Application of Floyd-Warshall Iterative Algorithm for Travelling to the Most Popular Tourist Attractions in Myanmar

Sanda San[#1], San San Maw[*2], Lin Lin Naing[#3]

[#]*Faculty of Computing, University of Computer Studies, Hinthada, Ayeyarwady Region, Myanmar*
[*]*Department of Engineering Mathematics, Mandalay Technological University, Mandalay, Myanmar*

Abstract *— In this paper, well-known Floyd-Warshall algorithm in graph theory isapplied to find the shortest path between two places. This algorithm is an All-Pairs-Shortest-Path (APSP) algorithm and can solveand optimize the efficient solution for the finding the shortest path between all pairs of vertices by taking every road as an edge and every vertex as city or country. Itmakes a table of all the distances between cities on a road map. It also uses a recursive approach to find the shortest distances and minimum time between all nodes in a graph. The goal of this paper is to find the effective ways in travelling for top seven tourists' attraction places in Myanmar by means of optimizing both distances and time taken between every pair of places using Floyd Warshall iterative algorithm.*

Keywords *— Shortest path, Floyd-Warshall, distance matrix and node sequence matrix, iterative algorithm, tourism.*

I. INTRODUCTION

Tourism industry is being recognized as one of the world's largest industries of revenue generation both in developed and developing countries.Tourism is travel for pleasure or business and may be domestic or international. Tourism has become an important source of income for many regions and even entire countries.Myanmar is a developing country and tourismis also a developingsector. It possesses great tourist potential and attractions in many fields. More than 1 million foreign tourists visited in 2012. The number of foreign arrivals reached more than 2.04 million, counting both air and overland arrivals in 2013.The Tourism Master Plan was created in 2013, targeting 7.5 million arrivals by 2020.In order to reach target arrivals, we need to upgrade the roads and places where tourists are attracted.The most popular tourist destinations in Myanmar are the big cities suchas Yangon and Mandalay. According to thereligious sites, there are Mon State,Pindaya, Kyaiktiyo and Hpa-An. Fornature trails, there are PyinOoLwin, InleLake, Kalaw, and Kengtung. As the ancient cities such as Bagan and Mrauk-U are also the most popular places in Myanmar as wellas beaches in Ngapali,MaungmaganandNgwe-Saung[1].Among these destinations, in this paper, we only focus for effective travelling to top seven places by using the Floyd-Warshall algorithm. It is an efficient iterative algorithm to find all-pairs shortest paths on a graph and it is guaranteed to find the shortest path between every pair of vertices in a graph.The algorithm iterates and updates the matrices, distance matrix and sequence matrix, in each iteration step. At the end of all the iterations, there exist two matrices keeping the shortest distances and shortest pathsfor all the vertices [2].

II. RESOURCES AND METHOD

The Floyd–Warshall algorithm was published in its currently recognized form by Robert Floyd in 1962 [2].The Floyd–Warshall algorithm compares all possible paths through the graph between each pair of vertices [3].The algorithm takes advantage of two matrices: The *D* matrix includes the distance between each connected node-pair and the *S* matrix includes the sequential relationship between nodes. The algorithm iterates and updates the matrices in each iteration step. At the end of all the iterations, these two matrices keep the shortest distance and neighbourhood information about all the nodes [4].

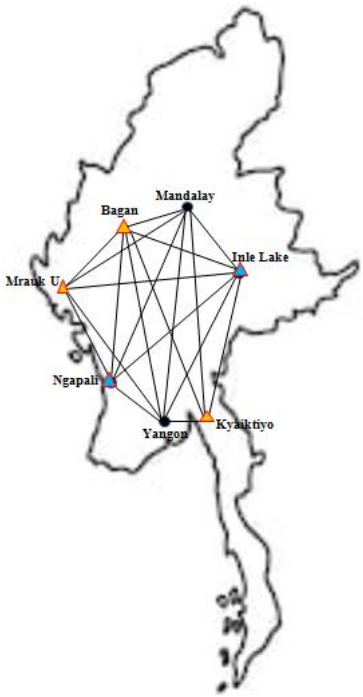

Fig.1 Representation for the most popular tourist attraction places in Myanmar by a connected graph.

A. Problem Definition

With so much competition in every aspect of life, no one ever wanted to waste it especially in travelling [5].The Floyd Warshall Algorithm can beapplied to solve the All-Pairs-Shortest-Path problems for optimal routing. The problem is to find shortest distances between every pair of vertices

according to the given edge between the distance of the mixed Graph (i.e., with both directed and undirected edges). Vertices are places, cities or countries and edges are roads in connected network.When tourists came to visit in our country, they may not be familiar with all interesting destinations of our country.In this situation, an effective travelling plan is crucial for achieving their satisfaction during their tour. This paper will addresses the issue of effective travelling plan during tour, and compute the physical travel route problem as an all-to-all shortest path problem in graph [6]. In the followingmixed graphin Fig.2, the undirected edges indicate howtourists travel to-and-fro on those routes and the directed edges are based on the travelling practice of tourists attheir tourism in Myanmar.

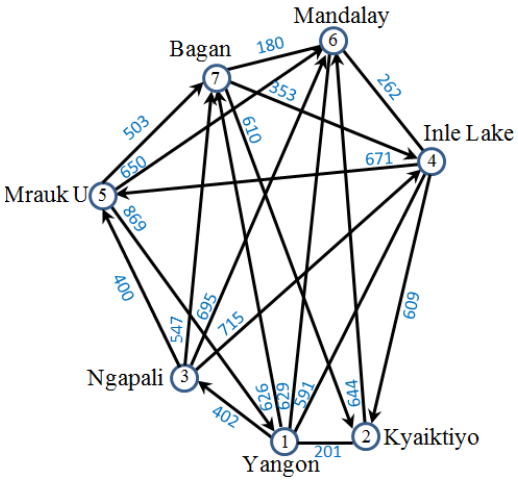

Fig. 2Representation by mixedgraph with distance (km) as weighted values.

B. Steps and Rules of Floyd-Warshall Algorithm

The Floyd-Warshall algorithm uses a dynamicprogrammingmethodologyand requires two matrices, namely distance matrix D and node sequence matrix S. It determinesthe shortest route between any two nodes or vertices in the network[7].The procedure is to calculate the matrix D of shortest-path weightsand then construct the matrix S form the Dmatrix.

Let $D^{(k)}$ be the $n \times n$matrix $\left[d_{ij}^{(k)}\right]$, where $d_{ij}^{(k)}$is the distance along the shortest pathfrom i to j. For initial step we create the initial distance matrix $D^{(0)}$and initial nodesequence matrix $S^{(0)}$ as follows [7]:

Step 1. Set all diagonal elements of $D^{(0)}$ to 0.
(i.e.,$d_{ij}^{(0)} = 0, i = 1, 2, 3, \ldots, n$)

Step 2. Set all elements of $S^{(0)}$ to zero. i.e., $s_{ij}^{(0)} = 0$for every $i, j = 1, 2, \ldots, n$.Then set $k = 1$.

Step$k$: $(1 \leq k \leq n)$.

If the condition$d_{ij}^{(k-1)} > d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$ (for $i, j = 1, 2, \ldots, n$. Provided$i \neq k\,; i \neq j$) is satisfied.

Step 3.Create $D^{(k)}$ by replacing $d_{ij}^{(k-1)}$ in $D^{(k-1)}$with $d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$.

Step 4.Create node sequence matrix $S^{(k)}$by replacing $s_{ij}^{(k-1)}$in$S^{(k-1)}$with k.Then, set$k = k + 1$, repeat step $k$until$k = n$.

After n steps, we can determine the shortest route between vertices i and j from the two matrices $D^{(n)}$ and $S^{(n)}$ using the following rules:

Rule 1. From $D^{(n)}, d_{ij}^{(n)}$ gives the shortest distance between vertices i and j.

Rule2.From $S^{(n)}$, determine the intermediate vertex $k = s_{ij}^{(n)}$which yields the route$i \to k \to j$. If$k = 0$, then stop. Otherwise, repeat the procedure between vertices $i$and k and vertices k and j.

The current iteration number is represented by the indices$k$, the row number of the matrices by i andthe column number of the matrices by j.For every new node we have to determine whether the path through new node is shorter than the previous path or not. If the path with new node is shorter, then we update the value for the shortest path.

III. RESULT AND DISCUSSION

The optimization for both distance and time taken betweenevery pair of the top seven tourists' attraction places in Myanmar has been done by using the steps and rules of Floyd-Warshall algorithm. The first distance value between two vertices is taken as infinity if they are not directly connected.Initially, the node sequence matrix is an empty matrix and its values are updatedin every part of iterations. However, the each value of columns for the shortest paths of a value in each row may be used for the initial values [2].

The algorithm iterates and generates $D^{(1)}$ and $S^{(1)}$ matrices from $D^{(0)}$ and $S^{(0)}$ matrices, $D^{(2)}$ and $S^{(2)}$ matrices from $D^{(1)}$and $S^{(1)}$matrices, and consequently,$D^{(n-1)}$and $S^{(n-1)}$ matrices from $D^{(n-2)}$and $S^{(n-2)}$ matrices. Basically, the procedure operates as follows: The iteration number k is also the index of the matrices ($D^{(k)}$and $S^{(k)}$) generated at the end of the iteration. In each iteration, the new values of $D^{(k)}$is obtained updating the $D^{(k-1)}$with respect to the nodeby index k. For instance, at the first iteration(k = 1), values in the first row and the first column of $D^{(0)}$matrix are used to update the other values in the matrix, respectively [2].Diagonal entries of each matrix will always be 0 and since, we use total of 7 places in our road network, total 7 matrices (excluding initial distance matrix) of order 7 × 7 will be seen in each iteration step [8].

Initial step: Step (0)

$$D^{(0)} = \begin{array}{c|ccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 1 & 0 & 201 & 402 & 591 & \infty & 629 & 626 \\ 2 & 201 & 0 & \infty & \infty & \infty & 644 & \infty \\ 3 & \infty & \infty & 0 & 715 & 400 & 695 & 547 \\ 4 & 591 & 609 & \infty & 0 & 671 & 262 & \infty \\ 5 & 869 & \infty & \infty & \infty & 0 & 650 & 503 \\ 6 & 629 & \infty & \infty & 262 & \infty & 0 & 180 \\ 7 & \infty & 610 & \infty & 353 & \infty & 180 & 0 \end{array}$$

$$S^{(0)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 56 & 7 & \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

At each step k (k = 1, 2, ..., 7), node k is considered as intermediate nodebetween every pair of nodes. The k^{th}row, k^{th} column, and diagonal elements of$D^{(k-1)}$ and $S^{(k-1)}$ will remain unchanged in step k. We have to update the values in the D matrix by using the condition;

$d_{ij}^{(k)} = \min\left\{d_{ij}^{(k-1)},\ d_{ik}^{(k-1)} + d_{kj}^{(k-1)}\right\}$ for step k.

For k=1,we need to calculate to update for all entries in D matrix by using respective pairs of row and column. For instance, the values in second row of$D^{(1)}$ matrix using each pairs with$i = 2$,and $j = 3, 4, 5, 6, 7$ and the values in the D matrix can be obtained as follows:

For $i = 2$ and $j = 3$, we use the condition:

$d_{23}^{(1)} = \min\left\{d_{23}^{(0)}, d_{21}^{(0)} + d_{13}^{(0)}\right\}$

$d_{23}^{(1)} = \min\{\infty, 201 + 402\}$.

Since $\infty > 201 + 402 = 603$, then updated value of $d_{23}^{(1)}$in $D^{(1)}$ matrix is 603.

For i=2, j=4:

$d_{24}^{(1)} = \min\left\{d_{24}^{(0)}, d_{21}^{(0)} + d_{14}^{(0)}\right\}$

$d_{24}^{(1)} = \min\{\infty, 201 + 591\}$

Since $\infty > 201 + 591 = 792$, then updated value of $d_{24}^{(1)}$in $D^{(1)}$ matrix changes into 792.

For i= 2, j= 5:

$d_{25}^{(1)} = \min\left\{d_{25}^{(0)}, d_{21}^{(0)} + d_{15}^{(0)}\right\}$

$d_{25}^{(1)} = \min\{\infty, 201 + \infty\}$

No update is performed and hence the value of$d_{25}^{(1)}$in $D^{(1)}$ matrix is ∞.

For i= 2, j= 6:

$d_{26}^{(1)} = \min\left\{d_{26}^{(0)}, d_{21}^{(0)} + d_{16}^{(0)}\right\}$

$d_{26}^{(1)} = \min\{644, 201 + 629\}$

Since $644 < 201 + 629 = 830$, then updated value of $d_{26}^{(1)}$in $D^{(1)}$ matrix is 644.

For i= 2, j= 7:

$d_{27}^{(1)} = \min\left\{d_{27}^{(0)}, d_{21}^{(0)} + d_{17}^{(0)}\right\}$

$d_{27}^{(1)} = \min\{\infty, 201 + 626\}$

Since$\infty > 201 + 626 = 827$, then updated value of $d_{27}^{(1)}$in $D^{(1)}$ matrix is827.

After calculating for the rest values of k and corresponding entries using similar calculation, $D^{(1)}$ matrix can be obtained as follows:

$$D^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 201 & 402 & 591 & \infty & 629 & 626 \\ 201 & 0 & 603 & 792 & \infty & 644 & 827 \\ \infty & \infty & 0 & 715 & 400 & 695 & 547 \\ 591 & 609 & 993 & 0 & 671 & 262 & 1217 \\ 869 & 1070 & 1271 & 1460 & 0 & 650 & 503 \\ 629 & 830 & 1031 & 262 & \infty & 0 & 180 \\ \infty & 610 & \infty & 353 & \infty & 180 & 0 \end{bmatrix} \end{array}$$

$$S^{(1)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 56 & 7 & \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

For k = 2,

After the generation of $D^{(1)}$and $S^{(1)}$matrices, the next iteration starts. Since iteration index $k = 2$, matrix updates are to be performed on the indices excluding $i = 2$ and $j = 2$. This means that we have to calculate the values for all entries of$D^{(1)}$ matrix by using the values in row 2 and column 2.

$$D^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 201 & 402 & 591 & \infty & 629 & 626 \\ 201 & 0 & 603 & 792 & \infty & 644 & 827 \\ \infty & \infty & 0 & 715 & 400 & 695 & 547 \\ 591 & 609 & 993 & 0 & 671 & 262 & 1217 \\ 869 & 1070 & 1271 & 1460 & 0 & 650 & 503 \\ 629 & 830 & 1031 & 262 & \infty & 0 & 180 \\ 811 & 610 & 1213 & 353 & \infty & 180 & 0 \end{bmatrix} \end{array}$$

$$S^{(2)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 56 & 7 & \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 2 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

For $k = 3$,

$$D^{(3)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 201 & 402 & 591 & 802 & 629 & 626 \\ 201 & 0 & 603 & 792 & 1003 & 644 & 827 \\ \infty & \infty & 0 & 715 & 400 & 695 & 547 \\ 591 & 609 & 993 & 0 & 671 & 262 & 1217 \\ 869 & 1070 & 1271 & 1460 & 0 & 650 & 503 \\ 629 & 830 & 1031 & 262 & 1431 & 0 & 180 \\ 811 & 610 & 1213 & 353 & 1613 & 180 & 0 \end{bmatrix} \end{array}$$

$$S^{(3)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 1 & 3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 3 & 0 & 0 \\ 2 & 0 & 2 & 0 & 3 & 0 & 0 \end{bmatrix} \end{array}$$

For $k = 4$,

$$D^{(4)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 201 & 402 & 591 & 802 & 629 & 626 \\ 201 & 0 & 603 & 792 & 1003 & 644 & 827 \\ 1306 & 1324 & 0 & 715 & 400 & 695 & 547 \\ 591 & 609 & 993 & 0 & 671 & 262 & 1217 \\ 869 & 1070 & 1271 & 1460 & 0 & 650 & 503 \\ 629 & 830 & 1031 & 262 & 933 & 0 & 180 \\ 811 & 610 & 1213 & 353 & 1024 & 180 & 0 \end{bmatrix} \end{array}$$

$$S^{(4)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 1 & 3 & 0 & 1 \\ 4 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 4 & 0 & 0 \\ 2 & 0 & 2 & 0 & 4 & 0 & 0 \end{bmatrix} \end{array}$$

For $k = 5$,

$$D^{(5)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 201 & 402 & 591 & 802 & 629 & 626 \\ 201 & 0 & 603 & 792 & 1003 & 644 & 827 \\ 1269 & 1324 & 0 & 715 & 400 & 695 & 547 \\ 591 & 609 & 993 & 0 & 671 & 262 & 1174 \\ 869 & 1070 & 1271 & 1460 & 0 & 650 & 503 \\ 629 & 830 & 1031 & 262 & 933 & 0 & 180 \\ 811 & 610 & 1213 & 353 & 1024 & 180 & 0 \end{bmatrix} \end{array}$$

$$S^{(5)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 1 & 3 & 0 & 1 \\ 5 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 5 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 4 & 0 & 0 \\ 2 & 0 & 2 & 0 & 4 & 0 & 0 \end{bmatrix} \end{array}$$

For $k = 6$,

$$D^{(6)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 201 & 402 & 591 & 802 & 629 & 626 \\ 201 & 0 & 603 & 792 & 1003 & 644 & 824 \\ 1269 & 1324 & 0 & 715 & 400 & 695 & 547 \\ 591 & 609 & 993 & 0 & 671 & 262 & 442 \\ 869 & 1070 & 1271 & 912 & 0 & 650 & 503 \\ 629 & 830 & 1031 & 262 & 933 & 0 & 180 \\ 809 & 610 & 1211 & 353 & 1024 & 180 & 0 \end{bmatrix} \end{array}$$

$$S^{(6)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 1 & 3 & 0 & 6 \\ 5 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 6 \\ 0 & 1 & 1 & 6 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 4 & 0 & 0 \\ 6 & 0 & 6 & 0 & 4 & 0 & 0 \end{bmatrix} \end{array}$$

For $k = 7$,

$$D^{(7)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 201 & 402 & 591 & 802 & 629 & 626 \\ 201 & 0 & 603 & 792 & 1003 & 644 & 824 \\ 1269 & 1157 & 0 & 715 & 400 & 695 & 547 \\ 591 & 609 & 993 & 0 & 671 & 262 & 442 \\ 869 & 1070 & 1271 & 856 & 0 & 650 & 503 \\ 629 & 790 & 1031 & 262 & 933 & 0 & 180 \\ 809 & 610 & 1211 & 353 & 1024 & 180 & 0 \end{bmatrix} \end{array}$$

$$S^{(7)} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{array} \begin{array}{c} \begin{array}{ccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 1 & 3 & 0 & 6 \\ 5 & 7 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 6 \\ 0 & 1 & 1 & 7 & 0 & 0 & 0 \\ 0 & 7 & 1 & 0 & 4 & 0 & 0 \\ 6 & 0 & 6 & 0 & 4 & 0 & 0 \end{bmatrix} \end{array}$$

Now, from matrices $D^{(7)}$and $S^{(7)}$, we can now determine the shortest route between every pair of places. Forinstance, the shortest distance from node 1 (YGN) to node 2 (KTO) is $d_{12}^{(7)} = 201\ km$.To determine the associated path, we

determinethe intermediate node$k = s_{12}^{(7)} = 0$, which indicates that there is no intermediate place between YGN and KTO.Again, shortest distance can be checked from node 7 (BGN) to node 3 (NPI) is $d_{73}^{(7)} = 1211\ km$. To determine the associated path, we consider forthe intermediate node$k = s_{73}^{(7)} = 6$, between node 7 and 3, yields the path$7 \rightarrow 6 \rightarrow 3$. Since from$S^{(7)}$, $s_{76}^{(7)} = 0$ and $s_{63}^{(7)} = 1$ (YGN) between node 6 (MDY) and node 3 (NPI), gives the path$6 \rightarrow 1 \rightarrow 3$. But, since, $s_{61}^{(7)} = 0$ and $s_{13}^{(7)} = 0$, therefore no further node exists between nodes 6 and 1 and also between nodes 1 and 3, respectively.

The combined result now gives the shortest path as$7 \rightarrow 6 \rightarrow 1 \rightarrow 3$(BGN-MDY-YGN-NPI).The associateddistance of the route is 1211 km. Another shortest distances and shortest paths can be checked using similar procedure as above by referring distance matrix $D^{(7)}$and node sequence matrix $S^{(7)}$.

The new values of the node sequence matrix$S^{(k)}$ is obtained updating the $S^{(k-1)}$with respect to the updated values of $D^{(k)}$ in each iteration. For example, at the first iteration($k = 1$), we can see the node sequence matrix $S^{(1)}$ followed by the distance matrix $D^{(1)}$ by updating the values of entries as "1"digit. The places of numbers"1s" in the node sequence matrix $S^{(1)}$were described those places in the $D^{(1)}$ matrix that have been updated in the first iteration. Likewise, the numbers 2, 3, 4, 5, and 6in the node sequence matrix S represent the places of entries in corresponding D matrix that have been substituted by updated values in the respective iteration steps 2, 3, 4, 5, and 6.

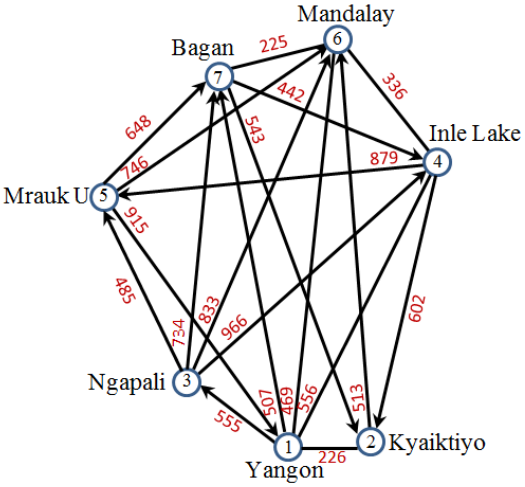

Fig. 3Representation by directed graph with time taken (minute) as weighted values

The weighted values in the Fig. 3 represent time taken (in minute) for the respective routes from Google Map.TheFloyd-Warshall algorithm can also be used to find the optimum travelling time in tourism. The same procedure of the algorithm has been applied to optimize the travelling time and can be seen in Table 1 as follows:

TABLE 1.THE SHORTEST PATHS FOR THE SHORTEST DISTANCE AND OPTIMIZED TRAVELLING TIMEBETWEEN ALL PAIR OF TOURISTS' ATTRACTIONSIN MYANMAR.

Starting place to destination place	Shortest distance (km)	Optimiz - edtravel lin-g time (min)	Shortest path (Based on travelling practice)
Yangon-Kyaiktiyo	201	226	YGN-KTO
Yangon-Ngapali	402	555	YGN-NPI
Yangon-Inle Lake	591	556	YGN-IL
Yangon-Myauk U	802	1040	YGN-NPI-MU
Yangon-Mandalay	629	469	YGN-MDY
Yangon-Bagan	626	507	YGN-BGN
Kyaiktiyo-Yangon	201	226	KTO-YGN
Kyaiktiyo-Ngapali	603	781	KTO-YGN-NPI
Kyaiktiyo-Inle Lake	792	782	KTO-YGN-IL
Kyaiktiyo-Myauk U	1003	1266	KTO-YGN-NPI-MU
Kyaiktiyo-Mandalay	644	513	KTO-MDY
Kyaiktiyo-Bagan	824	733	KTO-MDY-BGN (km) KTO-YGN-BGN(mn)
Ngapali-Yangon	1269	1302	NPI-MU-YGN (km) NPI-MDY-YGN(mn)
Ngapali-Kyaiktiyo	1157	1277	NPI-BGN-KTO
Ngapali-Inle Lake	715	966	NPI-IL
Ngapali-Myauk U	400	485	NPI-MJU
Ngapali-Mandalay	695	833	MPI-MDY
Ngapali-Bagan	547	734	NPI-BGN
Inle Lake-Yangon	591	556	IL-YGN
Inle Lake-Kyaiktiyo	609	602	IL-KTO
Inle Lake-Ngapali	993	1111	IL-YGN-NPI
Inle Lake-Myauk U	671	879	IL-MU
Inle Lake-Mandalay	262	336	IL-MDY
Inle Lake - Bagan	442	561	IL-MDY-BGN
Myauk U- Yangon	869	915	MU-YGN
Myauk U -Kyaiktiyo	1070	1141	MU-YGN-KTO
Myauk U -Ngapali	1271	1470	MU-YGN-BGN
Myauk U – Inle Lake	856	1082	MU-YGN-IL (km) MU-MDY-IL(mn)
Myauk U - Mandalay	650	746	MU-MDY
Myauk U - Bagan	503	648	MU-BGN
Mandalay- Yangon	629	469	MDY-YGN
Mandalay-Kyaiktiyo	790	695	MDY-BGN-KTO(km) MDY-YGN-KTO(mn)
Mandalay-Ngapali	1031	1024	MDY-YGN-NPI
Mandalay-Inle Lake	262	336	MDY-IL
Mandalay-Myauk U	933	1215	MDY-IL-MU
Mandalay- Bagan	180	225	MDY-BGN
Bagan - Yangon	809	694	BGN-MDY-YGN
Bagan -Kaiktiyo	610	543	BGN-KTO
Bagan - Ngapali	1211	1249	BGN-MDY-YGN-NPI
Bagan –Inle Lake	353	442	BGN-IL
Bagan – Myauk U	1024	1321	BGN-IL-MU
Bagan - Mandalay	180	225	BGN-MDY

The last column of above table shows the shortest paths for each pair of places on both distance and time based on the travelling practice of tourists. Most of the shortest paths are the same for both distance and time and very few routes are not. This is because of roads' qualities on those routes.

.

IV. CONCLUSION

Floyd-Warshalliterative Algorithm is applied for using dynamic programmingtechnique to find the shortest distance and minimum time for travellingto the most tourist attractions in Myanmar.Top seven tourists interesting places have been considered in this work. All weighted values for distance (km) and time (minute) to travel between the places in given road network are taken from Google Map application.Floyd-Warshall algorithm is an algorithm for solving all pairs of shortest path problem between every pair of vertices of the given graph. The Floyd–Warshall algorithm is a good choice for computing paths between all pairs of vertices in dense graphs, in which most or all pairs of vertices are connected by edges. Floyd-warshall's algorithm can be used not only just look for the shortest path between two particular nodes, but also the shortest path table between the nodes is created. The main advantage of Floyd-Warshall Algorithm is that it is extremely simple and easy to implement. The time complexity of the algorithm is $O(n^3)$ where n is the number of vertices in the graph. So its complexity depends only on the number of vertices in the graph. Based on the flowchart and detail calculation procedure, one can create computer codes such as C/C++, or JAVA, running these codes by using various weighted valuesfromactualinformation and data to solve general Floyd-Warshallshortest-path problems.

ACKNOWLEDGMENTS

The author would like to acknowledge her thank to Dr.Nilar Thein, Pro-rector, University of Computer Studies, Hinthada for her encouragement and permission todo research to write papers and tosubmit in conferences or journals. The author also would like to express her deeply gratitude to Dr. San San Maw, Professor, Department of Engineering Mathematics, Mandalay Technological University for her active encouragement, and perfect supervision throughout her research work and Dr. Lin Lin Naing, Professor and Head, Faculty of Computing, University of Computer Studies, Hinthada, for his invaluable comments and throughout her research work. Finally, the author's thanks are to her beloved mother, brother and sisters because her research cannot be done successfully unless they support it.

REFERENCES

[1] (2019) Tourism in Myanmar website. [Online]. Available: https://en.wikipedia. org/wiki/Tourism_in_Myanmar

[2] OrhanEroglu, ZaferAltugSayar and GurayYilmaz "An Optimization Approach for Airport Ground Operations with A Shortest Path Algorithm", Conference Paper INAIR, Amsterdam,NOV 2015.

[3] (2019)Floyd-Warshall Algorithm website.[Online]. Available:https://en.wikipedia. org/wiki/ Floyd–Warshall_algorithm/

[4] R W. Floyd, "Algorithm 97: shortest path".Communications of the ACM, Volume 5, Issue 6,pp 345, June, 1962.

[5] Eshu Gupta, "Global Positioning System using Dijkstra's Algorithm Traffic Planning System", International Journal of Computer Trends and Technology (IJCTT), Volume 35, Number 5, pp-231-235, May 2016,

[6] Rongyan Xu ; Dejun Miao ; Lu Liu ; John Panneerselva, "An Optimal Travel Route Plan for Yangzhou Based on the Improved Floyd Algorithm", Proceeding of IEEE International Conference on Internet of Things, 21-23,June, 2017.[Online]. Available: https://ieeexplore. ieee.org/document/8276747

[7] SantanuSaha Ray, Graph Theory with Algorithms and its Applications in Applied Science and Technology, Springer New Delhi Heidelberg New York Dordrecht London, Springer India, 2013.

[8] (2018) Design and Analysis of Floyd-Warshall Shortest Path Algorithm website, [Online]. Available: https://www.gatevidyalay. com/floyd-warshall-algorithm-shortest-path-algorithm/