	32	32	28	17	13	17	13	101	17	17	358	354
19	178	96	126	24	24	24	12	dv	352	164	112	112	40	97	25	25	25	149	97	97	438	366
22	70	46	76	56	56	76	160	232	dv	24	30	30	96	46	56	56	76	40	46	46	380	446
23	46	22	56	56	64	88	112	160	48	dv	6	10	84	22	56	64	88	16	22	26	360	434
25	52	16	52	22	26	28	32	112	48	6	dv	6	14	16	22	26	28	22	16	22	356	364
26	56	22	46	22	26	28	32	112	48	10	6	dv	14	22	22	26	28	26	22	16	350	364
27	134	30	60	26	30	24	22	40	144	88	14	14	dv	30	26	30	24	104	30	30	364	350
28	37	1	37	7	11	13	17	97	64	22	16	22	30	dv	6	10	12	14	8	14	348	356
30	75	7	37	1	5	7	13	25	112	60	22	22	26	6	dv	4	6	52	14	14	348	352
32	83	11	41	5	1	13	17	25	112	68	26	26	30	10	4	dv	12	60	18	18	352	356
34	107	13	43	7	7	1	13	25	124	92	28	28	24	12	6	6	dv	84	20	20	354	350
36	31	7	41	41	49	73	97	145	64	16	22	26	100	14	48	56	80	dv	6	10	352	426
37	37	1	37	7	11	13	17	97	64	22	16	22	30	8	14	18	20	6	dv	6	348	356
38	41	7	31	7	11	13	17	97	64	26	22	16	30	14	14	18	20	10	6	dv	342	356
39	5048	5014	5038	5014	5018	5020	5024	5104	5064	5026	5022	5016	5030	5014	5014	5018	5020	5018	5014	5008	dv	14
41	5126	5022	5052	5018	5022	5016	5014	5032	5160	5104	5030	5030	5016	5022	5018	5022	5016	5096	5022	5022	14	dv

3. Minimum Travel Cost Array

The minimum travel cost array for the network is the foundation for this approach. It will be created such that the cost to travel each node is the minimal possible value, as shown in Table 4. The least arrival and departure cost will be selected.

Table 4. *Initial minimum travel cost array*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
10	5	1	5	10	20
1	28,37	2	28	1	2
10	1	5	1	1	2
1	30	6	30	1	2
1	32	9	32	1	2
1	34	13	34	1	2
12	6,19,13	16	6	12	24
12	16	19	16	12	24
48	25,26	22	23	24	72
6	25	23	25	6	12
6	23,26	25	23,26	6	12
6	25	26	25	6	12
14	25,26	27	25,26	14	28
1	2	28	2	1	2
1	6	30	6	1	2
1	9	32	9	1	2
1	13	34	13	1	2
6	37	36	37	6	12
1	2	37	2	1	2
6	37	38	37	6	12
14	41	39	41	14	28
14	39	41	39	14	28
173	Total			140	304

$C_{arrival}$ is the cost to arrive to node and $C_{departure}$ is the cost to leave the node

3.1. Initial Array

The initial minimum travel array, displayed in Table 4, describes important characteristics for the network. For each node, the least cost to arrive is selected, and the least cost to depart from node is also selected.

Some nodes have more than one node as minimum travel cost, such as node (16) has three nodes with same value in arrival direction, and node (25) has two nodes in both directions.

The cost of the least cycle will exceed the higher value of each side sum of cost which equals 173 from arrival side.

The initial array in Table 4 does not include the required least cycle due to the following reasons:

1) The total cost for both sides are not equal
2) Closed loops exist for same node such as from 5 to 1, and from 1 to 5.

3.2. Adjacent Nodes

Adjacent-nodesare nodes tied to each other, where the minimal travel cost for node (i) is (jij) and in the same time the minimal travel cost for node (j) is (iji). This implies that the required minimal cycle will go through (ij or ji), and both must be adjacent. In Table 4, nodes (1,5) and nodes (39,41) are examples for adjacent-nodes.

Table 5. *List of adjacent-nodes*

#	Adjacent-nodes
1	(1,5)
2	(6,30)
3	(9,32)
4	(13,34)
5	(39,41)

These nodes give more degree of freedom for solution and can be possible alternates for minimum required cycle. Table 5 shows the list of adjacent-nodes.

The second version from minimum travel array will be created to avoid closed loops for same nodes, which is the case of all adjacent-nodes as shown in Table 6.

3.3. Second Array

Table 6. *Second minimum travel cost array*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
10	5	1	36	21	31
1	37	2	28	1	2
10	1	5	38	21	31
1	30	6	9	4	5
1	32	9	6	4	5
1	34	13	6	6	7
12	19,13	16	6	12	24
12	16	19	6,9,13	24	36
48	25,26	22	23	24	72
6	25	23	26	10	16
6	23	25	26	6	12
6	25	26	23	10	16
14	25	27	26	14	28
1	2	28	30	6	7
1	6	30	32	4	5
1	9	32	30	4	5
1	13	34	30	6	7
6	37	36	2	7	13
6	38	37	2	1	7
6	37	38	2	7	13
342	38	39	41	14	356
350	13	41	39	14	364
837	Total			225	1062

The blue fill indicates the change in values
The red bold means the node values can be reversed

Table 6 includes the second minimum travel array for each node, after preventing same node to be in arrival and departure directions. As example, node (1) in Table 4 had to start from node (5) to node (1) and return back to node (5), which is not allowed by definition in TSP. The change from Table 4to Table 6is marked in blue fill for cells.

Also, there are some nodes where the travel cost is constant in both directions, such as node (6). The minimal cost to cross node (6) is 5, and there are two possibilities for this travel. The first is to start from node (9) to node (6) then from node (6) to node (30). The second is the opposite direction which is (30, 6, 9). Both have to be considered for the required minimum cycle.The nodes where both directions have equal cost have red font with bold style.

Table 6 also shows increase in total cost for both sides, from arrival direction cost is 837, and from departure direction cost is 225. This means that the cost of least cycle will exceed 837.

3.4. Opposite Links

The second array in Table 6, although it prevented closed loops for the same node, allowed the opposite direction for same edge in different nodes. For example, in node (1), the travel path starts from node (1) to node (5). If we consider

node (5), its travel path starts from node (5) to node (1). Such contradiction needs to be prevented. Table 7 displays opposite edges in minimum travel array.

Table 7. *Opposite edges*

$C_{arrival}$	From	Node	To	$C_{departure}$
10	5	1		
10	1	5		
1	30	6		
1	6	30		
		9	6	4
		6	9	4
1	32	9		
1	9	32		
1	34	13		
1	13	34		
12	19,13	16		
12	16	19		
6	25	23		
6	23	25		
		23	26	10
		26	23	10
		30	32	4
		32	30	4
		39	41	14
		41	39	14

In Table 7, it is clear that adjacent-nodes appear in opposite directions as bounded by solid border. Testing each direction for adjacent-nodes is important to decide which direction has least cost and to be added to least cycle. The analysis will start with one-direction nodes and of highest cost, which are adjacent-nodes (1,5) and (39,41).

3.5. Resolving Opposite Nodes

Figure 1. shows the two possibility to arrange nodes (1,5) and (39,41).

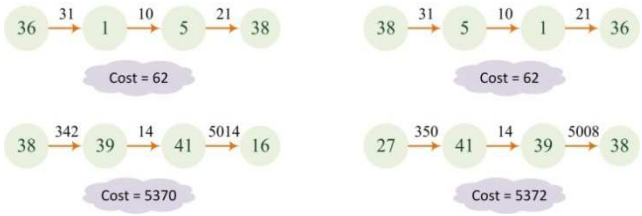

Figure 1. *Adjacent nodes (1,5) and (39,41)*

Figure 1 shows that the orders of adjacent-nodes (1,5) and (5,1) have same cost of 62. However, the cost of order (39,41) is 5370 while its opposite direction (41,39) is higher by 2. In order to maintain minimal cost for network the order of (39, 41) will be selected. This decision will imply that the order of (1,5) to be selected so that the node (38) can be common between them, as shown in Figure 2.

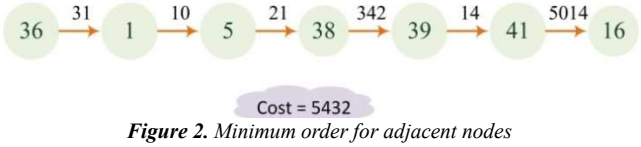

Figure 2. *Minimum order for adjacent nodes*

The minimum travel array will be updated to reflect the minimum order in Figure 2. Table 8 displays the third version of minimum travel array after its update with ordered adjacent-nodes shown in Figure 2. The cost of minimum cycle will exceed 5573 (sum of departure cost). In order for better understanding for the minimum travel array in Table 8, it will be rearranged as shown in Table 9.

Table 8. *Third minimum travel array*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
31	36	**1**	5	10	41
1	37	**2**	28	1	2
10	1	**5**	38	21	31
1	30	**6**	9	4	5
1	32	**9**	6	4	5
1	34	**13**	6	6	7
5014	41	**16**	6	12	5026
12	16	**19**	6,9,13	24	36
48	25,26	**22**	23	24	72
6	25	**23**	26	10	16
6	23	**25**	26	6	12
6	25	**26**	23	10	16
14	25	**27**	26	14	28
1	2	**28**	30	6	7
1	6	**30**	32	4	5
1	9	**32**	30	4	5
1	13	**34**	30	6	7
6	37	**36**	1	31	37
7	6	**37**	2	6	13
21	5	**38**	39	342	363
342	38	**39**	41	14	356
14	39	**41**	16	5014	5028
5545	Total			5573	11118

Green cells for nodes on minimum cycle.
Black bold nodes have final decision.

Table 9. *Third minimum travel array (rearranged)*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
6	37	**36**	1	31	37
31	36	**1**	5	10	41
10	1	**5**	38	21	31
21	5	**38**	39	342	363
342	38	**39**	41	14	356

14	39	41	16	5014	5028
5014	41	16	6	12	5026
1	37	2	28	1	2
1	30	6	9	4	5
1	32	9	6	4	5
1	34	13	6	6	7
12	16	19	6,9,13	24	36
48	25,26	22	23	24	72
6	25	23	26	10	16
6	23	25	26	6	12
6	25	26	23	10	16
14	25	27	26	14	28
1	2	28	30	6	7
1	6	30	32	4	5
1	9	32	30	4	5
1	13	34	30	6	7
7	6	37	2	6	13
5545	Total			5573	11118

In Table 9, some nodes have opposite directions, such as node (6) is connected from 30 to 6, while node (30) has connection from 6 to 30, this connection must be resolved. As node (6) and node (30) both can be reversed without change in cost, node (6) will remain constant, and node (30) will be reversed. Also, nodes (9, 23, 30, 34) will reversed as shown in Table 10.

Table 10. *Third minimum travel array (single direction all links)*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
6	37	36	1	31	37
31	36	1	5	10	41
10	1	5	38	21	31
21	5	38	39	342	363
342	38	39	41	14	356
14	39	41	16	5014	5028
5014	41	16	6	12	5026
1	37	2	28	1	2
1	30	6	9	4	5
4	6	9	32	1	5
1	34	13	6	6	7
12	16	19	6,9,13	24	36
48	25,26	22	23	24	72
10	26	23	25	6	16
6	23	25	26	6	12
6	25	26	23	10	16
14	25	27	26	14	28
1	2	28	30	6	7
4	32	30	6	1	5
1	9	32	30	4	5
6	30	34	13	1	7
7	6	37	36	6	13
5560	Total			5558	11118

Highlighted nodes in blue are reversed in this table

In Table 10, all the opposite links were resolved. The minimum bound for the network is 5560 from arrival side.

3.6. Starting Connecting Nodes

After the arrangement of all links in unique directions, the solution will be started. The node with the highest travel cost will start connecting to minimum cycle. FromTable 10 Third minimum travel array (single direction all links)Table 9, node (22) has the highest travel cost, in addition it has single direction from node (25) or (26) to node (23). In order to make decision about the order of node (22), nodes (23,25,26,27) need to be analysed in the same time. It is clear that these five nodes are adjacent to each other.

Figure 3 displays the possible order for these nodes, the lower order has cost of 92 which is less than the upper order. The arrival and departure for these nodes need also to be minimal.

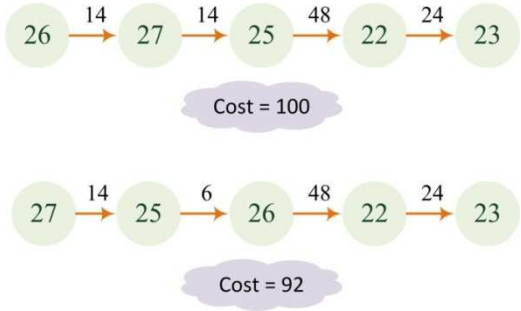

Figure 3. *Possible order for adjacent-nodes (22,23,25,26,27)*

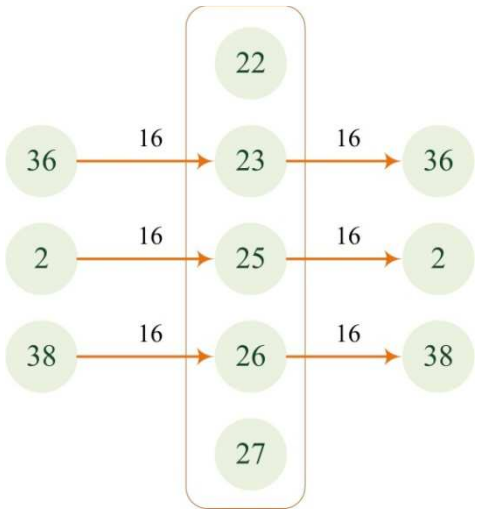

Figure 4. *Arrival and departure possibilities for adjacent-nodes (22,23,25,26,27)*

From Figure 4, the minimum cost to arrive to adjacent-nodes is 16, same as the minimum cost to leave them. Node (38) cannot be used anymore as it was connected previously. The order of upper nodes in Figure 3 will be changed to allow the cluster of nodes to start from node (25) and end at node (23), as shown in Figure 5.

Figure 5. *Modified order for nodes (22,23,25,26,27)*

Now, the minimum travel array in Table 10 will be updated to reflect the developing solution.

3.7. Developing the Solution (Fourth Array)

Table 11. *Fourth minimum travel array*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
1	37	2	25	16	17
16	2	25	27	14	30
14	25	27	26	14	28
14	27	26	22	48	62
48	26	22	23	24	72
24	22	23	36	16	40
16	23	36	1	31	47
31	36	1	5	10	41
10	1	5	38	21	31
21	5	38	39	342	363
342	38	39	41	14	356
14	39	41	16	5014	5028
5014	41	16	6	12	5026
1	30	6	9	4	5
4	6	9	32	1	5
1	34	13	6	6	7
12	16	19	6,9,13	24	36
7	6	28	30	6	13
4	32	30	6	1	5
1	9	32	30	4	5
6	30	34	13	1	7
7	6	37	2	6	13
5608	Total			5629	11237

In Table 11, the remaining maximum travel cost is for node (19). It will have arrival node from node (16).

Table 12. *Fourth array with node (19) connected*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
1	37	2	25	16	17
16	2	25	27	14	30
14	25	27	26	14	28
14	27	26	22	48	62
48	26	22	23	24	72
24	22	23	36	16	40
16	23	36	1	31	47
31	36	1	5	10	41
10	1	5	38	21	31
21	5	38	39	342	363
342	38	39	41	14	356
14	39	41	16	5014	5028
5014	41	16	19	12	5026
12	16	19	6,9,13	24	36
1	30	6	9	4	5
4	6	9	32	1	5
1	34	13	6	6	7
7	6	28	30	6	13
4	32	30	6	1	5
1	9	32	30	4	5
6	30	34	13	1	7
7	6	37	2	6	13
5608	Total			5629	11237

The solution now is developed based on minimum cost travel for high-cost nodes, as shown in Figure 6.

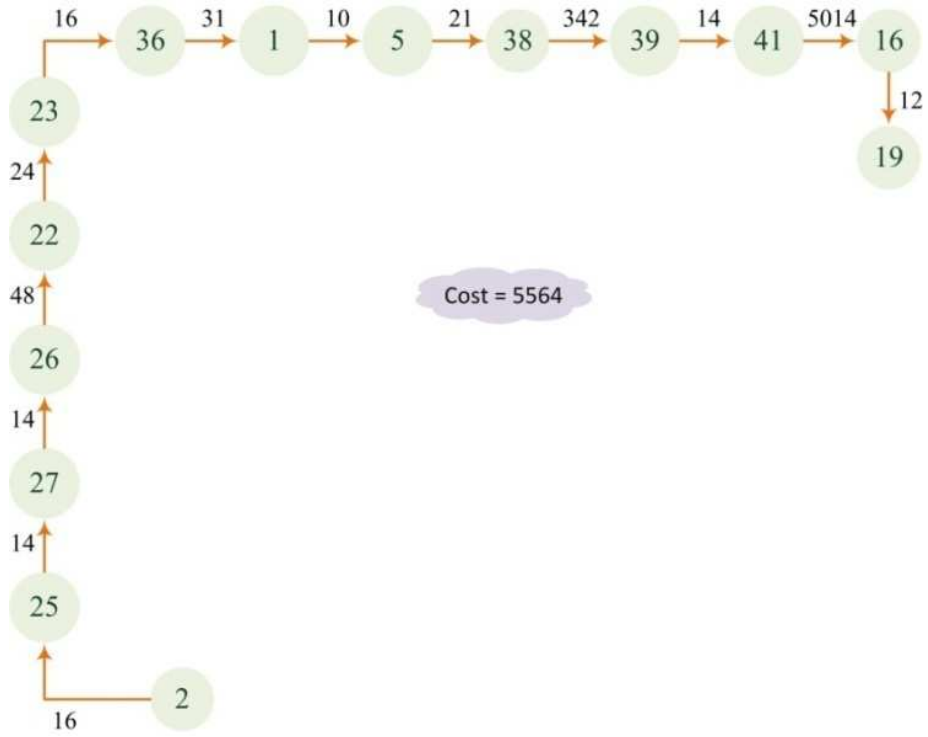

Figure 6. *Node (19) connected to solution*

3.8. Connecting Remaining Nodes

Furthermore, four groups of adjacent-nodes, making four clusters need to be connected in minimal cost to the cycle in Figure 6. The group of nodes (2,28,37) have to arrive to node (25), and all three nodes have same arrival cost. Also, same

group has to depart from group of nodes (6,30), as shown in Figure 7.

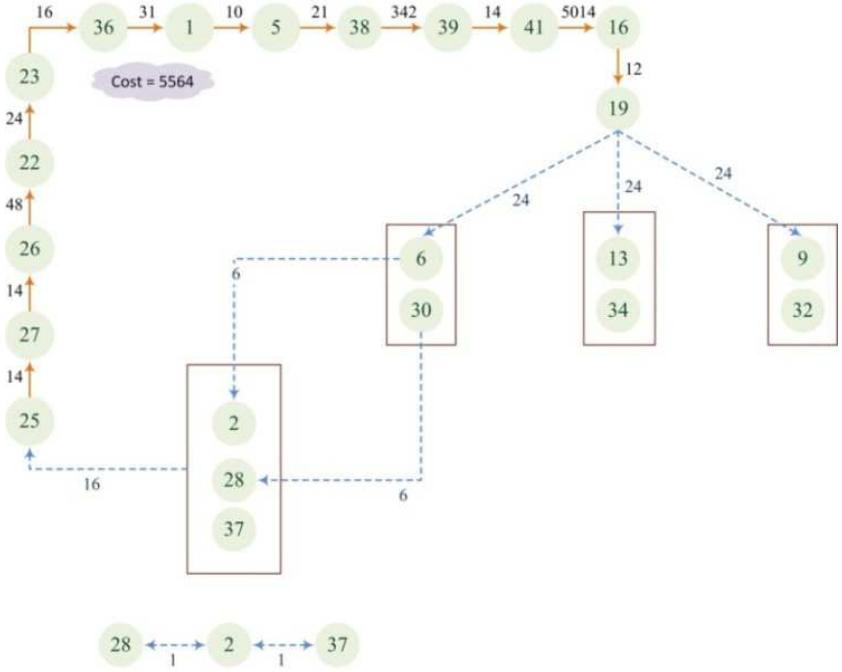

Figure 7. *Possible decisions for Node (19) departure*

In Figure 7, cluster (6,30) is required to be arrival for cluster (2,28,37) to maintain minimal connectivity. So, the cluster (6,30) will not be available for node (19) departure.This will reduce the possibility of node (19) to depart only to node (13) or node (9). This solution is shown in Figure 8.Also, as Figure 8 displays, the cluster (6,30) needs to arrive from node (9). Also, cluster (13,34) needs to depart to node (32). This means that the cluster (9,32) will not be departure for node (19). This leads that the cluster (13,34) will remain the only option for the departure of node (19).

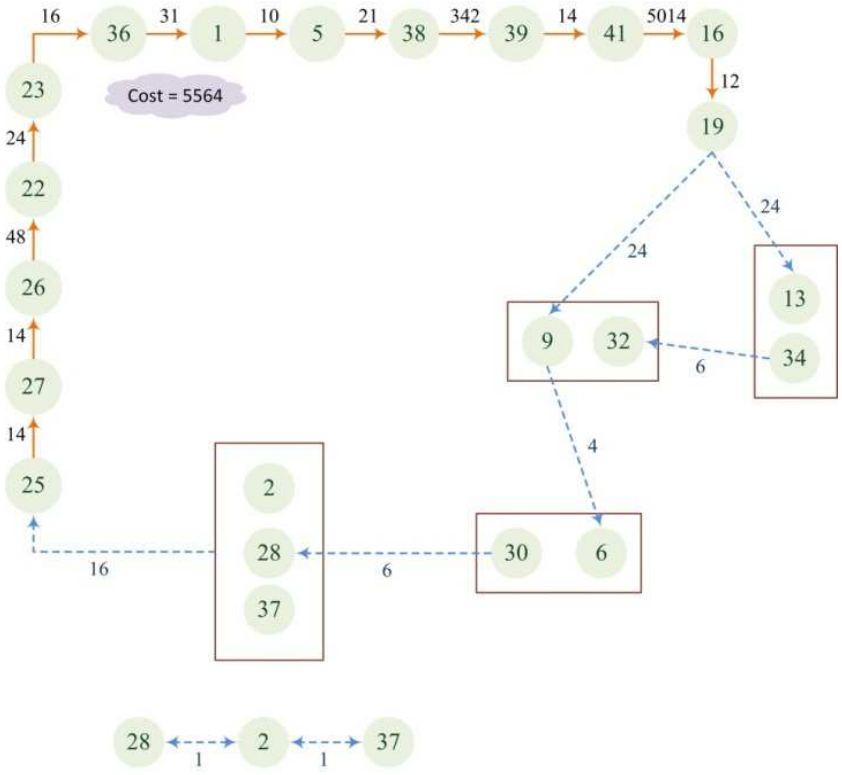

Figure 8. *Decision for Nodes (2, 28, 37)*

Figure 9 displays the development of he solution and the arrangement of adjacent-nodes in minimal cost. After that the final solution is achieved.

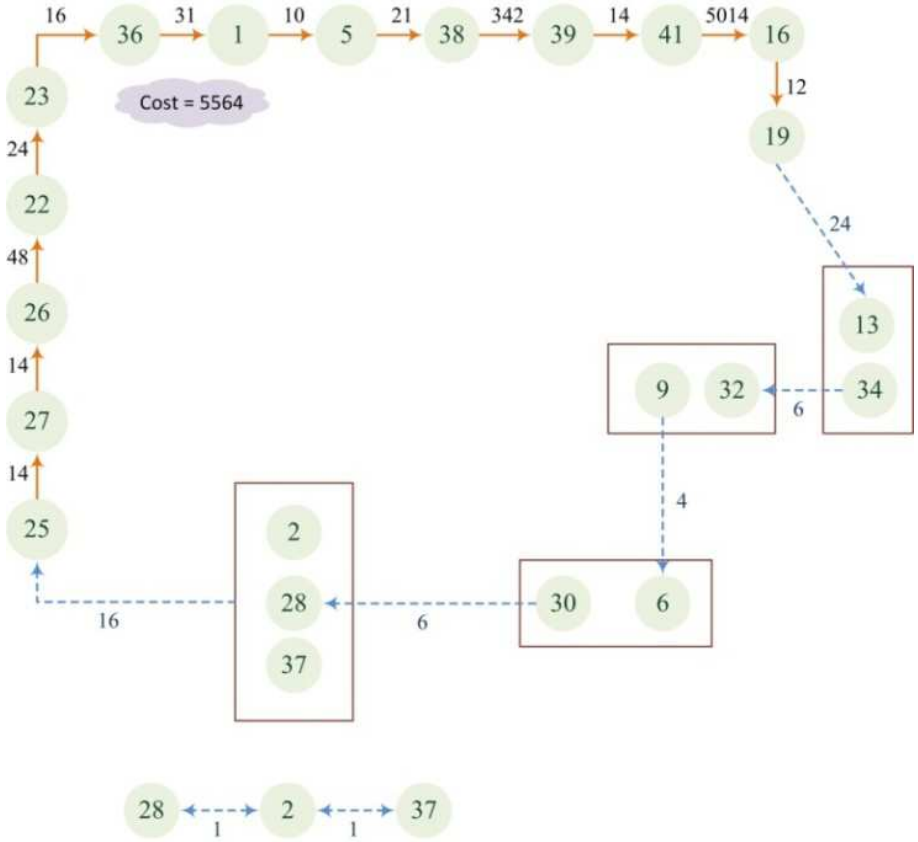

Figure 9. *Decision for Nodes (19, 34, 9)*

3.9. Final Solution for Reduced Network

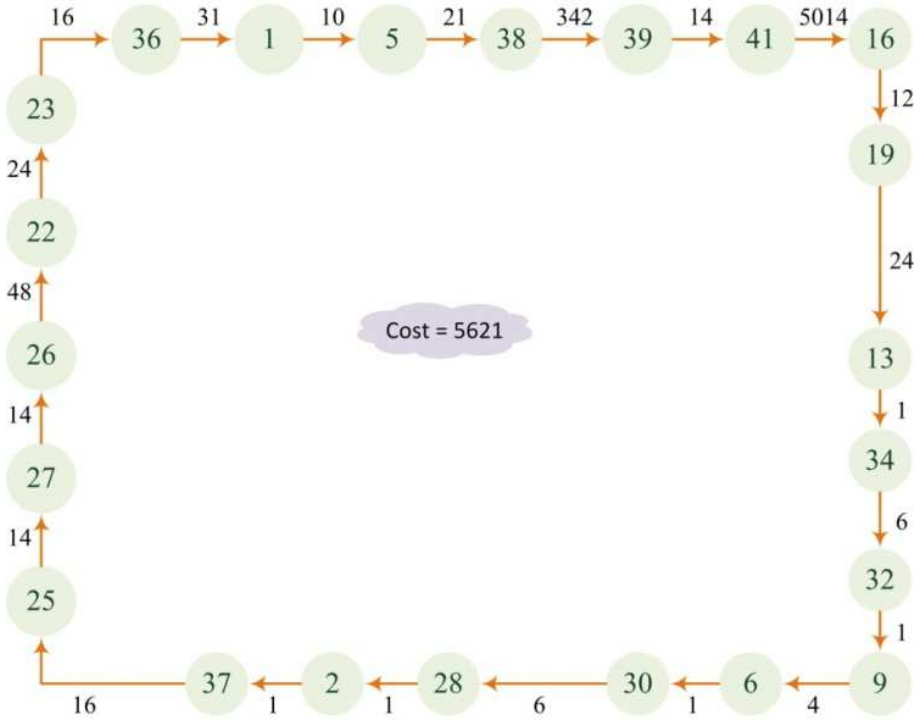

Figure 10. *Final solution for reduced network*

Final solution is displayed in Figure 10. In Figure 11, another non-least cycle is displayed.

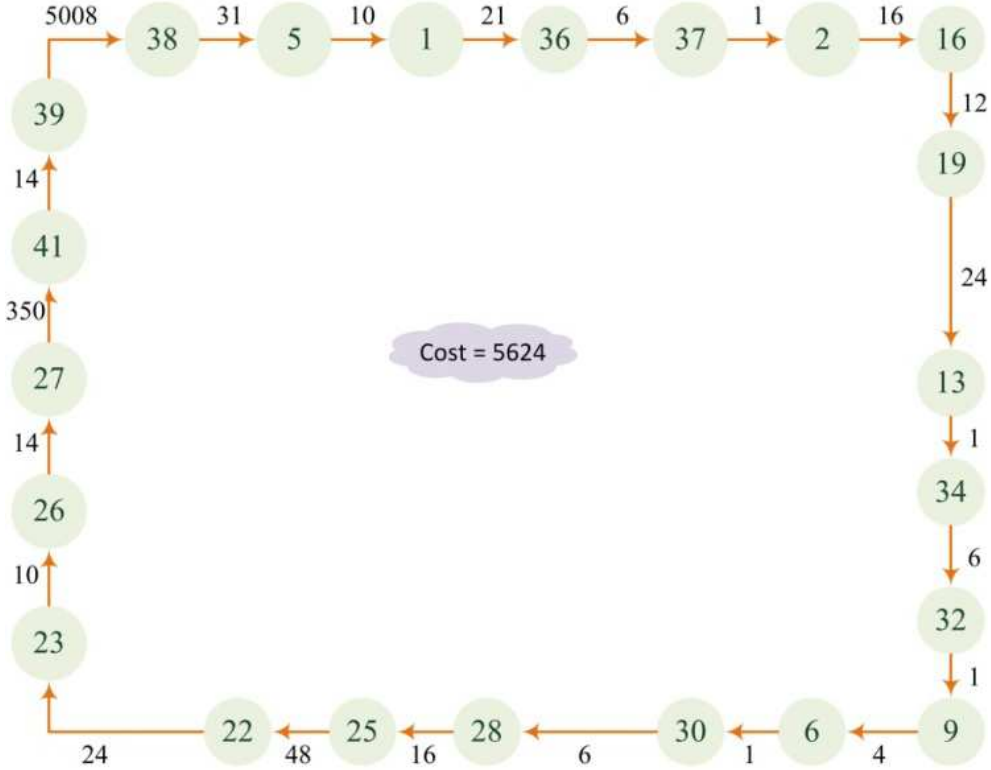

Figure 11. *Non-least cycle (cost = 5624)*

In Figure 12, Figure 13, Figure 14, and Figure 15, there are four near-minimum cycles for the reduced network.

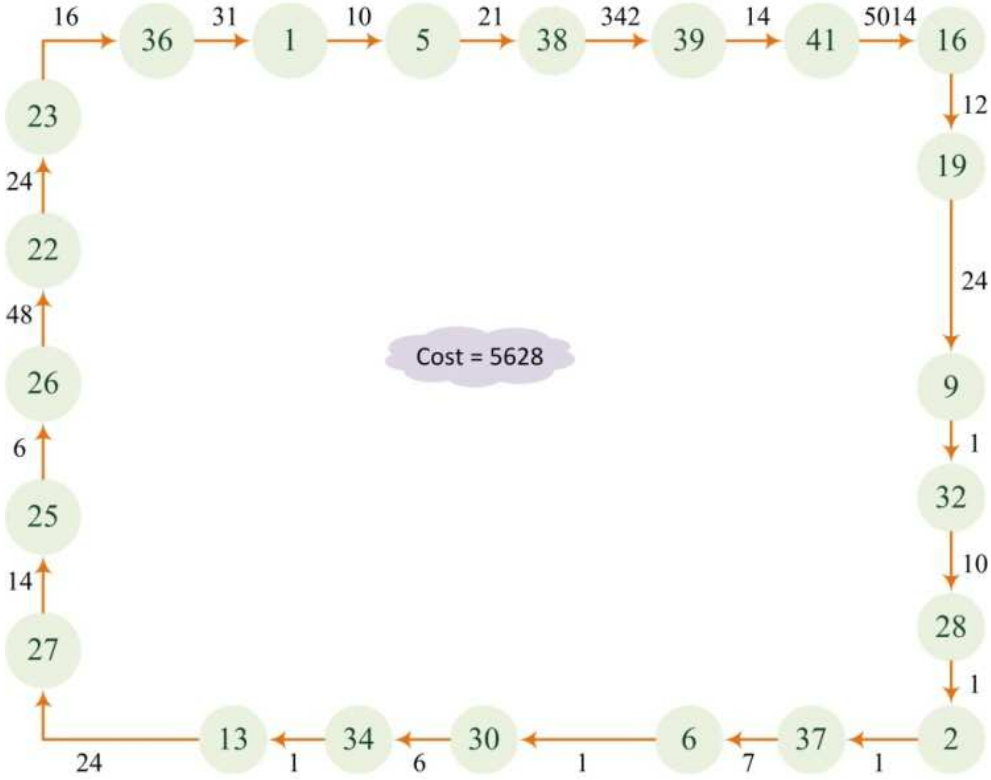

Figure 12. *Non-least cycle (cost = 5628)*

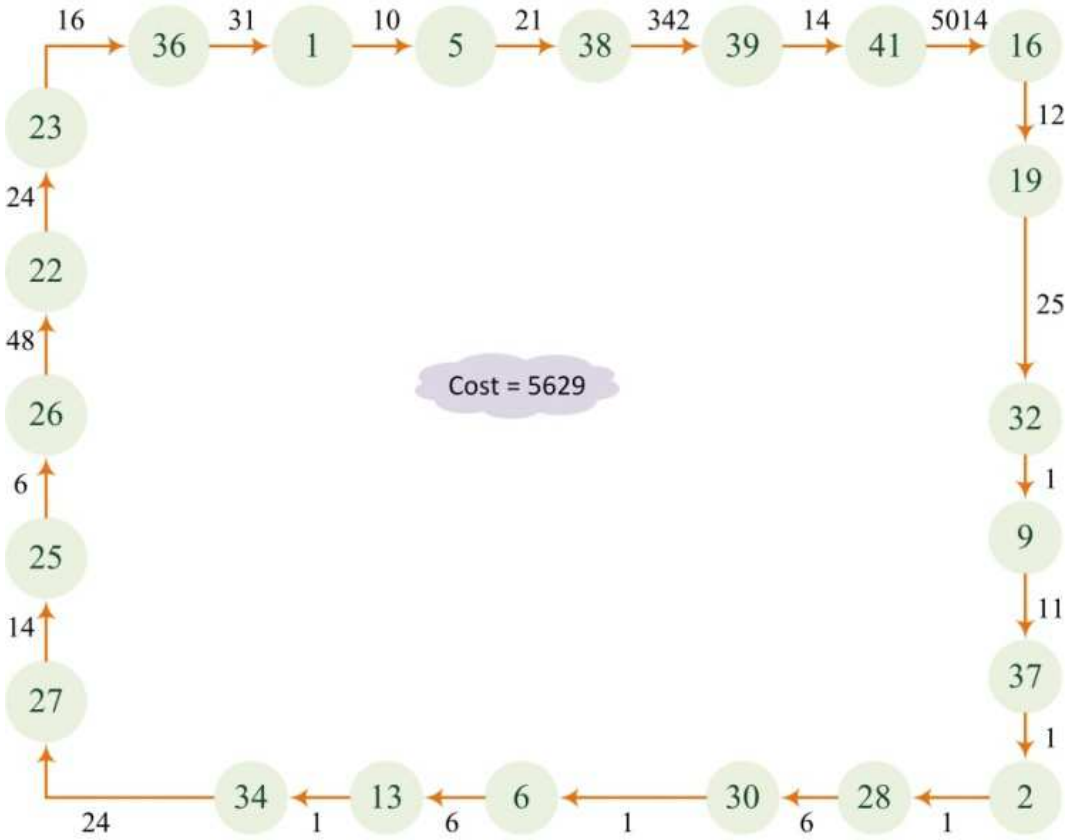

Figure 13. *Non-least cycle (cost = 5629)*

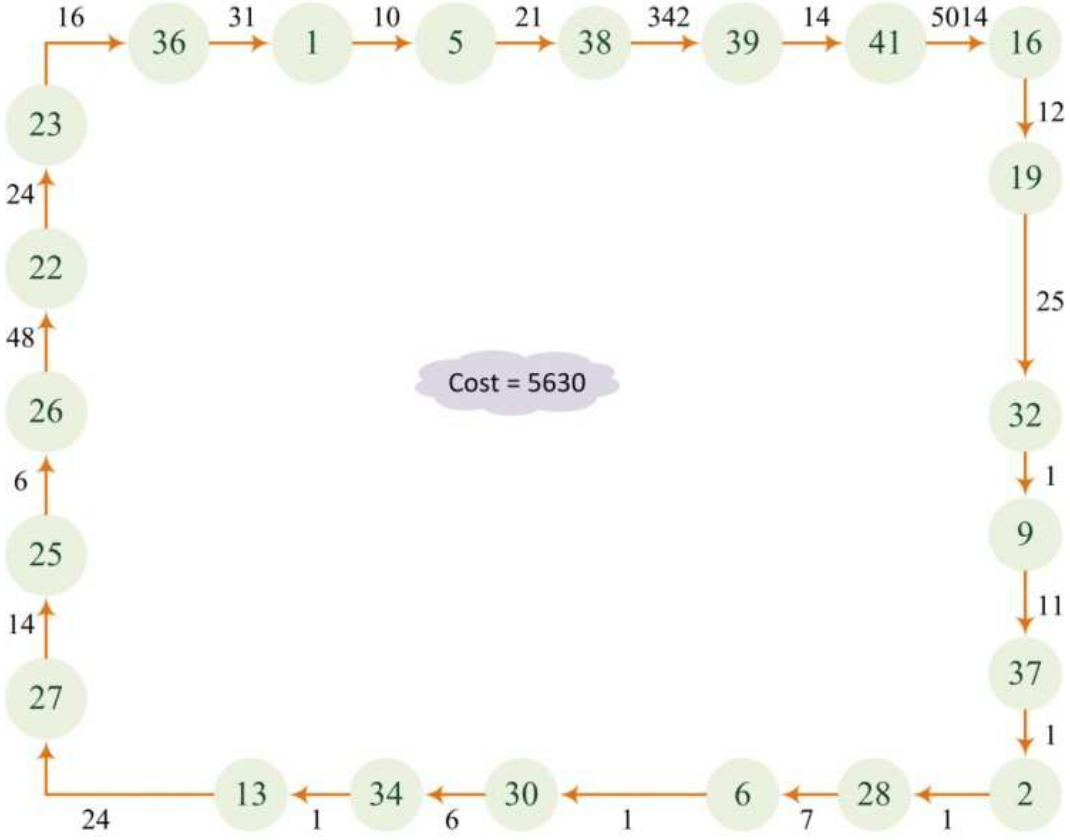

Figure 14. *Non-least cycle (cost = 5630)*

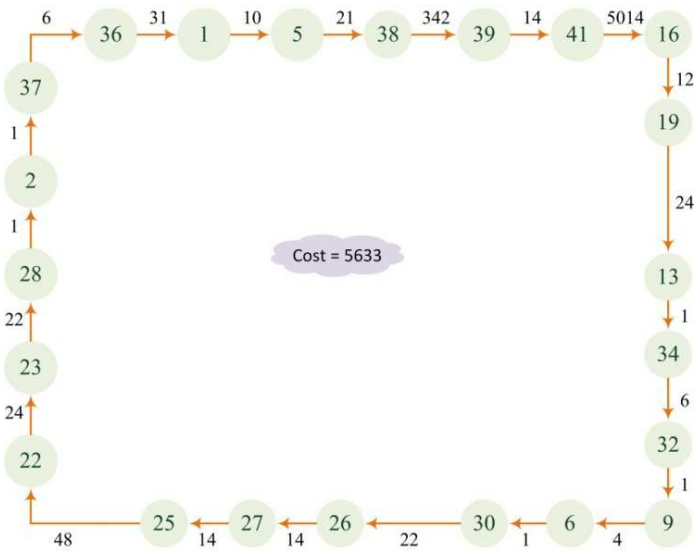

Figure 15. *Non-least cycle (cost = 5633)*

3.10. Final Solution for Full Matrix

By returning to the original problem of full matrix, all the identical nodes are connected with zero cost to least cycle, and the final solution is displayed in Figure 16.

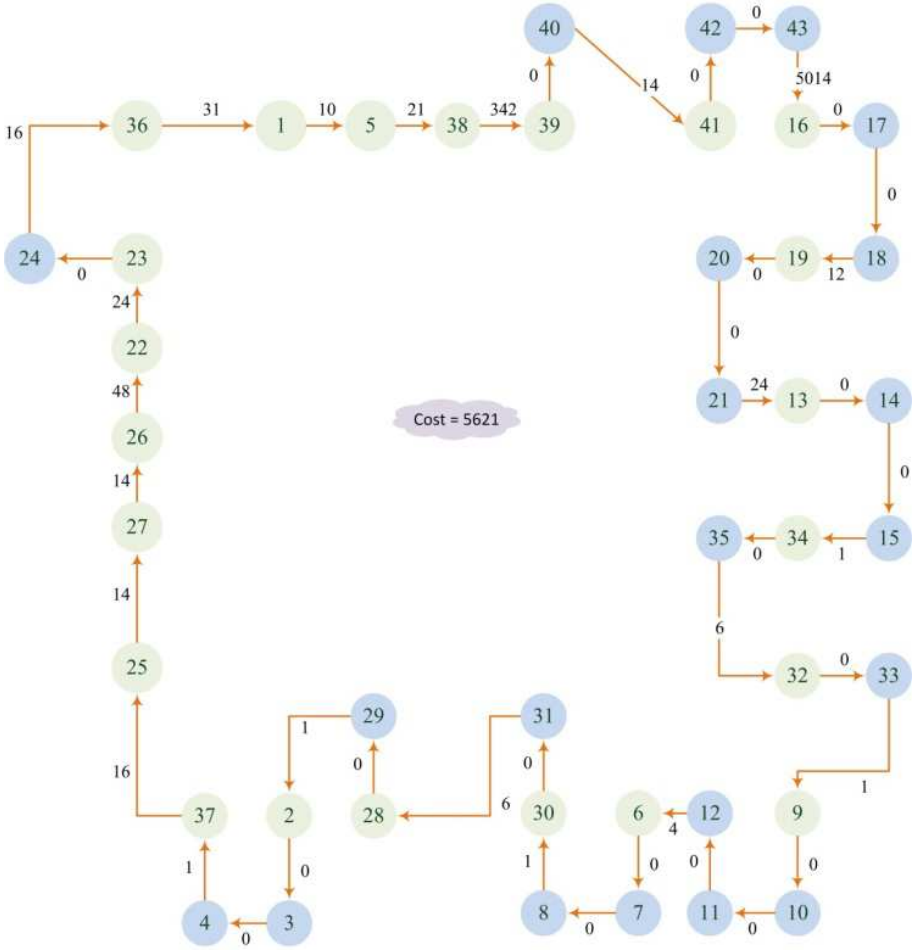

Figure 16. *Final solution for full network*

Green nodes are from reduced network
Blue nodes were not included in reduced network

Table 13. *Final solution array*

$C_{arrival}$	From	Node	To	$C_{departure}$	Cost Sum
31	36	1	5	10	41
10	1	5	38	21	31
21	5	38	39	342	363
342	38	39	41	14	356
14	39	41	16	5014	5028
5014	41	16	19	12	5026
12	16	19	13	24	36
24	19	13	34	1	25
1	13	34	32	6	7
6	34	32	9	1	7
1	32	9	6	4	5
4	9	6	30	1	5
1	6	30	28	6	7
6	30	28	2	1	7
1	28	2	37	1	2
1	2	37	25	16	17
16	37	25	27	14	30
14	25	27	26	14	28
14	27	26	22	48	62
48	26	22	23	24	72
24	22	23	36	16	40
16	23	36	1	31	47
5621	Total			5621	11242

4. Summary of Approach

1) Discover the identical nodes where their inner-cost equals zero in original network (Table 2), and remove them. Then, create the reduced network to be solved (Table 3).
2) Create the minimum travel cost array (Table 4) based absolute minimum cost of arrival to, and departure from, for each node in the network.
3) Discover adjacent-nodes in reduced network (Table 5). They make clusters in network, and need to be joined in minimal cost, and have minimal arrival and departure cost to least cycle.
4) Create 2nd minimum travel array (Table 6), where the arrival node is not the same as departure node for each node.
5) Create 3rd minimum travel array (Table 10), to remove opposite links in 2nd array.
6) Compute the minimum bound for the network. It equals the maximum value from the total cost of arrivalsideor the total cost of departure side from 3rd array (Table 10).
7) Start developing the solution by selecting the node with the highest travel cost and single direction to connect it to the least cycle (node 22 in Table 10).
8) For each cluster of nodes, the inner-cost of its nodes in addition to its arrival and departure cost must be minimal (Figure 4).
9) Compute available alternatives for the least cycle.
10) The first solution achieved solution, if it is not the least cycle, then it is the maximum bound for the network.

5. Discussion

1. The minimum travel cost approach is very effective in solving the general case for TSP. It starts with the least arrival and least departure cost for each node.
2. This approach computes a deterministic minimum bound for TSP based on the minimal travel for each node.
3. The minimum travel cost for each node is computed under the following conditions:
   a. Arrival node is different than departure node
   b. Cost to travel the node (i) is minimum ($C_i = C_{ai} + C_{id}$ = minimum) where (i) is the node of interest, (a) is the arrival node, and (d) is the departure node
   c. The link (ai) and the link (id) should not exist in opposite direction for any other node.
4. The solution should start with nodes with highest travel cost and uni-direction.
5. Adding any number of nodes identical to any node, with cost of zero to this node, does not affect the cost of solution.
6. Each node should follow its minimal path unless it is used as minimal arrival or departure for another node.
7. The best known solution in the TSPLIB website is 5620. However, the best solution achieved by the author is 5621.
8. The tables and computation were developed using Microsoft Excel spreadsheet.

6. Conclusion

The minimum travel cost approach is an effective algorithm to determine the minimum bound, maximum bound, and near-exact least cycle in polynomial time. It discovers the patterns and clusters which dominate the least cycle. However, this approach does not verify the least cycle or give its deterministic value.The minimum travel cost algorithm analyses the TSP in details, and is able to discover the possible realizations for the least cost tour which is not possible in other heuristic or exact algorithms (Gutin and Punnen 2007).

This approach can be used reversely to determine the longest cycle for TSP.

References

[1] Eleiche, Mohamed, and Bela Markus. "Applying minimu travel cost approach to 17-nodes travelling salesman problem." *GEOMATIKAI KOZLEMENYEK* (RESEARCH CENTRE FOR ASTRONOMY AND EARTH SCIENCES, HUNGARIAN ACADEMY OF SCIENCES) XIII, no. 2 (2010): 15-22.

[2] The TSPLIB website (https://www.tsp.gatech.edu/problem/index.html)

[3] GutinG., and Punnen. A.P. (2007): Experimental Analysis of Heuristics for the ATSP. In Gutin&Punnen (Eds.): The Traveling Salesman Problem and Its Variations. Springer, 2007. p.369-444.