Efficient Algorithms for Identifying Loop Formation and Computing θ Value for Solving Minimum Cost Flow Network Problems

TIMOTHY MICHAEL CHÁVEZ
Department of Computational Modeling and Simulation Engineering
Old Dominion University
1300 Engineering & Computational Sciences Building, Norfolk, VA 23529
UNITED STATES OF AMERICA

DUC THAI NGUYEN
Department of Civil & Environmental Engineering
Old Dominion University
135 Kaufman Hall, Norfolk, VA 23529
UNITED STATES OF AMERICA

Abstract: - While the minimum cost flow (MCF) problems have been well documented in many publications, due to its broad applications, little or no effort have been devoted to explaining the algorithms for identifying loop formation and computing the θ value needed to solve MCF network problems. This paper proposes efficient algorithms, and MATLAB computer implementation, for solving MCF problems. Several academic and real-life network problems have been solved to validate the proposed algorithms; the numerical results obtained by the developed MCF code have been compared and matched with the built-in MATLAB function Linprog() (Simplex algorithm) for further validation.

Key-Words: - Graphs Theory, Graphs and Networks, Network Theory, Minimal Cost Flow, Loop Formation
Received: December 2, 2020. Revised: May 31, 2021. Accepted: June 16, 2021. Published: June 24, 2021.

1 Introduction

There exist linear-programming (LP) problems [1-8] which exhibit a special structure, one balanced equation for each node in the network, that can be exploited for the development of efficient algorithms to obtain optimal solutions. The minimum cost flow (MCF) problems [9-11] are considered in this study due to their broad industrial, logistics applications. The transportation problem, the assignment problem, the shortest path problem, and more can be considered special cases of the MCF problem [12-15]. Although the MCF problem can be naturally formulated as a LP problem, the special structure of MCF problems need to be exploited to efficiently solve large-scale systems, which are otherwise not possible, or highly resource intensive, using a traditional LP/SIMPLEX algorithm.

A common scenario of a network-flow problem arising in industrial logistics concerns regarding the distribution of goods/products from a set of originating (supply) nodes to a set of destination (demand) nodes. The number of units available at each supply node and the number of units required at each demand node is assumed to be known. The delivered product from supply nodes to demand nodes often pass through "transshipment" nodes (neither supply nor demand), which may represent warehouses or distribution centers along the distribution path. The objective is to minimize the total cost of shipping the products to satisfy all consumers' requests at demand nodes. To facilitate the discussions, a simple 5-node and 7-link network [10], shown in Fig 1, was chosen; it will be used to explain different steps in the conventional MCF algorithm, with special emphasis on developing an efficient algorithm for identifying the loop-formation and computing the value for θ. In Fig 1, nodes #1 and #3 are identified as supply nodes (with the values $b_1 = 6$ and $b_3 = 4$), while nodes #4 and #5 are identified as demand nodes (with the values $b_4 = -5$ and $b_5 = -5$), and node #2 is a transshipment node (with the value $b_2 = 0$). The cost (C_{ij}) for transporting each unit of product on any directional link i-j (from node "i" to node "j") are also shown in Fig 1.

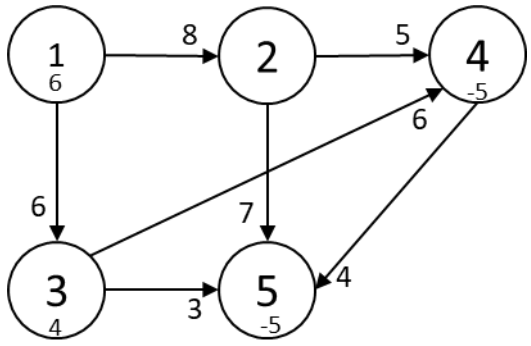

Fig 1. A 5-Node and 7-Link Network.

In Section 2, the MCF network presented in Fig 1 is formulated as a Linear Programming (LP) problem, so that its special structure can be high-lighted. Details of the three steps involved in each iteration for Phase 1 and Phase 2 of the MCF algorithms are explained in Section 3. The efficient algorithm for identifying loop-formation and computing θ value for MCF problems is also presented in Section 3. Numerical results for several academic and real-life networks are presented and validated in Section 4. Finally, conclusions are drawn in Section 5.

2 The MCF Linear Programming Problem

The MCF network presented in Fig 1 is formulated as a LP problem, to take advantage of one balanced equation for each node in the network. Based on the conservation of flow balance (equilibrium) at each node, such that:

$$FlowOut - FlowIn = NetSupply + Demand \quad (1)$$

The following equations can be written for the network shown in Fig 1:

Node	Flow Out – Flow In	Net Supply/Demand	
1	$+X_{12} + X_{13}$	$= b_1 = +6$	(2)
2	$-X_{12} + X_{24} + X_{25}$	$= b_2 = 0$	(3)
3	$-X_{13} + X_{34} + X_{35}$	$= b_3 = +4$	(4)
4	$-X_{24} - X_{34} + X_{45}$	$= b_4 = -5$	(5)
5	$-X_{25} - X_{35} - X_{45}$	$= b_5 = -5$	(6)

Thus, the following LP problem (Eq. 7) can be formulated from the network costs shown in Fig 1, and the balanced equations above.

The objective is to find the amount of link flow (X_{ij}) directly connected between nodes "i" and "j" such that the summation of the link costs times the link flows for the network is minimized:

$$MinimizeCOST = \sum_{i=1}^{n} \sum_{j=1}^{n} (C_{ij} \cdot X_{ij}) \quad (7)$$

with the constraints presented in Eqs. (2-6), n = number of nodes, C_{ij} = non-negative (integer) cost, and X_{ij} = non-negative (integer) flow.

It should be noted that the summation of all terms on the left-hand-side (LHS) of Eqs. (2-6) and the corresponding summation of all terms on the right-hand-side (RHS) of Eqs. (2-6) are both equal to zero. This special (unimodal) property will allow the MCF algorithm, which can be considered a special case of LP/Simplex solver, to obtain the integer values for X_{ij} at the optimum, without being required to solve a more costly Linear Integer Programming (LIP) problem. From the "duality theories" [1-4], the above 7-variable and 5-constraint "Primal" LP problem can be associated with the following 5-variable and 7-constraint "Dual" LP problem.

Find w_1 through w_5, such that $[(b_1 = 6) \cdot w_1 + (b_2 = 0) \cdot w_2 + (b_3 = 4) \cdot w_3 + (b_4 = -5) \cdot w_4 + (b_5 = -5) \cdot w_5]$ is maximized, and the following seven constraints (expressed in matrix notation) are satisfied:

$$\begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & -1 \end{bmatrix} \cdot \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ w_4 \\ w_5 \end{bmatrix} = \begin{bmatrix} C_{12} \\ C_{13} \\ C_{24} \\ C_{25} \\ C_{34} \\ C_{35} \\ C_{45} \end{bmatrix} \quad (8)$$

The above 7x5 matrix (of the "Dual" problem) is simply the transpose of the 5x7 coefficient matrix shown in Eqs. (2-6) of the "Primal" problem. The matrix shown in Eq. (8) can also be expressed as $w_i - w_j = C_{ij}$, and this equation will be used in Section 3.

3 The Minimum Cost Flow Algorithm

With regards to the conventional MCF problem, the objective for Phase 1 is to find the BFS, while the objective for Phase 2 is to obtain the OFS. The MCF method is an iterative process that consists of identifying which non-basic link (not currently in the solution) should be brought into the basic (active links) set, identifying the current loop, adjusting the link flow along the loop, and removing unneeded links.

3.1 Phase 1 – Calculate the Basic Feasible Solution

3.1.1 Initialization

To initialize the network and begin the iterative process of calculating the BFS, an artificial node #A

(with $b_A = 0$) is added to the original network (shown in Fig 1). Then, we add artificial links connected from supply nodes (node #1 and node #3) to node #A and artificial links connected from node #A to each demand and transshipment nodes (node #2, node #4, and node #5). (Note: transshipment nodes are treated as demand nodes.) In Phase 1, all artificial links have the cost $C_{ij} = \$1$ (cost-unit), and all original links have the cost $C_{ij} = \$0$ (cost-unit), as shown in Fig 2. This drives flow from the artificial links to the original links.

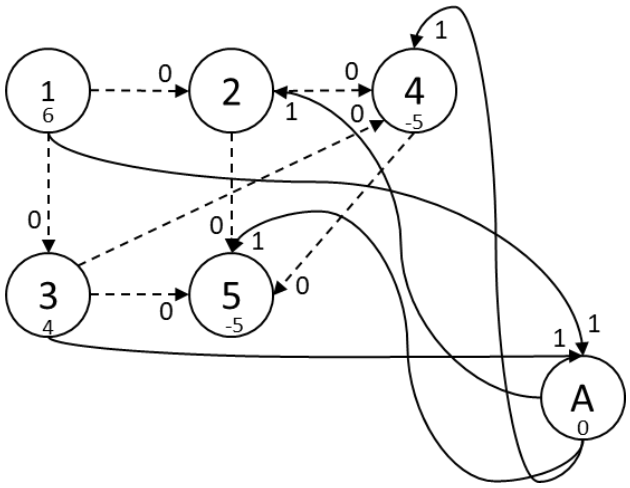

Fig 2. Addition of Artificial Node and Links

In Fig 2, there will be 10 units of flow-IN from the supply nodes ($b_1 = 6 + b_3 = 4$) to the artificial node #A and 10 units of flow-OUT from the artificial node #A to the demand nodes ($b_2 = 0 + b_4 = 5 + b_5 = 5$). This network is in equilibrium; 10 units IN and 10 units OUT. The initial cost for the network can be calculated as:

$$MinCOST = C_{1A} \cdot X_{1A} + C_{A2} \cdot X_{A2} + C_{3A} \cdot X_{3A} + C_{A4} \cdot X_{A4} + C_{A5} \cdot X_{A5}$$

$$\$1 \cdot (b_1 = 6) + \$1 \cdot (b_2 = 0) + \$1 \cdot (b_3 = 4) + \$1 \cdot (-b_4 = 5) + \$1 \cdot (-b_5 = 5) = \$20 \quad (9)$$

3.1.2 Iterative Step #1 – Calculate Node Weights

At the beginning of each iteration, we repeatedly apply the following equation among the basic links to calculate w, with $w_A = 0$ (associated with the artificial node #A):

$$w_i - w_j = C_{ij} \quad (10)$$

The equivalent version of Eq. (10), in matrix notation, has already been explained in Eq. (8) of Section 2.

Then, for our 5-node example with an artificial node:

$$w_1 - w_A = C_{1A}; w_1 - 0 = 1; w_1 = 1 \quad (11)$$
$$w_A - w_2 = C_{A2}; 0 - w_2 = 1; w_2 = -1 \quad (12)$$
$$w_3 - w_A = C_{3A}; w_3 - 0 = 1; w_3 = 1 \quad (13)$$
$$w_A - w_4 = C_{A4}; 0 - w_4 = 1; w_4 = -1 \quad (14)$$
$$w_A - w_5 = C_{A5}; 0 - w_5 = 1; w_5 = -1 \quad (15)$$

Remarks:

- In general, the selected #th node with the known/assigned value w is NOT always to be the artificial node. In the MATLAB code, $w = 0$ for node ## in which node ## has the largest number of connected basic links is selected; this should reduce the number of unknown node weights.
- In this example, since there are a total of six nodes (five original nodes plus the artificial node), the number of basic links required to solve for all w is five (six nodes total minus one).
- In applying Eq. (10), it is required that either w_i or w_j is known, so that either w_j (if w_i is known) or vice versa can be computed.
- Since this Step #1 is straight forward, the details for the MATLAB coding are not discussed here.

3.1.3 Iterative Step #2 – Determine Which Non-Basic Link Enters the Basic Set

Among the non-basic links (dashed links in Fig 2), one needs to determine which non-basic link should be added to the basic (solid links in Fig 2) group, based on the maximum positive value, v, in the following equation:

$$v = w_i - w_j - C_{ij} \quad (16)$$

Then, for the current example's non-basic links:

$$12: v = w_1 - w_1 - C_{12} = 1 - -1 - 0 = 2 \quad (17)$$
$$13: v = w_1 - w_3 - C_{13} = 1 - 1 - 0 = 0 \quad (18)$$
$$24: v = w_2 - w_4 - C_{24} = -1 - -1 - 0 = 0 \quad (19)$$
$$25: v = w_2 - w_5 - C_{25} = -1 - -1 - 0 = 0 \quad (20)$$
$$34: v = w_3 - w_4 - C_{34} = 1 - -1 - 0 = 2 \quad (21)$$
$$35: v = w_3 - w_5 - C_{35} = 1 - -1 - 0 = 2 \quad (22)$$
$$45: v = w_4 - w_5 - C_{45} = 1 - 1 - 0 = 0 \quad (23)$$

Remarks:

- Since the three non-basic links (1-2, 3-4, and 3-5) all have the same maximum positive value (= +2), based on Eq. (16), tie-breaker criterion needs to be considered. Although

some researchers have suggested a random selection is adequate in the case of a tie, the recommended criterion is to select the link which has the minimum, original cost (see Fig 1). Hence, in this example, link 3-5, with an original cost of $3, will be selected to enter the basic group.

- Since this Step #2 is straight forward, the details for MATLAB coding will not be discussed here.

3.1.4 Iterative Step #3 – Determine Which Basic Link is Removed

Among the basic links, one needs to determine which basic link(s) is/are to be removed from the basic set, by identifying the loop-formation and computing the value of θ.

At the end of Step #2, it has been determined that link 3-5 should "enter the basic set", with the flow value $X_{35} = \theta$, as shown in Fig 3.

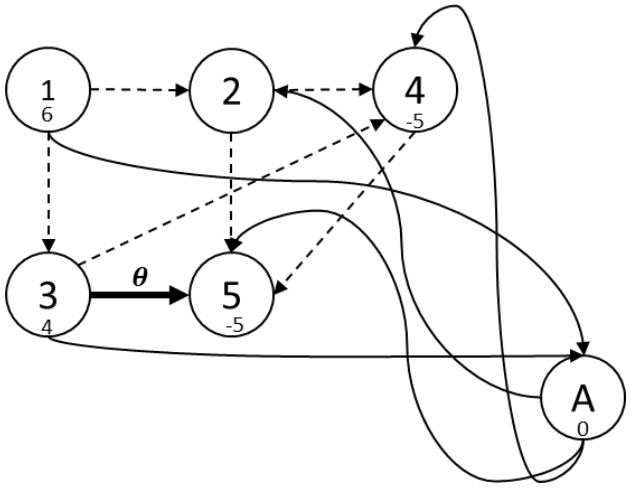

Fig 3. Link 3-5, As θ, Enters the Basic Set

By "observation", one can see that the loop can be formed by the following links 3-5 (with flow X_{35} = unknown value θ), 3-A (with flow $X_{3A} = 4 - \theta$), and A-5 (with flow $X_{A5} = 5 - \theta$) as shown in Fig 4.

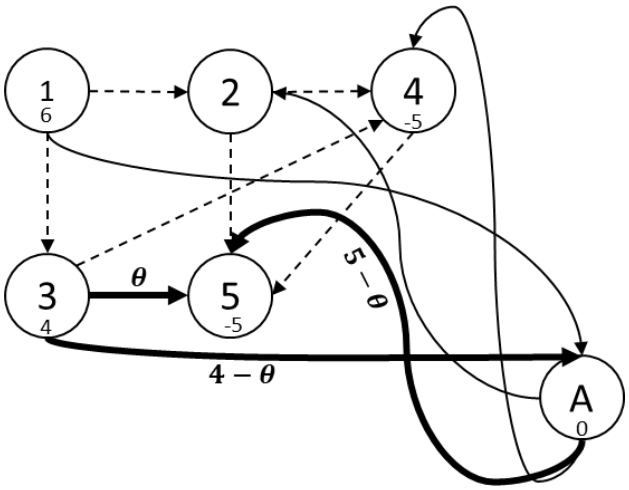

Fig 4. Observed Loop

The analysis is further restricted to only realistic solutions, introducing the constraint that the flow on any link must be "non-negative", then basic links:

$$35: X_{35} = \theta \geq 0 \quad (24)$$
$$3A: X_{3A} = 4 - \theta \geq 0; hence \theta \leq 4 \quad (25)$$
$$A5: X_{A5} = 5 - \theta \geq 0; hence \theta \leq 5 \quad (26)$$

Based on the requirements stated in Eqs. (24-26), one concludes that the requirement $\theta \leq 4$ will control the outcome; when $\theta \leq 4$ is satisfied, $\theta \leq 5$ is also automatically satisfied. Thus, the maximum positive value for θ is, $\theta = 4$.

Substituting the value of $\theta = 4$ into Eqs. (24-26), one obtains the following link values:

$$35: X_{35} = 4 \quad (27)$$
$$3A: X_{3A} = 4 - 4 = 0 (linkremoved) \quad (28)$$
$$A5: X_{A5} = 5 - 4 = 1 \quad (29)$$

To develop a simple algorithm which will automatically identify the loop-formation and compute the appropriate value for θ, one must answer the following 4 questions:

1. Among the basic links, how is it determined (automatically) which basic link i-j does NOT belong to the loop-formation; which links should NOT have their flow adjusted by the value of θ?
2. How is it determined (automatically) which basic links i-j (such as basic links 3-5, 3-A, and A-5) belong to the loop-formation?
3. How is the appropriate value for θ computed (automatically)?
4. For those links that should belong to the loop-formation (such as links 3-5, 3-A, and A-5), how is it determined that the value of θ should be "ADDED" or "SUBTRACTED"?

To answer the previous questions, the following terms are defined:

- A "dead node" is defined as a node which is connected by only one or no basic link (solid line in the figures), and
- A "dead basic link" is defined as a basic link (solid line) which is connected to a "dead node"; furthermore, "dead nodes" and any associated "dead links" can be DELETED (ignored or discarded) during the process of finding the θ value.

Based on the above definitions, and using Fig 4 (focusing on basic links only), one concludes that (see Fig 5):

- Node #1 is a "dead node" (because there is only one basic link 1-A connected to node #1), hence node #1's associated "dead link 1-A" can be DELETED.
- Node #2 is a "dead node" (because there is only one basic link A-2 connected to node #2), hence node #2's associated "dead link A-2" can be DELETED.
- Node #4 is a "dead node" (because there is only one basic link A-4 connected to node #4), hence node #4's associated "dead link A-4" can be DELETED.

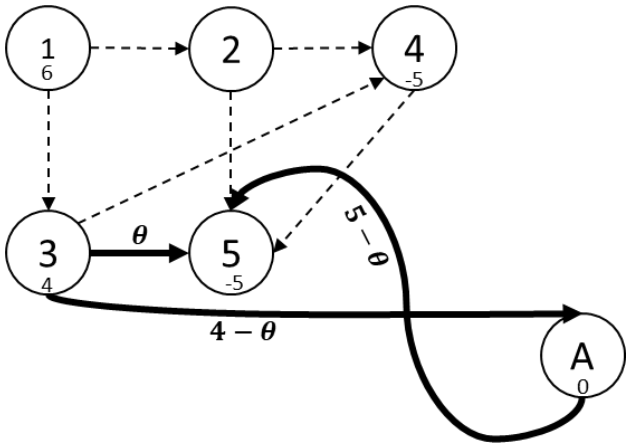

Fig 5. Dead Links Deleted

Note:

- The remaining nodes (#3, #5, and #A), and the remaining basic links (3-A, A-5, and 3-5) will form the closed loop-formation needed to calculate the value of θ.
- All nodes that belong to the closed loop-formation will be connected (or surrounded) by at most two basic links (i.e., node #3 has link 3-5 and link 3-A; no others).

The following algorithm can be developed to automatically identify the closed loop-formation and compute θ value (in Step #3):

Step #3.1 Identify all the "dead nodes".

Step #3.2 Identify and DELETE all the "dead basic links".

Step #3.3 In Step #2, it was determined that the non-basic link 3-5 will "enter the basic group"; node #3 is defined as the "loose node #i" and node #5 is defined as the "loose node #j".

Step #3.4 The "link status" for the "loose node #i = loose node #3" can be defined as one of the following four possibilities as detailed in Fig 6:

Link Status = 1: "out more",
(Link Status = 1 is selected see Fig 6)
Link Status = 2: "out less",
Link Status = 3: "in more", or
Link Status = 4: "in less"

Once the status for the current "loose node #i" is updated, the new "loose node #i" shifts to the next connected node, in this case node #A.

Step #3.5 The "link status" for the "loose node #j = loose node #5" can be defined as one of the following four possibilities as detailed in Fig 6:

Link Status = 1; "out more",
Link Status = 2; "out less",
Link Status = 3; "in more", or
(link Status = 3 is selected see Fig 6)
Link Status = 4; "in less"

Once the status for the current "loose node #j" is updated, the new "loose node #j" shifts to the next connected node, in this case node #A.

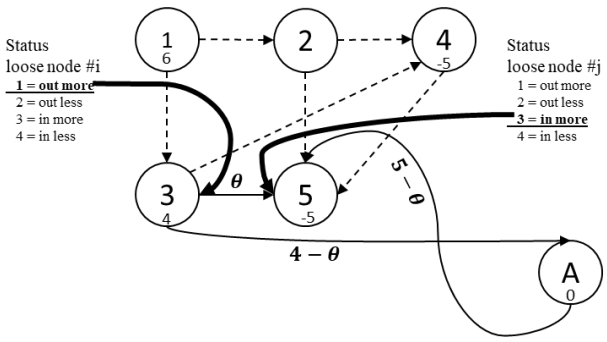

Fig 6. Status of Loose Nodes #i and #j

Step #3.6 In Fig 7, a basic link which is connected to the "loose node #i = loose node #3" can be EITHER case #1 OR case #4.

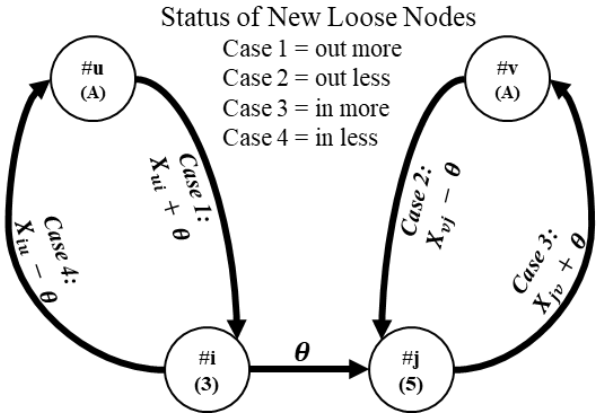

Fig 7. General Case for Status Selection

In **case #1**, if the basic link u-i is connected **TO** a loose node #i (at this point node #3), then the flow for link u-i is $X_{ui} + \theta$ (i.e., flow on u-i), so that the loose node #i will maintain its equilibrium. Under this scenario, the UPDATED loose node #i becomes loose node #u (node #A in our example), and the UPDATED link status for the loose node #u is 1, which means "out more". Since the new flow (= $X_{ui} + \theta$) is required to be non-negative, one can have any non-negative value for θ.

In **case #4**, if the basic link i-u is connected **FROM** a loose node #i (at this point node #3), then flow for link i-u is $X_{iu} - \theta$ (i.e., flow on i-u), so that the loose node #i will maintain its equilibrium. Under this scenario, the UPDATED loose node #i becomes loose node #u (node #A in our example), and the UPDATED link status for the loose node #u is 4, which means "in less". Since the new flow (= $X_{iu} - \theta$) is required to be non-negative, the value of θ must be $\leq X_{iu}$ (LESS THAN or EQUAL TO X_{iu}).

Step #3.7 In Fig 7, a basic link which is connected to the "loose node #j = loose node #5" can be EITHER case #2 OR case #3.

In **case #2**, if the basic link v-j is connected **TO** a loose node #j (at this point node #5), then the flow for link v-j is $X_{vj} - \theta$ (i.e., flow on v-j), so that the loose node #j will maintain its equilibrium. Under this scenario, the UPDATED loose node #j becomes loose node #v (node #A in our example), and the UPDATED link status for the loose node #v is 2, which means "out less". Since the new flow (= $X_{vj} - \theta$) is required to be non-negative, the value of θ must be $\leq X_{vj}$ (LESS THAN or EQUAL TO X_{vj}).

In **case #3**, if the basic link j-v is connected **FROM** a loose node #j, then the flow for link j-v is

$X_{jv} + \theta$ (i.e., flow on j-v), so that the loose node #j will maintain its equilibrium. Under this scenario, the UPDATED loose node #j becomes loose node #v (node #A in our example), and the UPDATED link status for the loose node #v is 3, which means "in more". Since the new flow (= $X_{jv} + \theta$) is required to be non-negative, one can have any non-negative value for θ.

Steps #3.6 and #3.7 are repeated until the "loose #i = loose #j" and the loop is closed.

The final value for θ should be the smallest, positive value among cases #1, #2, #3, and #4, to guarantee the flows on every basic link of the closed loop will be non-negative and maintain equilibrium.

After Step #3 in each iteration is completed, Steps #1 through #3 will be repeated in subsequent iterations Fig 8 – Fig 10), until the feasible solution, for Phase 1, is identified. The cost at the end of Phase 1 is zero because all artificial links with any flow have been removed (Fig 10). (Note: the values displayed are the link flows (X_{ij}) associated with the updated θ values or the original artificial links).

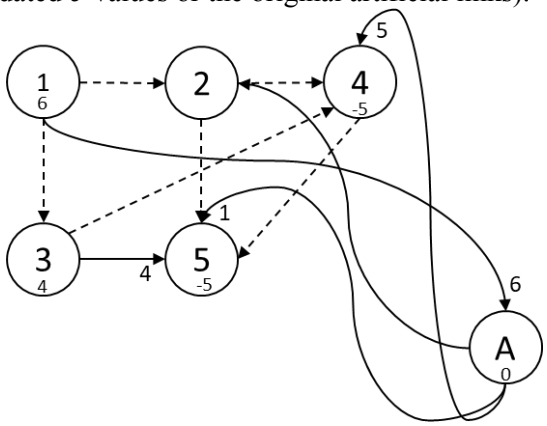

Fig 8. End of Phase 1, Iteration 1 (Cost = 12)

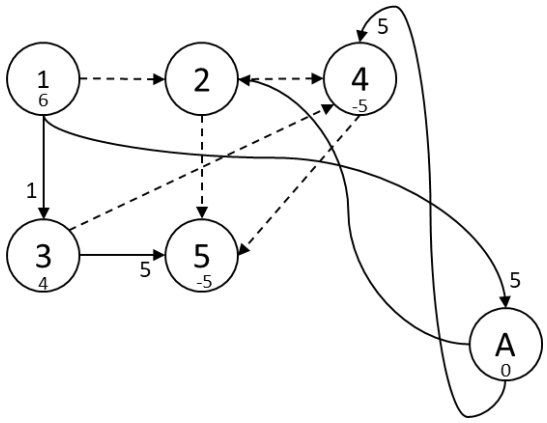

Fig 9. End of Phase 1, Iteration 2 (Cost = 10)

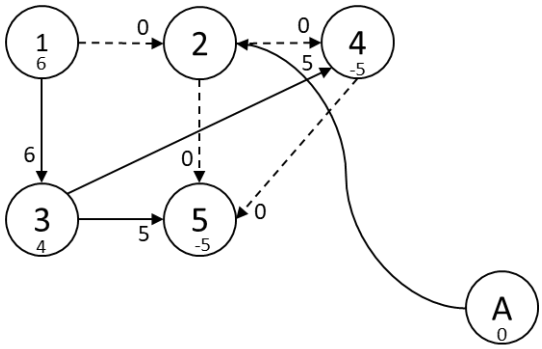

Fig 10. End of Phase 1, Iteration 3 (Cost = 0)

B. Phase 2 – Calculate the Optimal Final Solution (OFS)

The objective for Phase 2 is to find an overall, OFS for the original (in our example, Fig 1, a 5-node and 7-link) network.

The BFS obtained at the end of Phase 1 (see Fig 10) will be used as the beginning of Phase 2 (see Fig 11). In Phase 2, the original links' cost with the value C_{ij} are used, and an #th node, with the most connected links (in this case, node #3), is selected such that w_3 = known value = 0. Then, the same 3-step algorithm (steps #1, #2, and #3) discussed in Phase 1 will be applied for Phase2.

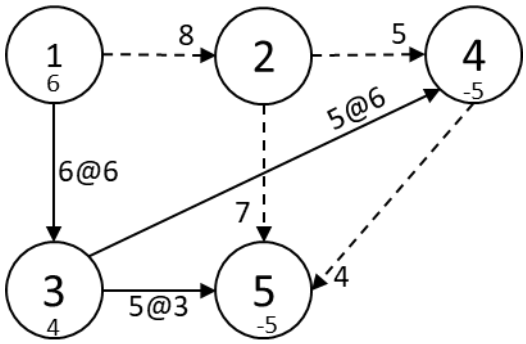

Fig 11. Feasible Solution to Start Phase 2

4 Numerical Results

Based on the 3-step MCF algorithm (applied in Phase 1 and Phase 2) presented in Section 3, and the newly proposed shortest path (SP) algorithm used in Phase 1 and presented in Section 4, the following academic and real-life networks are used to validate our proposed algorithm for identifying the closed loop-formation and computing the appropriate value for θ (see the discussions in 3.1.4, for Step #3 of Phase 1 and Phase 2). All our optimum results, for these examples, have been compared with, and validated using, the built-in MATLAB Linprog() function (Simplex algorithm) [16] and the results were recorded with each example in this section. The process times were measured over 36 trials and the average of the median 30 measurements were included in the analysis of each example.

4.1 Example #1: 5-Node/7-Link Network

The example #1 network is displayed in Fig 12. In this example the conventional MCF and Linprog() solutions produced the same optimum solution.

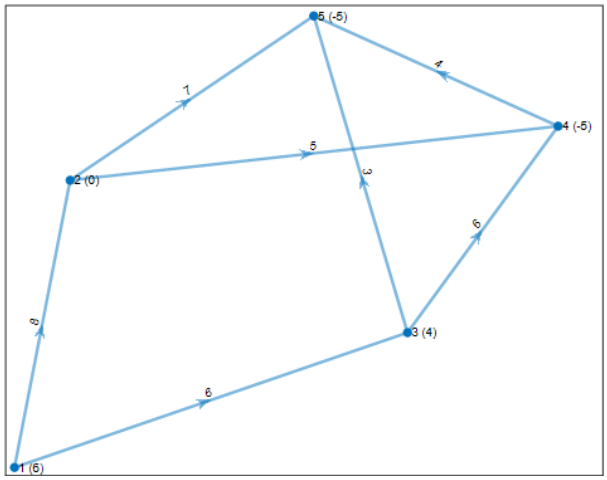

Fig 12. Original Network (5/7)

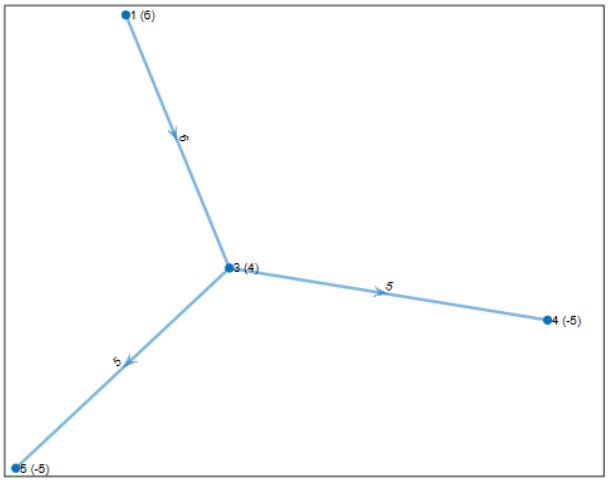

Fig 13. MCF Phase 1 (5/7) Solution = 81

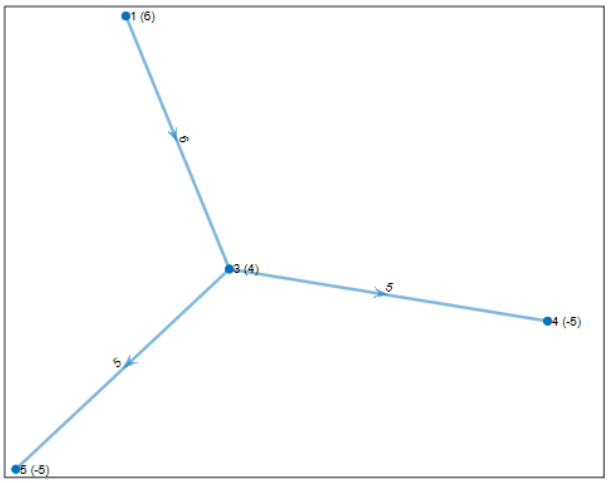

Fig 14. MCF Phase 2 (5/7) Solution = 81

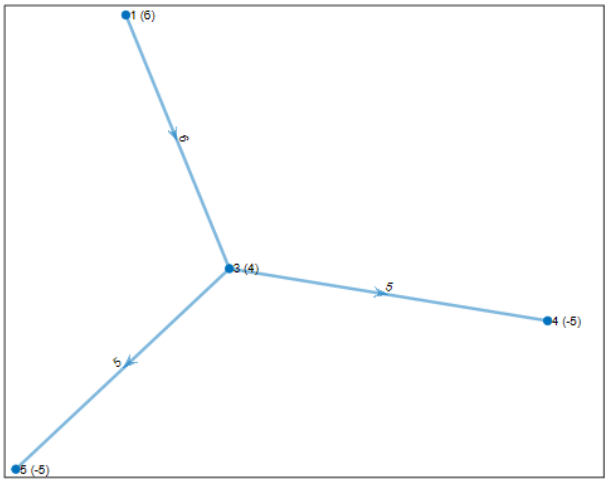

Fig 15. Linprog() Final (5/7) Solution = 81

4.2 Example #2: 9-Node/14-Link Network

The example #2 network is displayed in Fig 16. In this example, the MCF solution results in an optimal (minimum cost) value of 71 which is verified by the Linprog() solution; however, it required both Phase 1 (solution = 80) AND Phase 2 to reach the Linprog() verified solution (Fig 19).

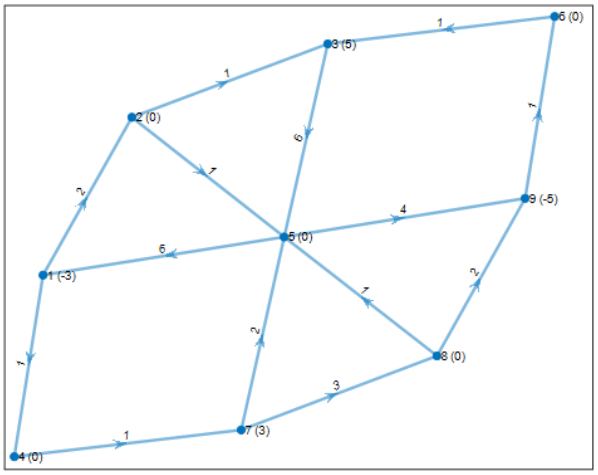

Fig 16. Original (9/14) Network

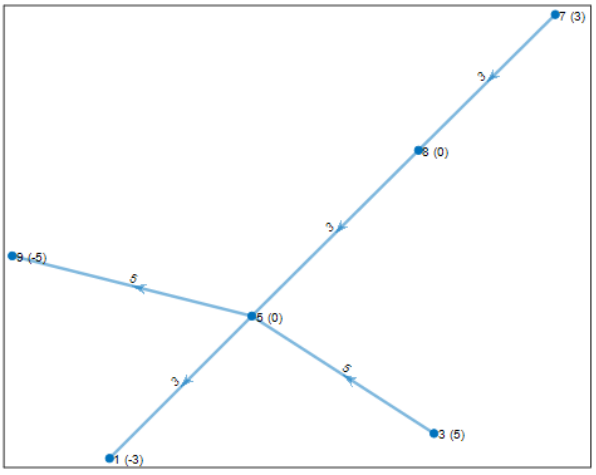

Fig 17. MCF Phase 1 (9/14) Solution = 80

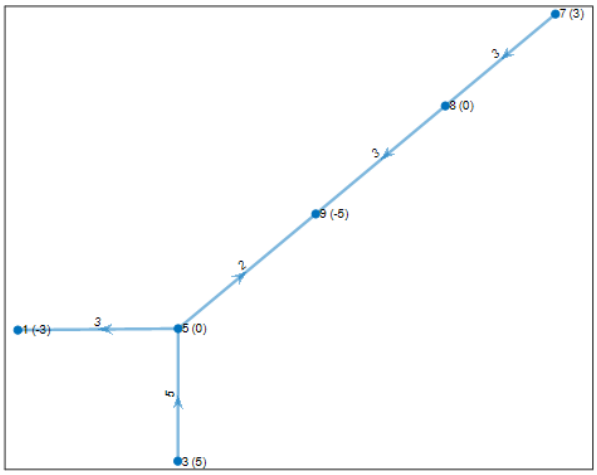

Fig 18. MCF Phase 2 (9/14) Solution = 71

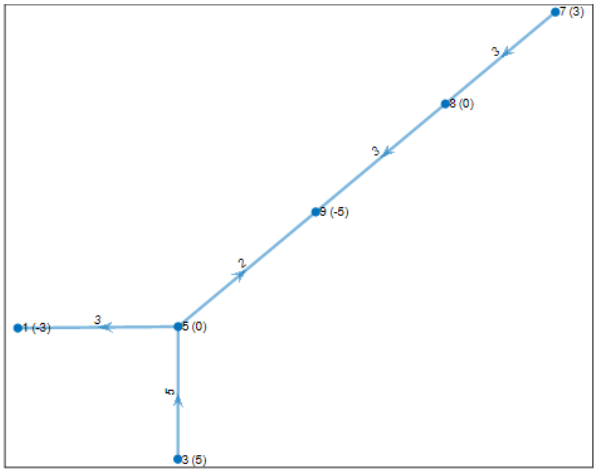

Fig 19. Linprog() Final (9/14) Solution = 71

4.3 Example #3: 20-Node/74-Link Network

The example #3 network is displayed in Fig 20. In this example, the MCF solution results in an optimal (minimum cost) value of 1775 which is verified by the Linprog() solution; however, it required both Phase 1 (solution = 2000) AND Phase 2 to reach the Linprog() verified solution (Fig 33).

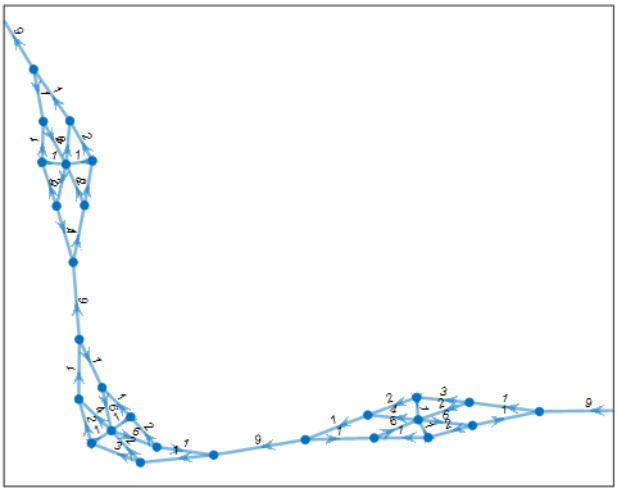

Fig 20. Portion of Original (225/375) Network

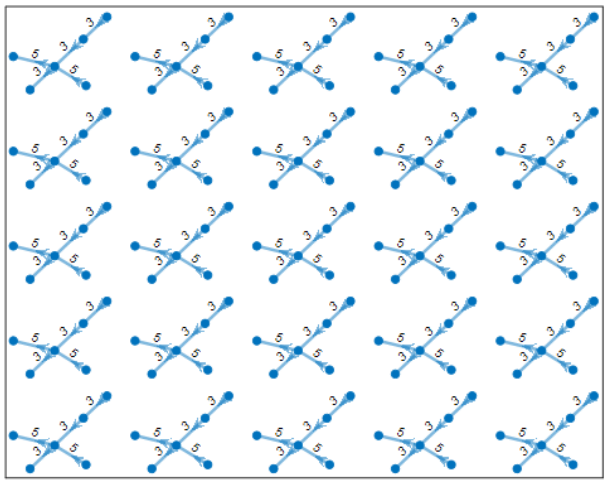

Fig 21. MCF Phase 1 (225/375) Solution = 2000

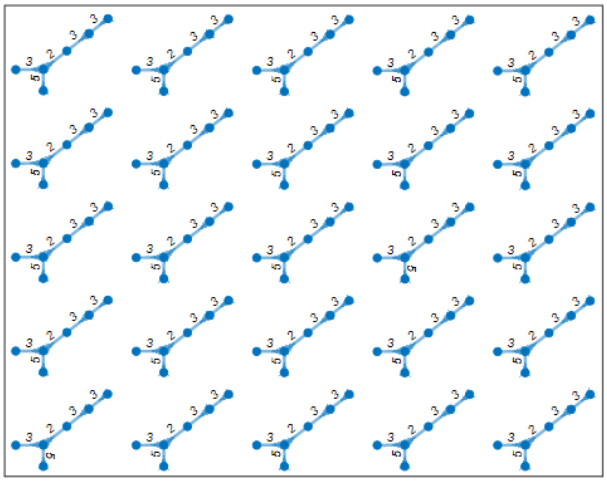

Fig 22. MCF Phase 2 (225/375) Solution = 1775

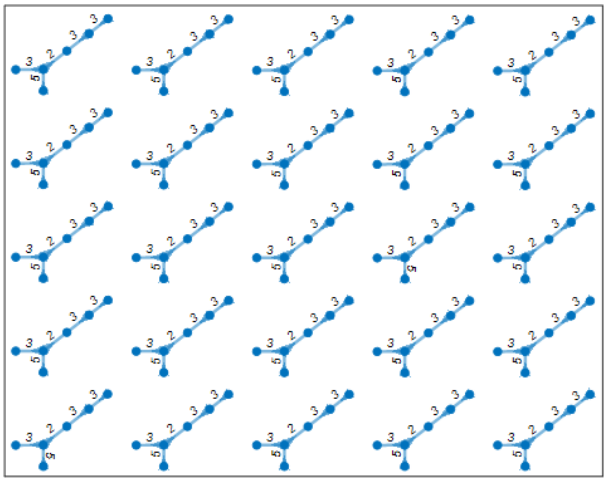

Fig 23. Linprog() Final (225/375) Solution = 1775

6.4 Example #4: 24-Node/76-Link Sioux Falls

The example #4 network is displayed in Fig 25. In this real-life, Sioux Falls (Fig 24) example, the conventional MCF method required Phase 2 to refine the feasible solution and deliver the OFS (Fig 27). However, the conventional MCF method produced an alternate optimal flow; indicating that there is more than one path that will result in the same optimal value or cost.

Fig 24 - Sioux Falls Map/Network

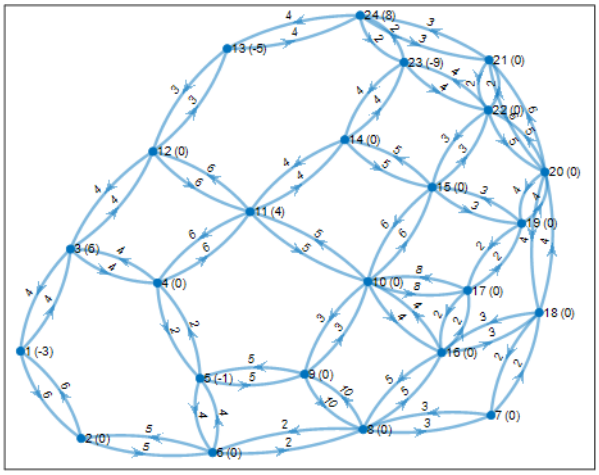

Fig 25. Original Network (24/76)

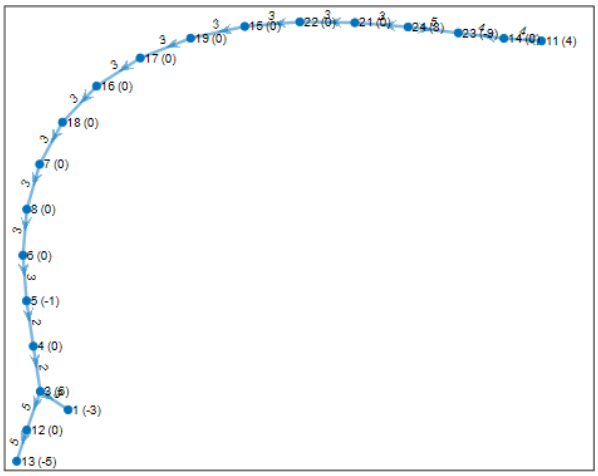

Fig 26. - (Real Life Sioux Falls) Conv MCF Phase 1 (24/76) Solution = 188

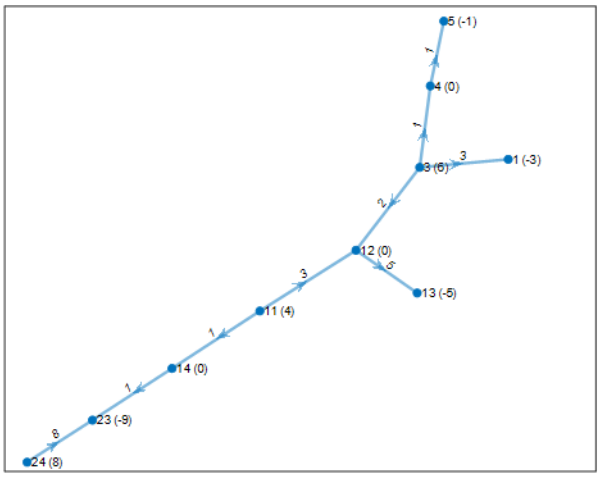

Fig 27. (Real Life Sioux Falls) Conv MCF Phase 2 (24/76) Solution = 83

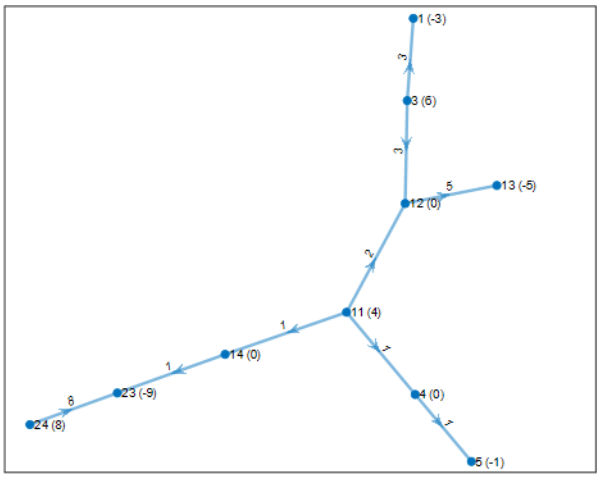

Fig 28. (Real Life Sioux Falls) Linprog() Final (24/76) Solution = 83

7 Conclusion

The most complicated step in the conventional MCF (Phase 1 and Phase 2) algorithm occurs in Step #3 (identifying the closed loop and computing the appropriate value for θ). To the authors' best knowledge, most (if not all) existing literature does NOT discuss the (automated) numerical algorithm to accomplish Step #3 of the conventional MCF method. This work has explained, in detail, the step-by-step procedures to implement the conventional MCF algorithm, with emphasis on Step #3 (see Section 3.1.4) to determine the loop-formation and value for θ.

References:

[1] R.T. Haftka and A. Gurdal, *Elements of Structural Optimization*, Kluwer Academic Publishers (Third revised and expanded edition) 1992.

[2] A.D. Belegundu and T.R. Chandrupatla, *Optimization Concepts and Applications in Engineering*, Cambridge University Press (Third edition) 2019.

[3] E.J. Haug and J.S. Arora, *Applied Optimal Design*, John Wiley & Sons 1979.

[4] D.G. Luenberger, *Introduction to Linear and Nonlinear Programming*, Addison-Wesley 1973.

[5] G.B. Dantzig, "Maximization of a Linear Function of Variables Subject to Linear Inequalities", T. C. Koopmans (Ed.), *Activity Analysis of Production and Allocation*, John Wiley & Sons 1951.

[6] G.B. Dantzig, "Linear Programming: The Story About How It Began: Some legends, a little about its historical significance, and comments about where its many mathematical programming extensions may be headed", *Operations Research*, Vol 50 (1), pp. 42-47, 2002.

[7] N. Karmarkar, "A New Polynomial Time Algorithm for Linear Programming", *Combinatorica*, Vol 4, pp. 373-395, 1984.

[8] R.G. Bland, "New Finite Pivoting Rules for the Simplex Method", *Mathematics of Operations Research*. Vol 2 (2), pp. 103–107, 1977.

[9] R.K. Ahuja, T.L. Magnanti, J.B. Orlin, *Network Flows. Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[10] M. Klein, "A Primal Method for Minimal Cost Flows with Applications to the Assignment and Transportation Problems". *Management Science*. Vol 14 (3), pp. 205–220, 1967.

[11] G. Srinivansan, [nptelhrd]. (2010, January 10). "Lec-23 Minimum Cost Flow Problem" [Video], https://youtu.be/UtSrgTsKUfU.

[12] G. Srinivansan, [nptelhrd]. (2009, August 31). "Lec-13 Transportation Problems" [Video], https://youtu.be/Q31jKiEXxdc.

[13] G. Srinivansan, [nptelhrd]. (2010, January 28). "Lec-20 Shortest Path Problem" [Video], https://youtu.be/wVu5qN0E4ww.
[14] G. Srinivansan, [nptelhrd]. (2009, August 31). "Lec-16 Assignment Problem - Hungarian Algorithm" [Video], https://youtu.be/BUGIhEecipE.
[15] G. Srinivansan, "Re: Question for Subject Matter Expert on Minimal Cost Flow" Personal Email (2020, October 13).
[16] "Linprog()," Solve linear programming problems - MATLAB, 1994. [Online]. Available: https://www.mathworks.com/help/optim/ug/linprog.html. [Accessed: March 15, 2021].
[17] 18. Transportation Networks for Research Core Team. Transportation Networks for Research. https://github.com/bsTabler/TransportationNetworks. [Accessed: March 15, 2021].

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.