Journal of Mathematics Research; Vol. 13, No. 5; October 2021
ISSN 1916-9795 E-ISSN 1916-9809
Published by Canadian Center of Science and Education

A Comprehensive Comparison of Label Setting Algorithm and Dynamic Programming Algorithm in Solving Shortest Path Problems

Douglas Yenwon Kparib[1], John Awuah Addor[1] & Anthony Joe Turkson[1]

[1] Department of Mathematics, Statistics and Actuarial Science, Takoradi Technical University, Takoradi, Ghana

Correspondence: Douglas Yenwon Kparib, Department of Mathematics, Statistics and Actuarial Science, Takoradi Technical University, Takoradi, Ghana

Received: February 25, 2021 Accepted: September 8, 2021 Online Published: September 21, 2021

doi:10.5539/jmr.v13n5p14 URL: https://doi.org/10.5539/jmr.v13n5p14

Abstract

In this paper, Label Setting Algorithm and Dynamic Programming Algorithm had been critically examined in determining the shortest path from one source to a destination. Shortest path problems are for finding a path with minimum cost from one or more origin (s) to one or more destination(s) through a connected network. A network of ten (10) cities (nodes) was employed as a numerical example to compare the performance of the two algorithms. Both algorithms arrived at the optimal distance of 11 km, which corresponds to the paths $1 \to 4 \to 5 \to 8 \to 10$,$1 \to 3 \to 5 \to 8 \to 10$, $1 \to 2 \to 6 \to 9 \to 10$ and $1 \to 4 \to 6 \to 9 \to 10$. Thus, the problem has multiple shortest paths. The computational results evince the outperformance of Dynamic Programming Algorithm, in terms of time efficiency, over the Label Setting Algorithm. Therefore, to save time, it is recommended to apply Dynamic Programming Algorithm to shortest paths and other applicable problems over the Label-Setting Algorithm.

Keywords: Label-Setting Algorithm, Dynamic Programming Algorithm, Shortest Path Problems, Dijkstra's algorithm, time efficiency

1. Introduction

A network is a collection of entities connected by some relationship, which can be represented as a graph. The Shortest Path Problems (SPP) are concerned with finding a path with minimum distance from one or more origins to one or more destinations through a network (Yongtaek & Sungmo, 2010; Paraveen & Neha, 2013).

Classical examples of Shortest Path Network Problems are Travelling Salesman Problem (TSP), Knapsack Problem (KP) and Capacitated Vehicle Routing Problem (CVRP). With the increasing application of Shortest Path Network Problems in human life, researchers in algorithms for solving shortest path network problems have been compelled to look outside the traditional algorithms such as Label Setting and Label Correcting, which have some deficits to novel algorithm such as Dynamic Programming.

This paper therefore makes a comparative analysis between Label Setting Algorithm and Dynamic Programming Algorithm using a network of ten (10) nodes network problems.

2. Method

The Method section describes in detail how the study was conducted, including conceptual and operational definitions of the variables used in the study, Different types of studies will rely on different methodologies; however, a complete description of the methods used enables the reader to evaluate the appropriateness of your methods and the reliability and the validity of your results, It also permits experienced investigators to replicate the study, If your manuscript is an update of an ongoing or earlier study and the method has been published in detail elsewhere, you may refer the reader to that source and simply give a brief synopsis of the method in this section.

2.1 Basic Features of the Problem

The problem at hand is a network or graph problem. The aim is to find all edges with minimum distance from the source to destination nodes.

2.2 Shortest Path Algorithms

This section looks at the Label-Setting (LS) and Dynamic Programming Algorithms for solving shortest path problems and the various ways that they are implemented. An algorithm is a systematic computational step used to solve a problem (Cormen *et al*., 2005).

2.2.1 Label-Setting Algorithm

This Label-Setting Algorithm was named after the famous Dutch computer scientist and mathematician, Dijktsra, whose 1959 algorithm became the foundation of this type of algorithm. The algorithm is known as Dijkstra's algorithm. According to Schrijver (2010), similar algorithm had been proposed independently by Leyzorek *et al.* (1957), Dantzig (1958), and Whiting and Hillier (1960).

Dijkstra's algorithm is a graph search algorithm that solves the single-source shortest path problem for a graph with non-negative edge path costs, producing a shortest path tree (Vaibhvi & Chitra, 2014). This algorithm is often used in routing problems and as a subroutine in other graph algorithms (Paraveen & Neha, 2013). For a given source vertex (node) in the graph, the algorithm finds the path with lowest cost (i.e. the shortest path) between that vertex and every other vertex.

At any intermediate step, the algorithm divides the nodes of the network under consideration into two groups: those which it designates as permanently labelled (or permanent), and those that it designates as temporarily labelled (or temporal). The distance labelled permanent to any permanent node represents the shortest from the source node to the current node (Ukwosah *et al.*, 2018).

2.2.2 Mathematical Formulation of the Problem Based on the Label-Setting Algorithms

Let d_i be the shortest distance from source node 1 to node i and define $d_{ij} \geq 0$ as the length of arc(i,j). Then the algorithm defines the label for an immediately succeeding node j as $[d,i] = \left[d_i + d_{ij}, i\right]$. The label for starting node is [0, -], signifying that the node has no predecessor.

Below are the steps of this algorithm:

Step 1: The source node (node 1) is assigned with the permanent label [0, -]; set i = 1.

Step 2: (i) Compute the temporary label$\left[d_i + d_{ij}, i\right]$for each node j that is reachable by node i, provided j is not permanently labelled. If node j is already labelled with$\left[d_j, k\right]$ through another node k and if $d_i + d_{ij} < d_j$, then replace$[d_j, k]$ with $\left[d_i + d_{ij}, i\right]$.

(ii) Stop, if all the nodes are permanently labelled. Otherwise, select the label$[d_r, s]$ having the shortest distance (d_r) among all the temporary labels (break ties arbitrarily). Set$i = r$ and repeat step 2.

2.3 Dynamic Programming

The concept of Dynamic Programming was first conceived by Richard Bellman in 1940s and was recognized by the Institute of Electrical and Electronic Engineers (IEEE) as analysis and engineering topic in 1953 (Smith *et al.*, 1997; Hillier & Lieberman, 2005).

Bellman, described Dynamic Programming as the process of solving problems where one needs to find the best decision, one after another (Adda *et al.*, 2003). It works on the principle of finding an overall solution by operating on an intermediate point that lies between where we are now and where we want to go. The procedure is recursive in that the next intermediate point is a function of the point already visited. The principle of Bellman optimality states that 'an optimal policy (set of decisions) has the property that whatever the initial state and decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision' (Smith *et al.*, 1997).

2.3.1 Mathematical Formulation of the Problem Based on Dynamic Programing Algorithm

A problem can be solved by the method of Dynamic Programming if it has the following properties:

a) The problem can be decomposed into a sequence of decisions made at various stages.

b) Each stage has a number of possible states.

c) A decision takes one from a state at one stage to some state at the next stage.

d) The best sequence of decision also known as policy at any stage is independent of the decisions made at prior stages.

e) There is a well-defined cost for traversing from state across stages. Moreover, there is a recursive relationship for choosing the best decision. The shortest routes will be determined between every two stages for a given network using the Dynamic Programming Algorithm.

Step 0. Step 0. Definition of variables

Let d_{ij} denote the distance from node i to node j.

Let i be the node it is at the stage n and j the nodes reachable by node i.

Let x_{ni}be the state variable.

Let $f_n(x_{n,i})$ denote the minimum distance of the objective function from node $x_{n,i}$ to the final destination 10.

Step 1: The structure of an optimal solution.

The first step of Dynamic Programming is to characterize the structure of an optimal solution.

Let divide the problem into five stages as follows;

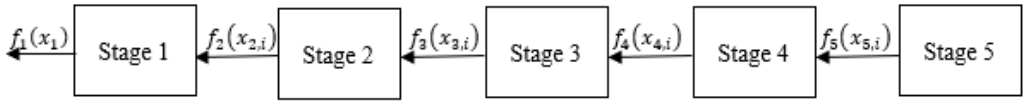

Figure 1. Schematic representation of the problem as a five-stage dynamic programming problem

Stage 1: Consists of city 1.

The return function is

$f_1(x_1) =$ immediate distance (stage1) + minimum future distance (stage2).

Stage 2: Consists of cities 2, 3 and 4.

The return function is

$$f_2(x_2) = \text{immediate distance (stage2)} + \text{minimum future distance (stage3)}$$

Stage 3: Consists of cities 5,6 and 7.

The return function is

$$f_3(x_3) = \text{immediate distance (stage3)} + \text{minimum future distance (stage4)}$$

Stage 4: Consists of cities 8 and 9.

The return function is

$$f_4(x_4) = \text{immediate distance (stage4)} + \text{minimum future distance (stage5)}$$

Stage 5: Consists of city 10.

This is the final stage. Therefore $f_5(x_5) = 0$.

Step 2: A recursive Solution

Let $f_n(x_{n,i})$ denote the minimum distance of the objective function from node $x_{n,i}$ to the final destination 10.

Let $f_n(x_{n,i})$ denote the optimal value of the objective function from any city $x_{n,i}$ to the final destination 10. Hence, the optimum is $f_1(x_1)$, the minimum of the sum of the distance from 1 to 10.

Thus, $f_n(x_{n,i}) = min\{d_{ij} + f_n(x_{n+1,j})$ subject to $d_{ij} \geq 0$ and an integer.

(i) For stage 1:$n = 1, x_1 = 1$

We minimize the distance from stage 1(1) to stage 5(10).

$$f_1(x_{1,i}) = min\{d_{ij} + f_2(x_{2,j})$$

(ii) For stage 2: $n = 2$

We minimize the distance from stage 2(2, 5, and 6) to stage 5(10).

$$f_2(x_{2,i}) = min\{d_{ij} + f_3(x_{3,j})$$

(iii) For stage 3: $n = 3$

We minimize the distance from stage 3(5, 6, and 7) to stage 5(10).

$$f_3(x_{3,i}) = min\{d_{ij} + f_4(x_{4,j})$$

(iv) For stage 4: $n = 4$

We minimize the distance from stage 4(8, 9) to stage 5(10).

$f_4(x_{4,i}) = min\{d_{ij} + f_5(x_{5,j})$.

(v) For stage 5: $n = 5, x_5$.

Since the ultimate destination (state x_5) is reached at the end of stage 5, $f_5(x_5) = 0$.

Step 3: Computing the stage-wise optimal distance

This is the final step in the complete process of the Dynamic Programming Algorithm. The details of this step will be outlined in section 3 to obtain the optimal distances at the various stages.

3. Results

In this section we present a ten-node network of 20 edges or acrs. In this paper, the nodes represent ten cities and the edges feature their interconnections indicated the respective distances connecting them (See Figures 1 and 2). First, the Dijkstra's algorithm will be implemented and then, the Dynamic Programming Algorithm to inform a comprehensive comparison. Each arc (i, j) is associated with the distance d_{ij} in km.

3.1 Implementation of the Dijkstra's Algorithm

The Dijkstra's algorithm is implemented using Figure 2. Each arc (i, j) is associated with the distance d_{ij} in km. The main aim is to determine the shortest path from the source city 1 (node 1) to the destination city 10 (node 10). Figure 2 was adopted from Hillier and Lieberman (2005), which describes a journey from a starting point, Missouri (node 1) to a destination, California (node 10).

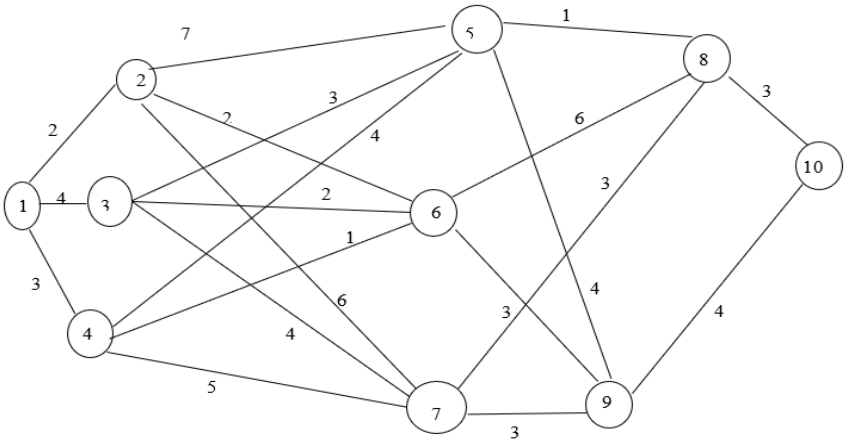

Figure 2. A network of ten cities featuring the distances linking them

For Dijkstra's algorithm, we start with iteration 0. The tables below provide the summary of results.

In the iterations, Column one denotes the nodes; column two, the labels; and column three, the status of the nodes.

Iteration 0: Node 1 is assigned the permanent label [0, -]:

Iteration 1: Nodes 2, 3, and 4 are reachable by node 1

Node	Label	Status
1	$[0, -]$	Permanent
2	$[0 + 2, 1] = [2, 1]$	Temporary
3	$[0 + 4, 1] = [4, 1]$	Temporary
4	$[0 + 3, 1] = [3, 1]$	Temporary

The least distance among the temporary labels [2, 1], [4, 1] and [3, 1] is 2, which corresponds with node 2 i.e., $[d_2]$. Thus, the status of node 2 is changed to permanent.

Iteration 2: Nodes 5, 6 and 7 are reachable by node 2

Node	Label	Status
1	$[0, -]$	Permanent
2	$[2, 1]$	Permanent
3	$[4, 1]$	Temporary
4	$[3, 1]$	Temporary
5	$[2 + 7, 2] = [9, 2]$	Temporary
6	$[2 + 2, 2] = [4, 2]$	Temporary
7	$[2 + 6, 2] = [8, 2]$	Temporary

Among the temporary labels [4, 1], [3, 1], [9, 2], [4, 2], and [8, 2], the smallest distance is 3, and that corresponds with

node 4 i.e.,d_4. Thus, the status of node 4 is changed to permanent.

Iteration 3: Nodes 5, 6 and 7 are reachable by node 4

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Temporary
4	$[3,1]$	Permanent
5	$[3+4,4]=[7,4]$	Temporary
6	$[4,2]$ or $[3+1,4]=[4,4]$	Temporary
7	$[8,2]$ or $[3+5,4]=[8,4]$	Temporary

Among the temporary labels [4, 1], [7, 4], [4, 2], [4, 4], [8, 4]and [8, 2], the smaller distance is 4, and that corresponds with node 3 and node 6 (break ties arbitrarily) i.e., $[d_3]$. Thus, the status of node 3 is changed to permanent.

Iteration 4: Nodes 5, 6 and 7 are reachable by node 3

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Permanent
4	$[3,1]$	Permanent
5	$[7,4]$ or $[3+4,3]=[7,3]$	Temporary
6	$[4,2]$ or $[3+1,4]=[4,4]$	Temporary
7	$[8,2]$ or $[8,4]$ *or* $[4+4,3]=[8,3]$	Temporary

Among the temporary labels [7, 4], [7, 3], [4, 2], [4, 4], [8, 2], [8, 3] and [8, 4], the smallest distance is 4, and that corresponds with node 6 i.e., $[d_6]$. Thus, the status of node 6 is changed to permanent.

Iteration 5: Nodes 8 and 9 are reachable by node 6.

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Permanent
4	$[3,1]$	Permanent
5	$[7,4]$ or $[3+4,3]=[7,3]$	Temporary
6	$[4,2]$ or $[3+1,4]=[4,4]$	Permanent
7	$[8,2]$ or $[8,4]$ or $[4+4,3]=[8,3]$	Temporary
8	$[4+6,6]=[10,6]$	Temporary
9	$[4+3,6]=[7,6]$	Temporary

Among the temporary labels [7, 4], [7, 3], [8, 2], [8, 3], [8, 4], [10, 6] and [7, 6], the smallest distance is 7, which corresponds with node 5 and node 9 i.e., $[d_5]$. Thus, the status of node 5 is changed to permanent.

Iteration 6: Nodes 8 and 9 are reachable by node 5.

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Permanent
4	$[3,1]$	Permanent
5	$[7,4]$ or $[3+4,3]=[7,3]$	Permanent
6	$[4,2]$ or $[3+1,4]=[4,4]$	Permanent
7	$[8,2]$ or $[8,4]$ or $[4+4,3]=[8,3]$	Temporary
8	$[7+1,5]=[8,5]$	Temporary
9	$[4+3,6]=[7,6]$	Temporary

Among the temporary labels [8, 2], [8, 3], [8, 4], [8, 5] and [7, 6], the smallest distance is 7, and that corresponds with node 9 i.e., $[d_9]$. Thus, the status of node 9 is changed to permanent.

Iteration 7: Node 10 is reachable by node 9.

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Permanent
4	$[3,1]$	Permanent
5	$[7,4]\ or\ [3+4,3] = [7,3]$	Permanent
6	$[4,2]\ or\ [3+1,4] = [4,4]$	Permanent
7	$[8,2]\ or\ [8,4]\ or\ [4+4,3] = [8,3]$	Temporary
8	$[7+1,5] = [8,5]$	Temporary
9	$[4+3,6] = [7,6]$	Permanent
10	$[4+7,9] = [11,9]$	Temporary

Among the temporary labels [8, 2], [8, 3], [8, 4], [8, 5] and [11, 9], the smallest distance is 8, and that corresponds with node 7 and node 8 i.e., $[d_7]$. Thus, the status of node 7 is changed to permanent.

Iteration 8: Nodes 8 and 9 are reachable by node 7.

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Permanent
4	$[3,1]$	Permanent
5	$[7,4]$ or $[3+4,3] = [7,3]$	Permanent
6	$[4,2]$ or $[3+1,4] = [4,4]$	Permanent
7	$[8,2]\ or\ [8,4]\ or\ [4+4,3] = [8,3]$	permanent
8	$[7+1,5] = [8,5]$	Temporary
9	$[4+3,6] = [7,6]$	Permanent
10	$[4+7,9] = [11,9]$	Temporary

Among the temporary labels [8, 5] and [11, 9], the smallest distance is 8, and that corresponds with node 8 i.e., $[d_8]$. Thus, the status of node 8 is changed to permanent.

Iteration 9: Node 10 is reachable by node 8.

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Permanent
4	$[3,1]$	Permanent
5	$[7,4]$ or $[3+4,3] = [7,3]$	Permanent
6	$[4,2]$ or $[3+1,4] = [4,4]$	Permanent
7	$[8,2]$ or $[8,4]$ or $[4+4,3] = [8,3]$	permanent
8	$[7+1,5] = [8,5]$	Permanent
9	$[4+3,6] = [7,6]$	Permanent
10	$[11,9]$ or $[3+8,8] = [11,8]$	Temporary

Among the temporary labels [11, 9] and [11, 8], the smallest distance is 11, which corresponds with node 10 i.e., $[d_{10}]$. Thus, the status of node 10 is changed to permanent.

Iteration 10: The final table, node 10 status is also changed to permanent

Node	Label	Status
1	$[0,-]$	Permanent
2	$[2,1]$	Permanent
3	$[4,1]$	Permanent
4	$[3,1]$	Permanent
5	$[7,4]$ or $[3+4,3] = [7,3]$	Permanent
6	$[4,2]$ or $[3 + 1,4] = [4,4]$	Permanent
7	$[8,2]$ or $[8,4]$ or $[4 + 4,3] = [8,3]$	permanent
8	$[7 + 1,5] = [8,5]$	Permanent
9	$[4 + 3,6] = [7,6]$	Permanent
10	$[11,9]$ or $[3 + 8,8] = [11,8]$	Permanent

3.2 Implementation of Dynamic Algorithm

In section 3.2, the Dynamic Programming algorithm is also implemented. The shortest routes will be determined between every two stages for the network, shown in Figure 1 using the Dynamic Programming Algorithm. Figure 3 gives the division of the Figure 2 into five stages.

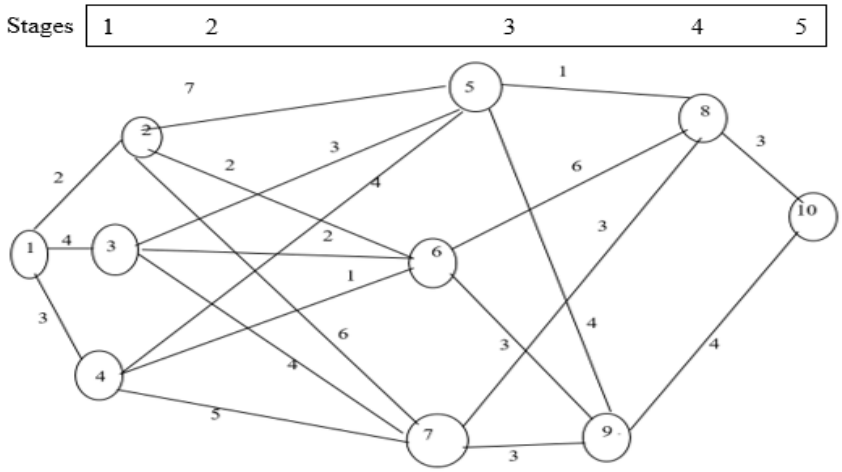

Figure 3. A five-stage division of figure 2

The main aim is to determine the minimum distance from node 1 to node 10.

To illustrate the Step 3, the computation of the stage-wise optimal distances (which was deferred above), we do as follows:

(i) Computations for stage 5 (node 10): $n = 5$.

Since node 10 is the final stage, there is distance after node 10, then $f_5(x_{5,10}) = 0$.

(ii) Computations for stage 4 (nodes 8, 9): $n = 4$. When there (is) one more stage to go $(n = 4)$

$$f_4(x_{4,8}) = d_{8,10} + f_5(x_{5,10}) = 3 + 0 = 3$$
$$f_4(x_{4,9}) = d_{9,10} + f_5(x_{510}) = 4 + 0 = 4$$

(iii) Computation for stage 3 (nodes 5, 6, 7): $n = 3$. When there are two more stages go $(n = 3)$.

$$f_3(x_{3,,5}) = min\begin{cases} d_{5,8} + f_4(x_{4,8}) = 1 + 3 = 4 \\ d_{5,9} + f_4(x_{4,9}) = 4 + 4 = 8 \end{cases}$$
$$result f_3(x_{3,5}) = 4$$

$$f_3(x_{3,6}) = min\begin{cases} d_{6,8} + f_4(x_{4,8}) = 6 + 3 = 9 \\ d_{6,9} + f_4(x_{4,9}) = 3 + 4 = 7 \end{cases}$$
$$result f_3(x_{3,6}) = 7$$
$$f_3(x_{3,7}) = min\begin{cases} d_{7,8} + f_4(x_{4,8}) = 3 + 3 = 6 \\ d_{7,9} + f_4(x_{4,9}) = 3 + 4 = 7 \end{cases}$$
$$result f_3(x_{3,7}) = 6$$

(iv) Computation for stage 2 (nodes 2, 3, 4): $n = 2$. The solution for the second stage ($n = 3$), where there are three stages to go.

$$f_2(x_{2,2}) = min\begin{cases} d_{2,5} + f_3(x_{3,5}) = 7 + 4 = 11 \\ d_{2,6} + f_3(x_{3,6}) = 4 + 7 = 11 \\ d_{2,,7} + f_3(x_{3,7}) = 6 + 6 = 12 \end{cases}$$
$$result f_2(x_{2,,2}) = 11$$
$$f_2(x_{2,3}) = min\begin{cases} d_{3,5} + f_3(x_{3,5}) = 3 + 4 = 7 \\ d_{3,6} + f_3(x_{3,6}) = 2 + 7 = 9 \\ d_{3,7} + f_3(x_{3,7}) = 4 + 6 = 10 \end{cases}$$
$$result f_2(x_{2,3}) = 7$$
$$f_2(x_{2,4}) = min\begin{cases} d_{4,5,} + f_3(x_{3,5}) = 4 + 4 = 8 \\ d_{4,6} + f_3(x_{3,6}) = 1 + 7 = 8 \\ d_{4,7} + f_3(x_{3,7}) = 6 + 6 = 12 \end{cases}$$
$$result f_2(x_{2,4}) = 8$$

(v) Computation for stage 1 (node 1): $n = 1$. Moving to the first stage $(n = 3)$, with all four stage(s) to go.

$$f_1(x_1) = min\begin{cases} d_{1,2} + f_2(x_{2,2}) = 2 + 11 = 13 \\ d_{1,3} + f_2(x_{2,3}) = 4 + 7 = 11 \\ d_{1,4} + f_2(x_{2,4}) = 3 + 8 = 11 \end{cases}$$
$$result f_1(x_1) = 11$$

Since the minimum distance is 11, $f_1(x_1) = 11$ which corresponds with $d_{1,3} + f_2(x_{2,3})$ and $d_{1,3} + f_2(x_{2,3})$. Note that x_{23} means stage 2, node 3 and x_{24} means stage 2, node 4.

The shortest path from node 1 to node 10 is therefore obtained by tracing the sequences:

(i) $x_1, x_{2,4}, x_{3,6}, x_{4,9}$ and $x_{5,10}$ as represented by the following order of flow: $1 \to 4 \to 6 \to 9 \to 10$

(ii) $x_1 \to x_{2,4} \to x_{3,5} \to x_{4,8} \to x_{5,10}$ which is indicated by the order of flow $1 \to 4 \to 5 \to 8 \to 10$ with a total distance 11 km.

(iii) $x_1, x_{2,3}, x_{3,5}, x_{4,8}$ and $x_{5,10}$. This is represented in the following order: $1 \to 3 \to 5 \to 8 \to 10$ with a total distance 11 km.

4. Discussion

We present in this section, the discussion of the results obtained from the Dijkstra's algorithm and that of the Dynamic Programming algorithm. In view of the Dijkstra's algorithm, we infer from Iteration 1 through to Iteration 10. The shortest path from node 1 to node 10 is therefore obtained by tracing the sequences below.

(i) $(10) \to [11, 8] \to (8) \to [7, 5] \to (5) \to [7, 4] \to (4) \to [3, 1] \to (1)$. Thus, the required path is represented by the sequence $1 \to 4 \to 5 \to 8 \to 10$ with total distance 11 km.

(ii) $(10) \to [10, 8] \to (8) \to [7, 5] \to (5) \to [7, 3] \to (3) \to [3, 1] \to (1)$. The required path is therefore, indicated by the sequence $1 \to 3 \to 5 \to 8 \to 10$ with total distance 11 km.

(iii) $(10) \to [11, 9] \to (9) \to [7, 6] \to (6) \to [4, 2] \to (2) \to [2, 1] \to (1)$. Again, it is obvious that the

required path is $1 \rightarrow 2 \rightarrow 6 \rightarrow 9 \rightarrow 10$ with total distance 11 km.

(iv) $(10) \rightarrow [11,9] \rightarrow (9) \rightarrow [7,6] \rightarrow (6) \rightarrow [4,4] \rightarrow (4) \rightarrow [3,1] \rightarrow (1)$. Therefore, the optimal path is exemplified by the sequence $1 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 10$ with total distance 11 km.

In summary, the optimal distance connecting the cities as obtained from the implemented Dijkstra's algorithm is 11 km.

As an alternative technique, the Dynamic Programming algorithm also yields an optimal or minimum distance of 11, as given by $f_1(x_1) = 11$ which corresponds to $d_{1,3} + f_2(x_{2,3})$ and $d_{1,3} + f_2(x_{2,3})$. It is imperative to define the notation x_{ij}. The first index i, denotes the stage whiles the second j, denotes the node. For instance, x_{23} means stage 2, node 3 and x_{24} means stage 2, node 4. As can be followed from the implementation of the Dynamic Programming algorithm, the shortest path from node 1 to node 10 is therefore obtained by tracing the following sequences:

(i) $x_1, x_{2,4}, x_{3,6}, x_{4,9}$ and $x_{5,10}$ as represented by the following order of flow: $1 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 10$ giving rise to an optimal distance of 11 km.

(ii) $x_1 \rightarrow x_{2,4} \rightarrow x_{3,5} \rightarrow x_{4,8} \rightarrow x_{5,10}$ which is indicated by the order of flow $1 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 10$ with a total distance 11 km.

(iii) $x_1, x_{2,3}, x_{3,5}, x_{4,8}$ and $x_{5,10}$. This is represented in the following order: $1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 10$ with a total distance 11 km.

5. Conclusion

In this paper, the performance of Dynamic Programming Algorithm is compared with that of the Label-Setting Algorithm. The two algorithms were implemented in solving the shortest path problem of a ten-city (node) connected by 20 edges representing varying distances in kilometers.

The two algorithms obtained the sequences that proxy the shortest or optimal path to the problem to be $1 \rightarrow 4 \rightarrow 5 \rightarrow 8 \rightarrow 10$,$1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 10$, $1 \rightarrow 2 \rightarrow 6 \rightarrow 9 \rightarrow 10$ and $1 \rightarrow 4 \rightarrow 6 \rightarrow 9 \rightarrow 10$ with a total distance of 11 km. Thus, in reality, the problem typifies a Multiple Shortest Path Problem. The results have uncovered the fact that Dynamic Programming is equally as good as the Label-Setting or the Dijkstra's algorithm, if and only if the problem is well formulated and the steps are thoroughly followed.

Albeit the fact that Hillier and Lieberman (2005) cited an inherent flaw of Dynamic Programming method in respect of its impromptu nature and the fact that it is designed to fit a particular problem rather than a variety of applications, it involves fewer iterations leading to fewer computational steps, thereby making it time efficient in relation to the Dijkstra's (Label-Setting) algorithm. The current trend in the development of algorithms focuses on algorithms that are exemplified by optimal efficiency (optimal performance) and computational or time efficiency. As evinced by the results, both the Dynamic algorithm and the Dijkstra's algorithm perform equally in terms of optimal efficiency. However, on grounds of time efficiency, Dynamic Programming algorithm outperforms the Dijkstra's algorithm. It is therefore recommended in this paper that Dynamic Programming should be applied in solving a shortest path and other applicable problems to save time. Also, further research needs to be done on other areas that Dynamic Programming Algorithm could be applied to solve problems since it provides quality results and saves time.

Acknowledgments

All authors made equal contributions in terms of the design, literature review, methods or mathematical formulation of the problem, mathematical computations, analysis and discussions of results, preparation of the initial manuscript and final review of the manuscript. All authors read through the final document and agreed to the content before the final submission. All authors declare that there is no competing interest as far as this paper is concerned.

References

Adda, J., Cooper, R., & Cooper, R. W. (2003). Dynamic economics: Quantitative methods and applications. MIT press.

Cormen, T. H., Leiserson, E. C., Rivest, L. R., & Stein, C. (2005). Introduction to algorithms (2nd ed.). Cambridge: MIT Pres.

Dantzig, G. B. (1958). Discrete-variable Extremum Problems. *Operations Research*, *5*(2), 266-288. https://doi.org/10.1287/opre.5.2.266

Dijkstra, E. (1959). A Note on Two Problems in Connection with Graphs. *Numerical Mathematics*, *1*(4), 395–412. https://doi.org/10.1007/BF01386390

Hillier, F. S., & Lieberman, G. (2005). Introduction to operations research (8th ed.). McGraw Hill.

Karmiloff-Smith, A. (1997). Crucial Differences between Developmental Cognitive Neuroscience and Adult Neuropsychology. *Developmental Neuropsychology*, *13*(4), 513-524. https://doi.org/10.1080/87565649709540693

Leyzorek, M., Gray, R. S., Johnson, A. A., Ladew, W. C., Meaker, S. R., Petry, R. M., & Seitz, R. N. (1957). Investigation of model techniques. *First Annual Report*, *6*.

Paraveen, S., & Neha, K. (2013). Study of Optimal Path Finding Techniques. *International Journal of Advancements in Technology*, *4*(2), 0976 – 4860.

Schriever, A. (2010). On the History of the Shortest Path Problem. *Documenta Mathematica*, *17*, 155.

Ukwosah, E. C., Oladunjoge, J. A., & Siman, E. (2018). A Study of Intelligent Route Guidance System Dijkstra's Heuristic Shortest Path Algorithm. *International Journal of Information, Technology and Innovations in Africa*, 1 – 23.

Vaibhvi, P., & Chitra, B. (2014). A Survey Paper of Bellman-Ford Algorithm and Dijkstra's Algorithm for Finding Shortest Path in GIB Application. *International Journal of P2P Network Trends and Technology*.

Whiting, P. D., & Hillier, J. A. (1960). A Method for Finding the Shortest Route through a Road Network. *Journal of the Operational Research Society*, *11*(12), 37-40. https://doi.org/10.1057/jors.1960.32

Yongtaek, L., & Sungmo, R. (2010). An Efficient Dissimilar Path Searching Method for Evacuation Routing. *KSCE Journal of Civil Engineering*, *14*(1), 61-67. https://doi.org/10.1007/s12205-010-0061-4