An Improved Algorithm of Juang's Method by Matrix Operations for Finding the Shortest Path in Networks

Fu Sen F. Lin and Cheng-Han Lee

***Abstract*—A new algorithm of finding the shortest path in networks by matrix operations and linear combinations of paths is presented in this paper. In 2007, a novel method for solving the shortest path problem was proposed by Juang. Juang's algorithm is based on the concept of Kruskal's algorithm associated with RREF processing (Gauss-Jordan elimination) and a matrix transformation. However, Juang's scheme only can work in some special well-connected networks. We therefore design a new approach by following the basic steps of Juang's algorithm and adding the linear combination of fundamental cut-sets. The presented algorithm (called JLL-SP algorithm) can overcome the failure cases of Juang's method and work correctly and efficiently for wider range of networks. The experimental results verified that the JLL-SP algorithm performs well in most unilaterally connected digraphs.**

***Index Terms*—Connected digraphs, linear transformation, linear combination, RREF matrix, shortest path.**

I. Introduction

Many problems in real world can be modeled as a network, such as transportation analyses, communication webs, or computer networks [1]-[4]. A network is a digraph (directed graph) containing two distinct vertices, say *s* and *t*, called starting vertex (or source, or initial node), and destination vertex (or sink, or terminal node), respectively. Suppose that every arc (i, j) in a network has a weight, which may represent the distance (or cost, or capacity) from vertex *i* to vertex *j*. Finding the shortest path between *s* and *t* is one of the most important problems in optimization or graph theory.

The most famous method for finding the shortest path is proposed by Edsger W. Dijkstra in 1959 [5]. Over many decades, a lot of schemes were proposed in order to solve this problem efficiently [6]-[11]. For instances, G. Gallo and S. Pallottino in 1988 presented a remarkable contribution in the implementations and comparisons of eight shortest path algorithms with different data structures [8]. F. B. Zhan and C. E. Noon ever gave a benchmark to evaluate the performance of 15 shortest path algorithms using real road networks in 1996 [10]. Later, Zhan identified out three fastest shortest path algorithms with the same experiments [11]. Within the three fastest shortest path algorithms, two of them are based on Dijkstra's algorithm with different implementation data structures. It is evident that Dijkstra's algorithm is efficient and popular.

In 2007, J. Y. Juang proposed a new method to compute the shortest path by finding the minimum spanning tree first and following a matrix transformation [12]-[14]. Juang's method is based on the concept of Kruskal's algorithm associated with matrix operations. This method is applicable, but it only can work in a narrow set of connected networks. We therefore present a new algorithm in this paper to improve Juang's method. Our approach executes the same basic steps of Juang's algorithm and then applies the linear combination of paths to find all feasible paths from *s* to *t*. The lowest weight of paths will emerge and the shortest path will be obtained. According to our experiments, the presented algorithm can perform well for most unilaterally connected digraphs.

The rest of this paper will be organized as follows. First of all, we shall give some basic definitions and terminology. Next, we shall briefly introduce the method of Juang. In the fourth section our algorithm will be presented and the experimental results will be shown in the fifth section. Finally, a concluding remark will be addressed and references will be given.

II. Preliminaries

A digraph, denoted as $G = (V, E)$, is a finite nonempty set V of objects called vertices (also called points or nodes) and a set E of ordered pairs of distinct vertices called arcs (or edges). The number of vertices is called the order of G, denoted as $|V|$, and the number of arcs is called the size of G, denoted as $|E|$.

For a connected digraph $G = (V, E)$, we assume that $|V| = n$ and $|E| = m$. In order that there is at least one path from *a* to *z* we require that *n* must be less than or equal to $m+1$, i.e., $n \le m+1$. Let (u, v) be an arc in V, that means, there is an edge from *u* to *v*.

For a given network, in order to assure that there exists at least one path from starting vertex to destination vertex, we shall restrict the underlying digraph *G* is unilaterally connected.

A digraph *G* is unilaterally connected if for every two distinct vertices *u* and *v* of *G* there exists either a $u \rightarrow v$ path, or $v \rightarrow u$ path, or both. If both $u \rightarrow v$ path and $v \rightarrow u$ path exist for every pairs *u* and *v* of *G*, then *G* is strongly connected.

For more details of digraphs, we refer to the books of Chartrand and Oellermann [1], or Winston [3], or Corman *et al*. [4].

III. Juang's Methods

Juang proposed two algorithms successively, one for finding the minimum spanning tree (MST) in 2006 [12] and

Manuscript received August 15, 2014; revised February 10, 2015.

The authors are with the Department of Computer Science and Engineering at National Taiwan Ocean University, Keelung, 20224 Taiwan (e-mail: fslin@ntou.edu.tw, wilson_1886@hotmail.com).

DOI: 10.7763/IJCTE.2016.V8.1032

the other for searching the shortest path in a digraph in 2007 [13], [14]. However, after our further investigation, his shortest path method only work for some connected digraph and will fail in some cases. We briefly state his basic operations:

A. Juang's Algorithm for MST

Many methods have been proposed for finding the minimum spanning tree. The most famous schemes are Kruskal's algorithm [15], Prim's algorithm [16], and Collin's algorithm which is the combination of both concepts [17]. The main idea of Juang's approach is based on the concept of searching the MST by Kruskal's algorithm and applies the elementary row operations, usually called RREF processing or Gauss-Jordan elimination, to the created matrix and a matrix transformation to obtain the fundamental cut-sets.

Given a unilaterally connected weighted digraph G (V, E), first of all, we suppose that the vertices V has been arranged by a sequence say {$a, b, c, \ldots, x, y, z$} with $s = a$ and $t = z$ and that the arcs are sorted in non-decreasing order of the weight of each arc. Thus, we can create an incidence matrix so that each vertex is corresponding to one row and each arc is corresponding to one column. That is, the *incidence matrix* $M = \left[M_{ij} \right]_{n\times m}$ is defined by

$$M_{ij} = \begin{cases} 1, & \text{if there is an arc } (i, j) \text{ from vertex } i \text{ to } j; \\ 0, & \text{if there is no arc } (i, j) \text{ or } (j, i); \\ -1, & \text{if there is an arc } (j, i) \text{ from vertex } j \text{ to } i. \end{cases} \quad (1)$$

Here, the row i is corresponding to the vertex i in V, but the column j is the j-th arc in the weighted order set of arcs, i.e., $j \in \{1, 2, \cdots, m\}$. Therefore, one can create the *flow table* of the given network. We now apply the Gauss-Jordan elimination to the matrix M in the *flow table* and then obtain a *reduced row echelon-form* (RREF) matrix. The last row of RREF matrix should be all zero and can be ignored. Therefore, the resulting matrix denotes as S remains n-1 rows. If the first n-1 columns do not form an identity matrix, we rearrange the columns where the row has the leading's 1 so that the first n-1 columns form an identity matrix I_{n-1}. That means, the resulting matrix will be the form

$$\left[I_{n-1} \mid F \right]_{(n-1)\times m} := S \quad (2)$$

Each row of S represents an independent (or fundamental) cut-set. The corresponding arcs of the first n-1 columns of matrix S will constitute the MST of the given digraph, since every arc in MST has the smallest weight in fundamental cut-set. This is Juang's algorithm for finding the MST of a given digraph. We refer it as JMST algorithm and summarize it as follows.

Algorithm of JMST:

1) Create the incidence matrix M by Eq.(1) of a given digraph G to form the *flow table* of G.
2) Apply the Guass-Jordan Elimination to the incidence matrix M and obtain the RREF Matrix.
3) Rearrange the columns so that it becomes the form of Eq.(2), if the first n-1 columns do not form I_{n-1} matrix.

The MST of the given digraph will be composed of the corresponding arcs of the columns of submatrix I_{n-1}. Notice that the MST of a digraph may not be an actual tree structure (with one root).

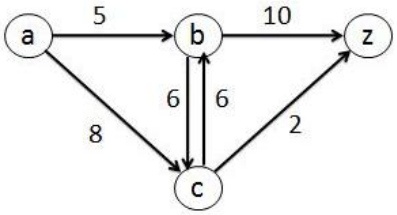

Fig. 1. A weighted digraph of network model.

We now give an example to illustrate this algorithm. Consider the weighted digraph G_1 shown in Fig. 1. We first create its incident matrix to form the *flow table* of G_1, shown in Table I. In this table, the word 'wts' represents the weight of each arc and '*cz*' represents the arc (c, z) from vertex c to vertex z.

TABLE I: THE FLOW TABLE OF G_1

wts	2	5	6	6	8	10
arcs	*cz*	*ab*	*bc*	*cb*	*ac*	*bz*
a	0	1	0	0	1	0
b	0	-1	1	-1	0	1
c	1	0	-1	1	-1	0
z	-1	0	0	0	0	-1

We apply the Gauss-Jordan elimination to the incidence matrix and obtain the RREF matrix S. According to JMST algorithm's step 3, we obtain MST of G_1 as the set {*cz*, *ab*, *bc*}.

TABLE II: THE RESULTING MATRIX S AND CORRESPONDING ARCS

2	5	6	6	8	10
cz	*ab*	*bc*	*cb*	*ac*	*bz*
1	0	0	0	0	1
0	1	0	0	1	0
0	0	1	-1	1	1

B. Juang's Algorithm for the Shortest Path

Later, Juang also proposed an algorithm of finding the shortest path for a connected digraph [14]. We call it as JSP algorithm. This approach is based on searching the MST and a matrix transformation. If the MST of a connected digraph contains a path from the starting vertex a to the destination vertex z, then we append a *hypothetical link* (or virtual arc), *za*, from z to a with zero weight, to obtain a cycle passing through the a and z vertices. The first part of JSP algorithm is the same as JMST algorithm, except that the virtual arc *za* is appended to the *flow table* and *incidence matrix*. The second part of this algorithm, a matrix transformation is performed. The transformation is defined by

$$S = [I_{n-1} \mid F_{(n-1)\times(m-n+1)}] \rightarrow \left[-F^T \mid I_{m-n+1} \right] := T \quad (3)$$

From the rows of matrix T, we can identify the feasible paths from a to z and select the lowest weight of each feasible path to obtain the shortest path. This algorithm is summarized as follows.

Algorithm of JSP:

Give a unilaterally connected weighted digraph G.

1) Create the incidence matrix M by Eq. (1) with appending a virtual arc za with zero weight to form the *flow table* of G.
2) Apply the Gauss-Jordan Elimination to the incidence matrix and obtain the RREF Matrix.
3) Rearrange the columns so that it becomes the matrix as Eq.(2), if the first n-1 columns do not form I_{n-1}.
4) Do a matrix transformation as Eq. (3) obtain matrix T.
5) Identify the first column of T with entry 1 rows, which may represent a path from a to z with the corresponding arcs to the entry-1 columns, but neglect the virtual arc za.
6) Compute the total weights of each path and select the minimum one.

The shortest path of the given graph G is therefore obtained.

Similarly, we use the same graph as Fig. 1 to illustrate the JSP algorithm. The *flow table* including the incidence matrix of G_1 is shown in Table III. After the processes of step 2 and step 3, the resulting matrix S is displayed in Table IV.

TABLE III: THE FLOW TABLE OF G_1 BY JSP ALGORITHM

0	2	5	6	6	8	10
za	*cz*	*ab*	*bc*	*cb*	*ac*	*bz*
-1	0	1	0	0	1	0
0	0	-1	1	-1	0	1
0	1	0	-1	1	-1	0
1	-1	0	0	0	0	-1

TABLE IV: THE MATRIX S OF G_1 BY JSP ALGORITHM

0	2	5	6	6	8	10
za	*cz*	*ab*	*bc*	*cb*	*ac*	*bz*
1	0	0	-1	1	-1	-1
0	1	0	-1	1	-1	0
0	0	1	-1	1	0	-1

According to the step 4 of JSP algorithm, a matrix transformation is made and the matrix T is obtained, and shown in Table V. One can identify the first column of T with entry 1 rows to obtain 3 paths from a to z. The total weights of these paths are also shown in the last column of Table V. Where 'WP' represents the total weight of each path and 'NP' represents no paths. One can see that the shortest path of G_1 comes from the third row and is corresponding to the set {*ac*, *cz*} with total weight 10 (neglecting the virtual arc *za*).

TABLE V: THE MATRIX T OF G_1 BY JSP ALGORITHM

0	2	5	6	6	8	10	WP
za	*cz*	*ab*	*bc*	*cb*	*ac*	*bz*	
1	1	1	1	0	0	0	13
-1	-1	-1	0	1	0	0	NP
1	1	0	0	0	1	0	10
1	0	1	0	0	0	1	15

C. The Presented Algorithm

We selected many digraphs with different shortest paths to implement JSP algorithm and found that it may fail in some cases. The problem is that the row vectors of the transformed matrix T may not correspond to a path from a to z. That means, some of the feasible paths do not emerge in the matrix T.

Therefore we found that the linear combination of the row vectors of the matrix T is an attainable strategy. Our idea is to split the row vectors of T into two sets. One with the first entry 1 (corresponding to virtual arc za) and the other with 0, denoted as T_1 and T_0, respectively. The row with the first entry -1 are useless and we ignore them. If a row vector with the first entry to be 1 and no entry to be -1 at the other places, then the corresponding arcs of the nonzero columns in this row form a cycle. Therefore, when the virtual arc za is removed, it become a path from a to z. If there exists -1 in this row, it will form a (short) cycle in the row, and then it cannot form a path when the virtual arc za is removed.

We now execute the linear combination by picking one vector from T_0 with some entry to be -1 and one vector from T_1 with the entry to be 1 in the same place, and adding them together. The resulting vector should be put in the set T_1. After the process of linear combination, we shall have a new T_1 set and the nonzero columns of most rows in T_1 will correspond to a path from a to z. Consequently, all of the feasible path will be displayed in T_1 set. Comparing their total weights, we can obtain the shortest path from a to z. This is our advanced algorithm contrast to JSP algorithm. We refer it as JLL-SP algorithm (The algorithm of Juang, Lin, and Lee for the shortest path.).

Algorithm of JLL-SP:

1) Pick every row vector from T_0 with some entry to be -1 and a vector from T_1 with the entry to be 1 in the same place and add them together and put it in the set T_1.
2) Identify each row of T_1 without entry -1 and thus the corresponding arcs of entry 1 form a path.
3) Compute the total weights of each path and compare their weights.

The path with the lowest weight will be the shortest path of the given graph.

IV. EXPERIMENTAL RESULTS

We have tested many different types of networks with the presented JLL-SP algorithm. This algorithm performs correctly and efficiently for most cases.

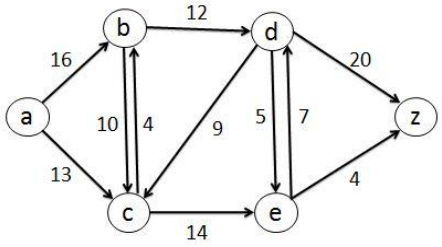

Fig. 2. A digraph of network model.

TABLE VI: THE FLOW TABLE OF G_2

0	4	4	5	13	7	9	10	12	14	16	20
za	*cb*	*ez*	*de*	*ac*	*ed*	*dc*	*bc*	*bd*	*ce*	*ab*	*dz*
-1	0	0	0	1	0	0	0	0	0	1	0
0	-1	0	0	0	0	0	1	1	0	-1	0
0	1	0	0	-1	0	-1	-1	0	1	0	0
0	0	0	1	0	-1	1	0	-1	0	0	1
0	0	1	-1	0	1	0	0	0	-1	0	0
1	0	-1	0	0	0	0	0	0	0	0	-1

We now select a more complicated example shown in Fig. 2, say G_2, to illustrate the JLL-SP algorithm. The first step of the JLL-SP algorithm is to execute the steps 1 to 4 of JSP algorithm to the graph G_2. We therefore obtain the *flow table* (Table VI), matrix S (Table VII), and matrix T (Table VIII),

sequentially.

TABLE VII: THE MATRIX S OF G_2 BY JLL-SP ALGORITHM

0	4	4	5	13	7	9	10	12	14	16	20
za	*cb*	*ez*	*de*	*ac*	*ed*	*dc*	*bc*	*bd*	*ce*	*ab*	*dz*
1	0	0	0	0	0	1	0	-1	-1	0	0
0	1	0	0	0	0	0	-1	-1	0	1	0
0	0	1	0	0	0	1	0	-1	-1	0	1
0	0	0	1	0	-1	1	0	-1	0	0	1
0	0	0	0	1	0	1	0	-1	-1	1	0

TABLE VIII: THE MATRIX T OF G_2 BY JLL-SP ALGORITHM

0	4	4	5	13	7	9	10	12	14	16	20
za	*cb*	*ez*	*de*	*ac*	*ed*	*dc*	*bc*	*bd*	*ce*	*ab*	*dz*
0	0	0	1	0	1	0	0	0	0	0	0
-1	0	-1	-1	-1	0	1	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0
1	1	1	1	1	0	0	0	1	0	0	0
1	0	1	0	1	0	0	0	0	1	0	0
0	-1	0	0	-1	0	0	0	0	0	1	0
0	0	-1	-1	0	0	0	0	0	0	0	1

Observing the matrix T (Table VIII), one can see that the JSP algorithm fails for the network G_2. It cannot obtain the shortest path from the row vectors of T. Therefore, we apply the step 2~5 of JLL-SP algorithm to the matrix T. That is, the matrix T is split into two sets and the linear combination of the row vectors of T is performed. The resulting matrix (final T_1) is displayed in Table IX. One can see that each row of T_1 represents one feasible path from a to z. Their total weights from the top to the bottom are 38, 31, 37, 44, 48, 49, 54, and 67, respectively. The smallest weight is 31, which is the second row of T_1 and the corresponding shortest path is the set {*ac*, *ce*, *ez*}(The virtual arc *za* is neglected).

TABLE IX: MATRIX T_1 (AFTER LINEAR COMBINATION)

0	4	4	5	13	7	9	10	12	14	16	20
za	*cb*	*ez*	*de*	*ac*	*ed*	*dc*	*bc*	*bd*	*ce*	*ab*	*dz*
1	1	1	1	1	0	0	0	1	0	0	0
1	0	1	0	1	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	1	0
1	0	1	0	0	0	0	1	0	1	1	0
1	0	0	0	0	0	0	0	1	0	1	1
1	1	0	0	1	0	0	0	1	0	0	1
1	0	0	0	1	1	0	0	0	1	0	1
1	0	0	0	0	1	0	1	0	1	1	1

V. CONCLUSION

In this paper, we have presented an advanced algorithm, called JLL-SP algorithm, to improve the drawback of JSP algorithm. The new JLL-SP can perform correctly, contrast to the JSP algorithm, for wider range of networks.

One may wonder that applying matrix operations should increase the complexity of computation. It is true if we merely consider the order of the arithmetic operations. However, the RREF processing or every matrix operations are just working on the numbers 1, 0 and -1. The built-in function *rref.m* in MATLAB performs rapidly in low label (or machine) language. To evaluate the complexity of the JLL-SP algorithm and to compare with three fastest shortest path algorithms presented in [11] will be put in our future work.

Acknowledgment

The authors are grateful to professor J. Y. Juang for his invaluable suggestions. We wish that he has a wonderful retirement life.

REFERENCES

[1] G. Chartrand and O. R. Oellermann, *Applied and Algorithmic Graph Theory*, McGraw-Hill, 1993.
[2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms and Applications*, Englewood Cliffs, NJ: Prentice Hall, 1993.
[3] W. L. Winston, *Operations Research: Applications and Algorithms*, 3rd ed. Duxbury Press, CA, 1994, pp. 394-459.
[4] T. H. Corman, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, McGraw-Hill, 2001.
[5] E. W. Dijkstra, "A note on two problems in connection with graphs," *Numeriche Mathematik*, vol. 1, pp. 269-271, 1959.
[6] R. B. Dial, "Algorithm 360: Shortest path forest with topological ordering," *Communications of the ACM*, vol. 12, pp. 632-633, 1969.
[7] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithm," in *Proc. 25th Annual Symposium on Foundations of Computer Science*, pp. 338-346. 1984.
[8] G. Gallo and S. Pallottino, "Shortest paths algorithms," *Annals of Operations Research*, vol. 13, pp. 3-79, 1988.
[9] B. V. Cherkassky, A. V. Goldberg, and T. Radzik, "Shortest paths algorithms: Theory and experimental evaluation," Technical Report, Computer Science Department, Stanford University, pp. 93-1480, 1993.
[10] F. B. Zhan and C. E. Noon, "Shortest path algorithms: An evaluation using real road networks," *Transportation Science*, vol. 32, no. 1, pp. 65-73, 1996.
[11] F. B. Zhan, "Three fastest shortest path algorithms on real road networks: Data structures and procedures," *Journal of Geographic Information and Decision Analysis*, vol. 1, no. 1, pp. 69-82, 1997.
[12] J. Y. Juang, "Numerical method for the solutions of min-cut max-flow problem," in *Proc. 2006 CACS Automatic Control Conference*, St. John's University, Tamsui, Taiwan, 2006, pp. 111-115.
[13] J. Y. Juang, "Numerical method for the solutions of shortest-path problems," in *Proc. CACS Automatic Control Conference*, National Chung Hsing University, Taichung, Taiwan, 2007.
[14] J. Y. Juang, "Numerical method to the solution of optimization problems in network model analysis," in *Proc. 2008 CACS Automatic Control Conference*, National Cheng Kung University, Tainan, Taiwan, 2008.
[15] J. B. Kruskal, "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical Society*, vol. 7, no. 1, pp. 48-50, 1956.
[16] R. C. Prim, "Shortest connection networks and some generalizations," *BSTJ*, vol. 36, no. 6, November 1957.
[17] M. Sollin, "Le trace de canalization," *Programming, Games, and Transportation Networks*, 1965.

Fu Sen F. Lin received the M.S. and Ph.D. degrees in the Mathematics Department in 2000 and 2003, respectively from Oregon State University, Oregon, USA. He is currently an assistant professor in the Department of Computer Science and Engineering at National Taiwan Ocean University, Keelung, Taiwan. His research interests include network models, matrix computations, and computational complex analysis.

Cheng-Han Lee received the B. S. degree in computer science in 2011 from Tatung University, Taipei, Taiwan and M.S. degree in computer science and engineering in 2014 at National Taiwan Ocean University, Keelung, Taiwan. He currently works for military service. His research interests are network models and scientific computing.