International Journal of Mathematical, Engineering and Management Sciences
Vol. 8, No. 5, 991-1005, 2023
https://doi.org/10.33889/IJMEMS.2023.8.5.057

Extended TANYAKUMU Labelling Method to Compute Shortest Paths in Directed Networks

Trust Tawanda
Department of Statistics and Operations Research,
National University of Science and Technology, Bulawayo, Zimbabwe.
E-mail: trust.tawanda@nust.ac.zw, trustawanda@gmail.com

Elias Munapo
School of Economics and Decision Sciences,
North West University, Mafikeng Campus Mafikeng, South Africa.
E-mail: elias.munapo@nwu.ac.za, emunapo@gmail.com

Santosh Kumar
School of Mathematical and Geospatial Sciences,
RMIT University, Melbourne, Australia.
Corresponding author: santosh.kumar@rmit.edu.au, santosh.kumarau@gmail.com

Philimon Nyamugure
Department of Statistics and Operations Research,
National University of Science and Technology, Bulawayo, Zimbabwe.
E-mail: philimon.nyamugure@nust.ac.zw, pnyamugure@gmail.com

(Received on May 3, 2023; Accepted on July 12, 2023)

Abstract
Shortest path problem (SPP) has various applications in areas such as telecommunications, transportation and emergency services, and postal services among others. As a result, several algorithms have been developed to solve the SPP and related problems. The current paper extends the TANYAKUMU labelling method for solving the Travelling salesman problem (TSP) to solve SPP in directed transportation networks. Numerical illustrations are used to prove the validity of the proposed method. The main contributions of this paper are as follows: (i) modification of TSP algorithm to solve single source SPP, (ii) the developed method numerically evaluated on four increasingly complex problems of sizes 11×11, 21×21, 23×23 and 26×26 and lastly (iii) the solutions obtained from solving these four problems are compared with those obtained by Minimum incoming weight label (MIWL) algorithm. The proposed algorithm computed the same shortest paths as the MIWL algorithm on all four problems.

Keywords- TANYAKUMU labelling method, MIWL algorithm, Shortest path problem, Transportation network.

1. Introduction
Determination of the shortest route is a basic problem in network theory; a lot of approaches have been suggested, yet this problem remains active and significant due to its applications in real-life industrial situations. Shortest path problems (SPP) deal with the computation of the shortest path or route from the source node to the destination node through a set of intermediate nodes in the network. The weight on any link joining any two given nodes in the network may be distance, time taken, accident probability, or fuel consumption, among others. Companies that are in the shipping and transportation business need to use the shortest routes when shipping their product from one place to another, thus saving on fuel, vehicle tires, and driver overtime, among others. Shortest path problems can exist in directed networks, undirected networks, as well as multi-objective directed and undirected networks. Shortest path problem

has many applications in different areas, which include fire station services, emergency services, network communications (Di Caprio et al., 2022); Data routing, web search optimization, and social network analysis (D'Emidio, 2020); Telecommunication (Lewis, 2020); and Travel time reduction (Nguyen et al., 2022). Shortest path problem has several variants, such as constrained shortest path problem (Wu et al., 2023), Clustered shortest path problem (Petrovan et al., 2023), Minmax regret shortest path problem (Carvalho et al., 2023), SPP with positive integer weight in linear time (Thorup, 1999), SPP with parallel graph library (Edmonds et al., 2006), and SPP with specified nodes (Saksena and Kumar, 1966), among others.

Kumar et al. (2013) proposed a minimum weight labelling method for computing the shortest path in a non-directed network. In a *k*-node network, the algorithm terminates after (*k*-1) iterations. No comparisons with other related algorithms were reported. Rosita et al. (2019) combined the Dijkstra algorithm and multi-criteria decision making (MCDM) to list non-dominated solutions with various parameters such as distance, risk, congestion, and risk. Numerical illustrations prove that the combination of the Dijkstra algorithm and MCDM can yield high-quality results. Ojekudo and Akpan (2017) applied the Dijkstra algorithm to determine transportation routes for the paint industry. The results indicated that the company can reduce travel distance if they implement the solution. Salem et al. (2022) applied the Dijkstra algorithm to schedule flights with a comparison of route findings using simulation environments. Experimental results revealed that simulations can help find the shortest path within a few seconds. Tawanda (2013) developed a non-iterative optimal tree algorithm. The algorithm transforms the network into a tree through arc and node replications, thus determining the shortest branch from the source to the destination node. Numerical illustrations were used to prove the validity and applicability of the algorithm. Maposa et al. (2014) presented a non-iterative algorithm for finding the shortest route. Twelve criteria were used to compare the algorithm to other algorithms such as the Dijkstra algorithm and the dynamic algorithm, among others. Tawanda (2018) computed *k* possible critical paths in the project network using Tawanda's non-iterative optimal tree algorithm for the shortest path problem. The algorithm was compared with the Critical Path Method (CPM) and the modified Dijkstra's algorithm. Hasan et al. (2007) considered a genetic algorithm for SPP. Henzinger et al. (2014) proposed a sub-quadratic algorithm for SPP.

Liang et al. (2021) improved the Ant colony optimization (ACO) algorithm to improve tourism route planning. Factors such as weather and comfort degree were considered in the shortest path computations. The results revealed that the route computed improved the tourist experience as well as reducing path length by 20.8%. Zhang et al. (2021) applied an improved multi-objective Ant colony optimization (ACO) algorithm to solve the ship weather routing problem. Factors such as sailing time, fuel consumption, and navigation safety were considered. Simulations were used to prove the effectiveness of the ACO algorithm. Akdas et al. (2021) considered route optimization for solid waste management in Maltepe, Istanbul, using the ACO algorithm. A better path was determined with 13% efficiency compared to the existing path. Deng et al. (2012) proposed Fuzzy Dijkstra's algorithm to compute the shortest distances in uncertain environments. A numerical illustration was used to demonstrate how the algorithm iterates as it computes the shortest path in an uncertain environment. Fuzzy Dijkstra's algorithm was evaluated on a 23-node directed network, and the results obtained were the same as those obtained by Agarana et al., (2016). D'Emidio (2020) presented two dynamic algorithms to compute the shortest routes in massive time evolving networks. The algorithms were tested on synthetic and real-world networks and proved to be good algorithms in terms of the solutions computed. Kumar et al. (2022) proposed the use of virtual directions to reduce computational complexity in non-directed networks. Lewis (2020) developed algorithms that determine the shortest routes in graphs with penalties due to vertex transfer. The idea of the algorithms is based on Dijkstra's algorithm. These variants are so useful to model situations like public transport transfers and junction delays. Nguyen et al. (2022) considered tubular space shortest path problems and developed an algorithm to solve tubular space shortest path problems. Computed solutions were better when compared to Dijkstra's algorithmic solutions in terms of path length, smoothness, accuracy, and algorithm calculation speed.

This paper has been organized into seven sections. The TANYAKUMU labelling method for TSP is presented in Section 2, and the Extended TANYAKUMU labelling method for computing the shortest path problem in directed transportation networks is presented in Section 3. The minimum incoming weight label (MIWL) algorithm for SPP is presented in Section 4. Numerical illustrations are presented in Section 5. In Section 6, comparative analysis is presented, and lastly, concluding remarks and further research suggestions are presented in Section 7.

1.1 Motivation

Several applications of the shortest path problem have motivated the development of alternative algorithms for computing the shortest path and its respective distance. These applications call for new methods that can solve the problem in a better way. Many algorithms in the literature are heuristic algorithms that compute near-optimal solutions to the problem (Wayahdi et al., 2021). In this research, we are motivated to develop an exact algorithm for the shortest path problem. The complexity of existing methods has a negative bearing on the teaching side of the algorithms; thus, most algorithms are software-dependent, and numerical illustrations cannot be used to demonstrate how the algorithms iterate as they compute the required solutions. TSP-based algorithms have never been extended to solve the shortest path problem; hence, in this research, we are motivated to solve the SPP using the TSP-based algorithms, thus integrating TSP and SPP.

2. TANYAKUMU Labelling Method

In this paper, we extend the TANYAKUMU labelling method for TSP to solve SPP. This method was proposed in 2022 to solve the TSP. The method makes use of node labels as it computes the optimal tour. The algorithm is simple and straight forward hence, it can be used for teaching purposes. In a network with V nodes, the algorithm requires a V-1 number of iterations to compute all possible optimal tours if there is more than one optimal tour in the travelling salesman network. The algorithm was later modified by Tawanda et al., (2023) to solve a variant of the TSP, namely the Equality generalized travelling salesman problem (E-GTSP). The algorithm given in Algorithm 1 Table 1 gives the notations used in Algorithm 1 and the proposed Algorithm 2.

Algorithm 1: Modified TANYAKUMU labelling method to solve E-GTSP.

```
Initialization: Label all vertices with label $\boldsymbol{w_{(1,j)^z}}$ for all $\boldsymbol{j \in V = V_1 \cup V_2 \;...\; \cup V_m}$. $\boldsymbol{w^*_{(1,j)^0} = w_{(1,j)^0}}$.
*Step* 1. For all $\boldsymbol{j \in w^*_{(1,j)^z}}$ and $\boldsymbol{V_i}$ where $\boldsymbol{i = 1, 2, 3, \;...\;, m}$
        Label $\boldsymbol{V_i}$ with label $(\infty)$
        Else label $\boldsymbol{V_i}$ with label $\boldsymbol{w_{(1,j)^{z+1}}}$
        $\boldsymbol{w_{(1,j)^{z+1}} = w^*_{(1,k)^z} + c_{(k,j)}}$.
*Step* 2. For all $\boldsymbol{j \in V_i}$ where $\boldsymbol{i = 1, 2, 3, \;...\;, m}$
        Compute $\boldsymbol{w^*_{(1,j)^{z+1}} = Min\{w_{(1,j)^{z+1}}\}}$
*Step* 3. If $\boldsymbol{z < m - 1}$. Assign $\boldsymbol{j \to k}$ and go to step 1
        Else if $\boldsymbol{z = m - 1}$ => all $\boldsymbol{V_i}$ are visited exactly once. Go to step 4.
*Step* 4. Return to $\boldsymbol{j = 1}$(Home city) and go to step 2, then STOP.
Tour corresponding to the minimum weight label $\boldsymbol{w^*_{(1,j)^{z=m}}}$ is the minimum weight tour.
Stop.
```

Table 1. Notations for the TANYAKUMU labelling method.

Notation	Description
V	Number of vertices or nodes in the network
V_i	Cluster i
L_0	Set of all visited nodes
L^*_0	Home city set
$w_{(s,j)^z}$	Label for node j showing shortest distance from source node or home s to j through z number of intermediate nodes or iterations and w is the shortest distance
$w^*_{(s,k)^z}$	Minimum weight node label for the currently added node k
∞	Restriction node label
$UB^*_{(s,j)}$	Previously best upper bound label for node j

3. Extended TANYAKUMU Labelling Method for Solving Shortest Path Problems

Algorithm 2 gives the steps of the proposed method. This method is capable of computing the exact shortest distances and respective shortest paths from the source node to all other nodes in the network. The principle of the algorithm is to determine the shortest paths to any node in the network by comparing the distances found to the known upper bound and updating it as the algorithm iterates.

Algorithm 2: The proposed method: Extended TANYAKUMU labelling method for SPP.

```
Step 1. Initialization
        Label all vertices directly connected to the start node (s) with the node label $w_{(s,j)^z}$
Set $w_{(s,j)^z} = w^*_{(s,j)^z} = UB^*_{(s,j)}$
Step 2. For all $w^*_{(s,j)^z}$
 Set $j \rightarrow k$
If $k$ is directly connected to $j$
Label $j$ with $w_{(s,j)^{z+1}}$
$w_{(s,j)^{z+1}} = w^*_{(s,k)^z} + c(k,j)$
Else if $k$ is not directly connected to $j$
Label $j$ with $(\infty)$
Step 3. Compute $w^*_{(s,j)^{z+1}} = Min\{w_{(s,j)^{z+1}}\}$
If  $w^*_{(s,j)^{z+1}} \leq UB^*_{(s,j)}$ for some $j$
Set $w^*_{(s,j)^{z+1}} = UB^*_{(s,j)}$  and,
        Consider corresponding $j$ and go to Step 2
          Else if $w^*_{(s,j)^{z+1}} > UB^*_{(s,j)}$ for all $j$
        Go to step 4.
Step 4. Shortest distance to node $j$ is given by $UB^*_{(s,j)}$ found during the search. Then,
Stop.
```

Theorem 1. Algorithm 2 has a worst-case time complexity of $O(n^3)$.
Proof. It is easy to note that at each iteration, the algorithm requires operations that are within the range $0 <$ Operations $\leq n^2$. The algorithm terminates after a number of iterations that are way less than n. It then follows that the worst time complexity of Algorithm *2* is $O(n^3)$.

4. The Minimum Incoming Weight Label Algorithm

The minimum incoming weight label (MIWL) algorithm was proposed by Munapo et al. (2008) to compute the shortest paths in directed networks. The algorithm was later modified by Kumar et al. (2013) to determine the shortest paths in undirected networks. Given in Algorithm 3 below are the steps of the MIWL algorithm. The proposed algorithm in this paper is compared with the MIWL algorithms on problem instances of different complexities. MIWL is used for comparison purposes because the algorithm is simple and computes exact solutions to SPP.

Algorithm 3: Minimum Incoming Weight Label (MIWL) Algorithm.

```
Step 1. Set $\boldsymbol{k} \leftarrow \mathbf{2}$, label source '0'
Step 2. Compute the minimum weight $\boldsymbol{w_k}$ for node $\boldsymbol{k}$
$\boldsymbol{w_k = Min\{w_{l1,k}, w_{l2,k}, \ . \ . \ . \ , w_{lk,k}\}}$ where $\boldsymbol{w_{l1,k}, w_{l2,k}, \ . \ . \ . \ , w_{lk,k}}$ are the weights associated with $\boldsymbol{lk}$ incoming links to node $\boldsymbol{k}$. Using the minimum weight, modify associated with all the incoming activities to and outgoing activities from node $\boldsymbol{k}$.
Step 3. If $\boldsymbol{k < n - 1}$, then set $\boldsymbol{k \leftarrow k + 1}$ and go to Step 2
Step 4. All nodes have been labelled except for the sink node $\boldsymbol{n}$. This implies that we can compute the minimum weight $\boldsymbol{w_n}$ for the sink node. Use $\boldsymbol{w_n}$ to modify the weights of incoming links to the sink node by subtracting $\boldsymbol{w_n}$ from their weights. The minimum weight recorded at the sink node is the shortest distance along the shortest path from the source node to the sink node.
Step 5. Backtracking is used to trace all the links in the shortest paths through the nodes with a weight of zero after modifications. The shortest path from the source node to the sink node is unique if and only if each node has only one link with a zero modified weight link entering that node. Else alternative paths exist and can be determined during backtracking.
```

5. Numerical Illustrations

To demonstrate how the proposed method computes the shortest paths from the source node to all other nodes in the network, we present two illustrative examples with detailed solution procedures. The numerical illustrations prove that the proposed algorithm can be used for teaching purposes since it is easy to follow.

5.1 Example

Consider the following: The 7-node network in Figure 1 was used by Srinivasan (2017) to demonstrate how Dijkstra's algorithm computes the shortest distance from source node 1 to target node 7.

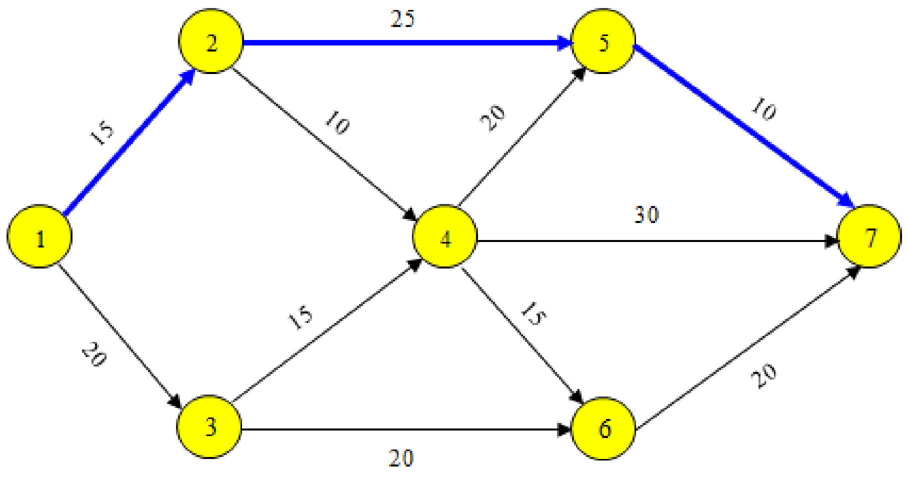

Figure 1. Seven nodes transportation network.

5.2 Solution

Initialization: Connect the source node 1 to all other nodes that are directly connected to it by labelling the neighborhood nodes, thus nodes 2 and 3 by node labels $15_{(1,2)^0}$ and $20_{(1,3)^0}$ respectively. The upper bound node labels corresponding to all visited nodes are given by $w^*{}_{(1,j)^0} = \{15_{(1,2)^0}, 20_{(1,3)^0}\}$ and to all unvisited nodes is (∞). Assign $j \to k$ and go to the next iteration.

***Iteration* 1:** Consider updating labels $w^*{}_{(1,k)^0}$ to find new labels for the nodes, thus $w_{(1,j)^1}$ as shown in Table 2 below. The upper bound distances and corresponding paths are given by $w^*{}_{(1,j)^1} = \{25_{(1,2,4)^1}, 40_{(1,2,5)^1}, \text{and } 40_{(1,3,6)^1}\}$. Consider all node labels as input for the next iterations that satisfies the condition $w^*{}_{(1,j)^1} \leq UB^*_{(1,j)}$. Assign $j \to k$ and go to the next iteration.

Table 2. Modified link distances after iteration 1.

$w^*{}_{(1,k)^0}$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$
$15_{(1,2)^0}$	∞	∞	$25_{(1,2,4)^1}$	$40_{(1,2,5)^1}$	∞	∞
$20_{(1,3)^0}$	∞	∞	$35_{(1,3,4)^1}$	∞	$40_{(1,3,6)^1}$	∞
$w^*{}_{(1,j)^1}$	∞	∞	$25_{(1,2,4)^1}$	$40_{(1,2,5)^1}$	$40_{(1,3,6)^1}$	∞

***Iteration* 2:** Consider updating labels $w^*{}_{(1,k)^1}$ to find new labels for the nodes, thus $w_{(1,j)^2}$ as shown in Table 3 below. The upper bound distances and corresponding paths are given by $w^*{}_{(1,j)^2} = \{45_{(1,2,4,5)^2}, 40_{(1,2,4,6)^2}, 50_{(1,2,5,7)^2}\}$. Consider all node labels as input for the next iterations that satisfy the condition $w^*{}_{(1,j)^1} \leq UB^*_{(1,j)}$ except the sink node label. Using this condition, the upper bound node labels for nodes 5 and 7 are discarded, and upper bound node label for node 6 is considered as input for the next iteration. Assign $j \to k$ and go to the next iteration.

Table 3. Modified link distances after iteration 2.

$w^*{}_{(1,k)^1}$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$
$25_{(1,2,4)^1}$	∞	∞	∞	$45_{(1,2,4,5)^2}$	$40_{(1,2,4,6)^2}$	$55_{(1,2,4,7)^2}$
$40_{(1,2,5)^1}$	∞	∞	∞	∞	∞	$50_{(1,2,5,7)^2}$
$40_{(1,3,6)^1}$	∞	∞	∞	∞	∞	$60_{(1,3,6,7)^2}$
$w^*{}_{(1,j)^2}$	∞	∞	∞	$45_{(1,2,4,5)^2}$	$40_{(1,2,4,6)^2}$	$50_{(1,2,5,7)^2}$

***Iteration* 3:** Consider updating labels $w^*{}_{(1,k)^2}$ to find new labels for the nodes, thus $w_{(1,j)^3}$ as shown in Table 4 below. The upper bound distances and corresponding paths are given by $w^*{}_{(1,j)^2} = \{60_{(1,2,4,6,7)^3}\}$. At this iteration we only have one node labels that corresponds to the sink node; this implies that we do not have any node label that can be used as input for the next iteration. As a result, the algorithm terminates, and all shortest paths form the source to other nodes in the network have been computed.

Table 4. Modified link distances after iteration 3.

$w^*{}_{(1,k)^2}$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$
$40_{(1,2,4,6)^2}$	∞	∞	∞	∞	∞	$60_{(1,2,4,6,7)^3}$
$w^*{}_{(1,j)^3}$	∞	∞	∞	∞	∞	$60_{(1,2,4,6,7)^3}$

The best upper bound node labels to nodes 2, 3, 4, 5, 6 and 7 give the shortest path, and the respective shortest distance. The best upper bound node label for node 2 and 3 are given by $15_{(1,2)^0}$ and $20_{(1,3)^0}$, and was computed in the initialization step. The best upper bound for node 4 is given by $25_{(1,2,4)^1}$, and was found in iteration 1. The best upper bound node label for node 5 is given by $40_{(1,2,5)^1}$, and was found on iteration 1. The best upper bound node labels for node 6 are given by $40_{(1,3,6)^1}$ and $40_{(1,2,4,6)^2}$, computed on iterations 1 and 2, respectively. Lastly the best upper bound node label for node 7 is given by $50_{(1,2,5,7)^2}$, computed on iteration 2. Table 5 below shows all shortest paths from node 1 to all other nodes in the network.

Table 5. Computed single source shortest paths.

Start node	Target node	Shortest path	Path length
1	2	(1) → (2)	15
1	3	(1) → (3)	20
1	4	(1) → (2) → (4)	25
1	5	(1) → (2) → (5)	40
1	6	(1) → (3) → (6) or (1) → (2) → (4) → (6)	40
1	7	(1) → (2) → (5) → (7)	50

This solution is optimal and is the same as the one obtained by Srinivasan, (2017). The only difference is that the algorithm used fewer iterations to compute all the shortest paths in the network.

5.3 Example 2

An eight-node network in Ojekudo and Akpan, (2017) is considered. Nodes 1 and 8 are the source and destination nodes, respectively. All other nodes in the network are intermediate nodes. Figure 2 below shows the eight-node network.

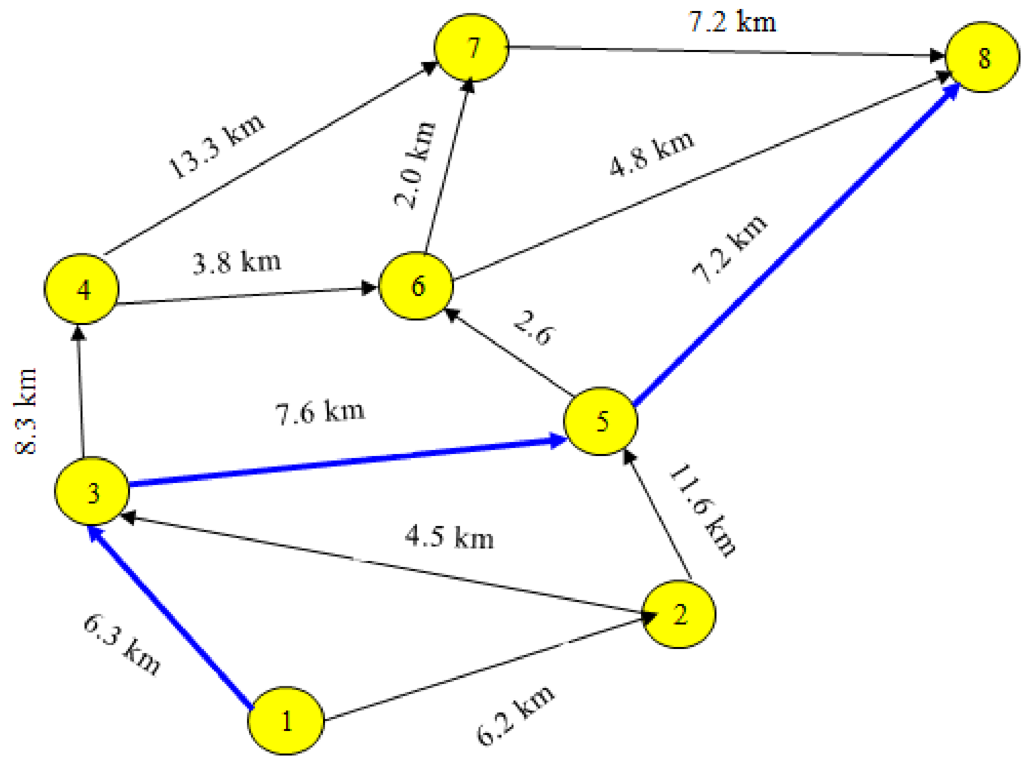

Figure 2. Eight nodes transportation network.

5.4 Solution

***Initialization step*:** Connect the source node 1 to all other nodes that are directly connected to it by labelling the neighborhood nodes, thus nodes 2 and 3 by node labels $6.2_{(1,2)^0}$ and $3.6_{(1,3)^0}$ respectively. The upper bound node labels corresponding to all visited nodes are given by $w^*{}_{(1,j)^0} = \{6.2_{(1,2)^0}, 3.6_{(1,3)^0}\}$ and to all unvisited nodes is (∞). Assign $j \to k$ and go to the next iteration.

***Iteration* 1:** Consider updating labels $w^*{}_{(1,k)^0}$ to find new labels for the nodes, thus $w_{(1,j)^1}$ as shown in Table 6 below. The upper bound distances and corresponding paths are given by $w^*{}_{(1,j)^1} = \{10.7_{(1,2,3)^1}, 14.6_{(1,3,4)^1}, \text{and } 13.9_{(1,3,5)^1}\}$. Consider all node labels as input for the next iterations that satisfies the condition $w^*{}_{(1,j)^1} \leq UB^*_{(1,j)}$. Using this condition, the upper bound node label for node 3 on this iteration is discarded, and upper bound node labels for nodes 4 and 5 is considered as input for the next iteration. Assign $j \to k$ and go to the next iteration.

Table 6. Modified link distances after iteration 1.

$w^*{}_{(1,k)^0}$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
$6.2_{(1,2)^0}$	∞	$10.7_{(1,2,3)^1}$	∞	$17.8_{(1,2,5)^1}$	∞	∞	∞
$3.6_{(1,3)^0}$	∞	∞	$14.6_{(1,3,4)^1}$	$13.9_{(1,3,5)^1}$	∞	∞	∞
$w^*{}_{(1,j)^1}$	∞	$10.7_{(1,2,3)^1}$	$14.6_{(1,3,4)^1}$	$13.9_{(1,3,5)^1}$	∞	∞	∞

***Iteration* 2:** Consider updating labels $w^*{}_{(1,k)^1}$ to find new labels for the nodes, thus $w_{(1,j)^2}$ as shown in Table 7 below. The upper bound distances and corresponding paths are given by $w^*{}_{(1,j)^2} = \{16.5_{(1,3,5,6)^2}, 27.9_{(1,3,4,7)^2}, \text{and } 21.1_{(1,3,5,8)^2}\}$. Consider all node labels as input for the next iterations that satisfies the condition $w^*{}_{(1,j)^2} \leq UB^*_{(1,j)}$ except the destination node. Using this condition, the upper bound node label for node 8 on this iteration is discarded, and upper bound node labels for nodes 6 and 7 is considered as input for the next iteration. Assign $j \to k$ and go to the next iteration.

Table 7. Modified link distances after iteration 2.

$w^*{}_{(1,k)^1}$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
$14.6_{(1,3,4)^1}$	∞	∞	∞	∞	$18.4_{(1,3,4,6)^2}$	$27.9_{(1,3,4,7)^2}$	∞
$13.9_{(1,3,5)^1}$	∞	∞	∞	∞	$16.5_{(1,3,5,6)^2}$	∞	$21.1_{(1,3,5,8)^2}$
$w^*{}_{(1,j)^2}$	∞		∞	∞	$16.5_{(1,3,5,6)^2}$	$27.9_{(1,3,4,7)^2}$	$21.1_{(1,3,5,8)^2}$

***Iteration* 3:** Consider updating labels $w^*{}_{(1,k)^2}$ to find new labels for the nodes, thus $w_{(1,j)^3}$ as shown in Table 8 below. The upper bound distances and corresponding paths are given by $w^*{}_{(1,j)^2} = \{18.5_{(1,3,5,6,7)^3} \text{ and } 21.3_{(1,3,5,6,8)^3}\}$. Consider all node labels as input for the next iterations that satisfies the condition $w^*{}_{(1,j)^3} \leq UB^*_{(1,j)}$ except the destination node. Using this condition, the upper bound node label for node 8 on this iteration is discarded and upper bound node label for node 7 is considered as input for the next iteration. Assign $j \to k$ and go to the next iteration.

Table 8. Modified link distances after iteration 3.

$w^*{}_{(1,k)^1}$	$j = 2$	$j = 3$	$j = 4$	$j = 5$	$j = 6$	$j = 7$	$j = 8$
$16.5_{(1,3,5,6)^2}$	∞	∞	∞	∞	∞	$18.5_{(1,3,5,6,7)^3}$	$21.3_{(1,3,5,6,8)^3}$
$27.9_{(1,3,4,7)^2}$	∞	∞	∞	∞	∞	∞	$35.1_{(1,3,4,7,8)^3}$
$w^*{}_{(1,j)^3}$	∞	∞	∞	∞	∞	$18.5_{(1,3,5,6,7)^3}$	$21.3_{(1,3,5,6,8)^3}$

***Iteration* 4**: Consider updating labels ${w^*}_{(1,k)^3}$ to find new labels for the nodes, thus $w_{(1,j)^4}$ as shown in Table 9 below. The upper bound distance and corresponding path is given by ${w^*}_{(1,j)^4} = \{25.7_{(1,3,5,6,7,8)^4}\}$. At this iteration we only have one node label that corresponds to the sink node 8, this implies that we do not have any node label that can be used as input for the next iteration. As a result, the algorithm terminates and all shortest paths form the source to other nodes in the network have been computed.

Table 9. Modified link distances after iteration 4.

${w^*}_{(1,k)^1}$	$j=2$	$j=3$	$j=4$	$j=5$	$j=6$	$j=7$	$j=8$
$18.5_{(1,3,5,6,7)^3}$	∞	∞	∞	∞	∞	∞	$25.7_{(1,3,5,6,7,8)^4}$
${w^*}_{(1,j)^4}$	∞	∞	∞	∞	∞	∞	$25.7_{(1,3,5,6,7,8)^4}$

The best upper bound node labels for nodes 2, 3, 4, 5, 6, 7 and 8 give the shortest path, and the respective shortest distance. The best upper bound node label for node 2 and 3 are given by $6.2_{(1,2)^0}$ and $6.3_{(1,3)^0}$, and were computed in the initialization step. The best upper bound node label for node 4 is given by $14.6_{(1,2,4)^1}$, and was computed on iteration 1. The best upper bound node label for node 5 is given by $13.9_{(1,3,5)^1}$, and was computed on iteration 1. The best upper bound node label for node 6 is given by $16.5_{(1,3,5,6)^2}$, computed on iteration 2. The best upper bound node label for node 7 is given by $18.5_{(1,3,5,6,7)^3}$ computed on iteration 3. Lastly, the best upper bound node label for node 8 is given by $21.1_{(1,3,5,8)^2}$ computed on iteration 2. Table 10 below shows all shortest paths from node 1 to all other nodes in the network.

Table 10. Computed single source shortest paths.

Start node	Target node	Shortest path	Path length
1	2	(1) → (2)	6.2
1	3	(1) → (3)	6.3
1	4	(1) → (2) → (4)	14.6
1	5	(1) → (3) → (5)	13.9
1	6	(1) → (3) → (5) → (6)	16.5
1	7	(1) → (3) → (5) → (6) → (7)	18.5
1	8	(1) → (3) → (5) → (8)	21.1

The computed shortest paths are optimal and are the same as the ones in Ojekudo and Akpan (2017). The proposed algorithm used fewer iterations to determine all the shortest paths in the network.

6. Comparative Analysis

In order to demonstrate the robust performance of the Extended TANYAKUMU labelling method, comparative analysis is considered in this section. The proposed method was applied to four increasingly complex problems. The results obtained manually are compared with the results obtained by MIWL algorithm. This was done to prove the efficiency and validity of the method. We considered 11×11 transportation network in Di Caprio et al. (2022), 21×21 transportation network in Agarana et al. (2016), 23×23 transportation network in Akram et al. (2021) and 26×26 random network in Figure 5. Comparative analysis focused on the shortest path computed, and the respective shortest distance obtained. We managed to do comparisons up to 26 nodes network due to the fact that all computations are computed manually. The networks for comparative analysis are given in Figures 3-6 below.

***Network* 1:** Consider a transportation network in Figure 3 below with 11 nodes and 25 links. Node (1) is the starting point and node (11) is the target node. Link weights are given in Table 11.

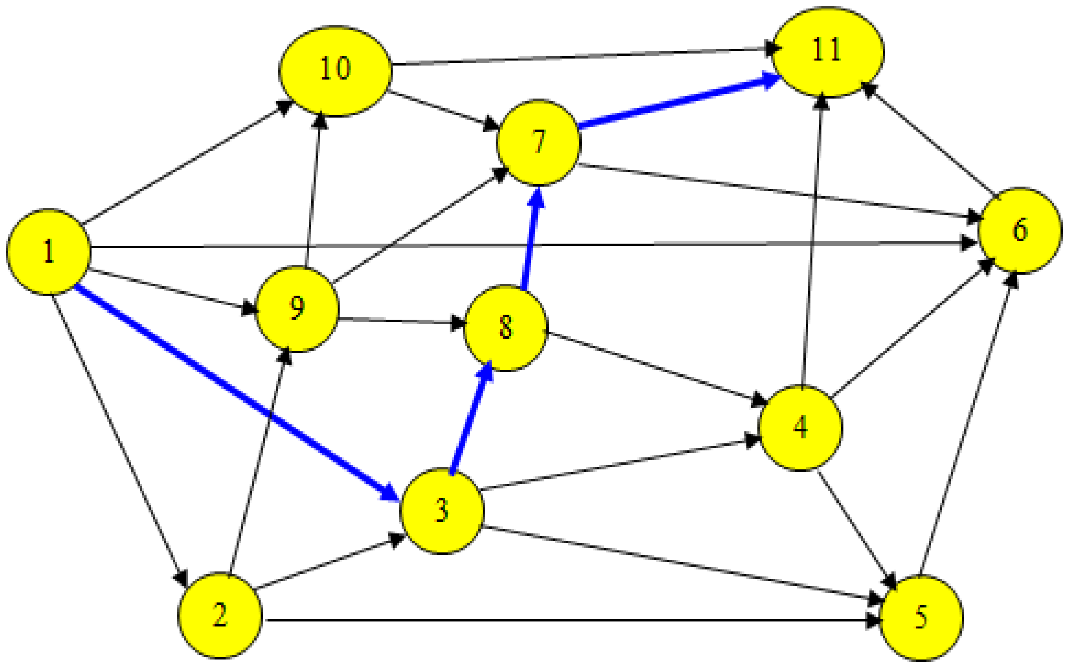

Figure 3. Eleven nodes transportation network.

Table 11. Link costs for network 1 in Figure 3.

Link	Cost	Link	Cost	Link	Cost	Link	Cost	Link	Cost
(1, 2)	800	(1, 3)	35	(1, 6)	650	(1, 9)	290	(1, 10)	420
(2, 3)	180	(2, 5)	495	(2, 9)	90	(3, 4)	650	(3, 5)	730
(3, 8)	42	(4, 5)	190	(4, 6)	310	(4, 11)	71	(5, 6)	610
(6, 11)	23	(7, 6)	390	(7, 11)	45	(8, 4)	710	(8, 7)	230
(9, 7)	120	(9, 8)	13	(9, 10)	23	(10, 7)	330	(10, 11)	125

***Network* 2:** Consider a transportation network in Figure 4 below with 21 nodes and 40 links. Node (A) is the starting point and node (U) is the target node. Link weights are given in Table 12.

Table 12. Link costs for network in Figure 4.

Link	Cost	Link	Cost	Link	Cost	Link	Cost	Link	Cost
(A, B)	270	(A, C)	250	(A, D)	2000	(B, I)	1800	(C, B)	1800
(C, H)	290	(C, I)	2000	(D, C)	750	(D, E)	2700	(E, C)	450
(E, F)	1200	(F, G)	7000	(F, L)	500	(G, H)	1300	(H, I)	2500
(H, J)	2500	(I, J)	2000	(J, O)	270	(K, H)	500	(K, J)	2200
(K, M)	900	(L, G)	350	(L, K)	10000	(L, M)	1500	(L, Q)	400
(M, N)	300	(M, P)	450	(N, J)	2000	(N, O)	3200	(O, T)	3200
(P, O)	2600	(P, S)	180	(Q, M)	550	(Q, P)	1000	(Q, R)	700
(R, S)	400	(R, U)	550	(S, T)	480	(S, U)	500	(U, T)	270

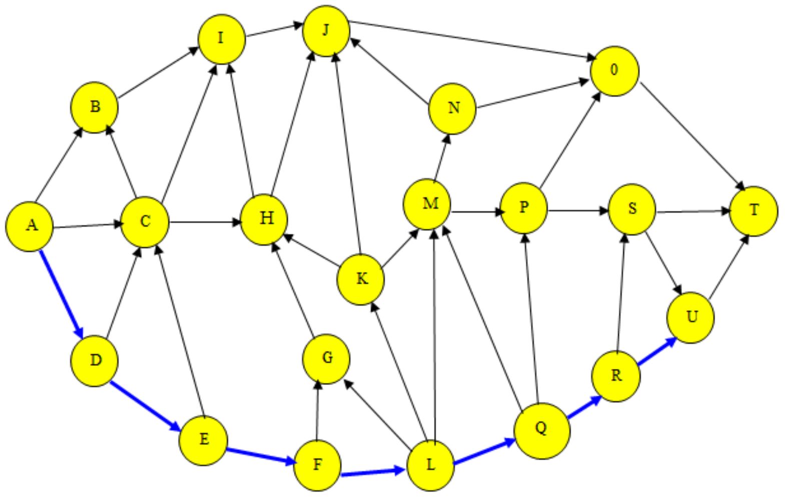

Figure 4. Twenty-one nodes transportation network.

***Network* 3:** Consider a transportation network in Figure 5 below with 23 nodes and 40 links. Node (1) is the starting point and node (23) is the target node. Link costs are given in Table 13.

Table 13. Link costs for network in figure 5.

Link	Cost	Link	Cost	Link	Cost	Link	Cost	Link	Cost
(1, 2)	12	(1, 3)	9	(1, 4)	8	(1, 5)	7	(2, 6)	5
(2, 7)	6	(3, 8)	10	(4, 7)	17	(4, 11)	6	(5, 8)	6
(5, 11)	7	(5, 12)	10	(6, 9)	6	(6, 10)	10	(7, 10)	9
(7, 11)	6	(8, 12)	5	(8, 13)	3	(9, 16)	6	(10, 16)	12
(10, 17)	15	(11, 14)	8	(11, 17)	6	(12, 14)	13	(12, 15)	12
(13, 15)	10	(13, 19)	17	(14, 21)	11	(15, 18)	8	(15, 19)	5
(16, 20)	9	(17, 20)	7	(17, 21)	6	(18, 21)	15	(18, 22)	3
(18, 23)	5	(19, 22)	15	(20, 23)	13	(21, 23)	12	(22, 23)	4

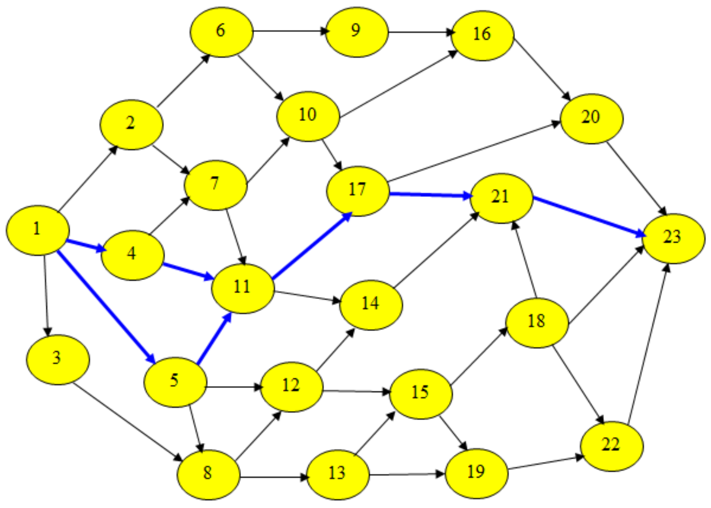

Figure 5. Twenty-three nodes transportation network.

***Network* 4:** Consider a transportation network in Figure 6 below with 26 nodes and 49 links. Node (A) is the starting point and node (Z) is the target node. Link costs are given in Table 14.

Table 14. Link costs for network in Figure 6.

Link	Cost	Link	Cost	Link	Cost	Link	Cost	Link	Cost
(A, B)	1	(A, G)	8	(B, C)	2	(B, H)	4	(C, D)	3
(C, H)	1	(D, E)	8	(D, J)	5	(D, I)	3	(E, F)	1
(E, K)	9	(F, K)	2	(G, H)	2	(G, L)	2	(H, L)	2
(H,M)	5	(I, H)	4	(I, M)	2	(J, I)	5	(J, M)	1
(J, O)	3	(K, J)	3	(K, P)	10	(L, Q)	6	(L, R)	8
(M, L)	2	(M, R)	2	(M, S)	6	(N, M)	4	(N, T)	3
(O, N)	11	(O, P)	5	(O, U)	6	(P, V)	13	(Q, R)	1
(Q, W)	8	(R, S)	2	(R, W)	15	(S, T)	5	(S, Y)	2
(T, O)	1	(U, Y)	1	(U, Z)	3	(V, U)	1	(V, Z)	1
(W, X)	7	(X, S)	1	(X, Y)	2	(Y, Z)	10		

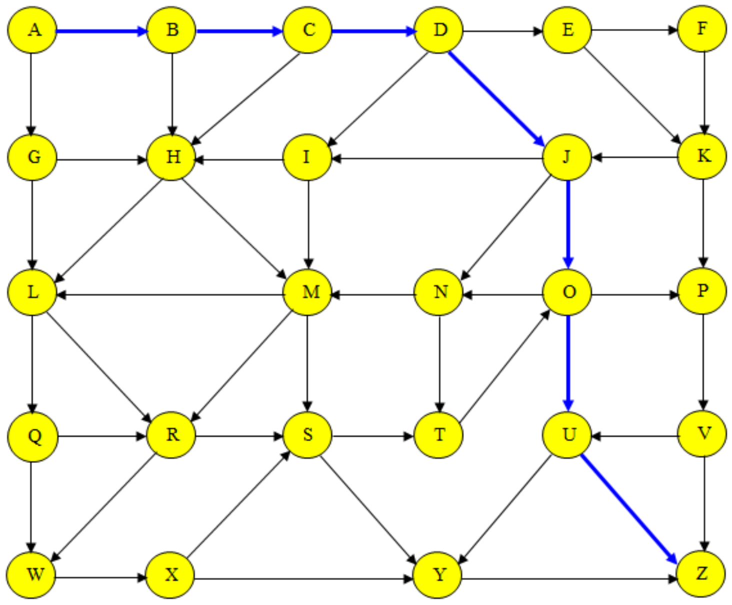

Figure 6. Twenty-six nodes transportation network.

A summary of all computational results is provided in Table 15.

Table 15. Comparative results.

Problem Source	Problem Size	Solution Methods	Shortest path	Shortest Distance	Number of Iterations
Di Caprio et al., (2022)	11 × 11	MIWL Algorithm	(1) → (3) → (8) → (7) → (11)	352	10
		Proposed Method	(1) → (3) → (8) → (7) → (11)	352	05
Agarana et al., (2016)	21 × 21	MIWL Algorithm	(A) → (D) → (E) → (F) → (L) → (Q) → (R) → (U)	8050	20
		Proposed Method	(A) → (D) → (E) → (F) → (L) → (Q) → (R) → (U)	8050	08
Akram et al., (2021)	23 × 23	MIWL Algorithm	(1) → (5) → (11) → (17) → (21) → (23)	38	22
			(1) → (4) → (11) → (17) → (21) → (23)	38	
		Proposed Method	(1) → (5) → (11) → (17) → (21) → (23)	38	06
			(1) → (4) → (11) → (17) → (21) → (23)	38	
Random graph	26 × 26	MIWL Algorithm	(A) → (B) → (C) → (D) → (J) → (O) → (U) → (Z)	23	25
		Proposed Method	(A) → (B) → (C) → (D) → (J) → (O) → (U) → (Z)	23	08

In all the problems considered for comparative analysis, it should be noted that the proposed algorithm has computed the same solutions as MIWL algorithm. The number of iterations required by the MIWL algorithm are dependent on the number of nodes, whereas the number of iterations for the Extended TANYAKUMU labelling method depends on the number of links in the shortest path. Thus, the exact number of iterations cannot be pre-determined but a general upper bound using the number of nodes in the network can be used.

7. Conclusion and Further Research

In this paper, the authors have extended the TANYAKUMU labelling method to determine the shortest path in transportation networks. In other words, a new algorithm has been developed for solving SPP, that is simple, easy to apply and can be used for teaching purpose. In this research, we have demonstrated that TSP based algorithm can be extended to solve SPP. The algorithm computes shortest paths from the source node to all other nodes in directed network. The algorithm is applicable only on directed transportation networks. We carried out comparative analysis using the problems of different sizes and complexity to proof the validity of the proposed method. Extended TANYAKUMU labelling method was used to compute optimal shortest paths manually in directed networks with up to 26 nodes, thus, this simple approach can be used to solve large problems. The computed shortest paths were compared with shortest paths computed using MIWL algorithm. The comparative analysis revealed that the Extended TANYAKUMU labelling method and the MIWL algorithm computed the same optimal solutions in all transportation networks considered in this study. Future development of this study should focus on the development of a software or a code for the proposed method to perform computational comparisons on lager network. Future studies will also consider introducing fuzzy theory to the Extended TANYAKUMU labelling method for shortest paths problem. Lastly, we will consider extending the shortest path algorithm to compute critical paths in project networks (Tawanda, 2018; Munapo et al., 2008).

Conflict of Interest

The authors confirm that there is no conflict of interest to declare for this publication.

Acknowledgements

The authors would like to acknowledge the APC funded by the North West University, South Africa. The authors would like to thank the Editor-in-Chief, section editors and anonymous reviewers for their comments that helped them to improve the quality of this work.

References

Agarana, M.C., Omoregbe, N.C., & Ogunpeju, M.O. (2016). Application of Dijkstra algorithm to proposed tramway of a potential world class university. *Applied Mathematics*, *7*, 496-503. https://doi.org/10.4236/am.2016.76045.

Akdaş, H.Ş., Demir, Ö., Doğan, B., Bas, A., & Uslu, B.Ç. (2021). Vehicle route optimization for solid waste management: A case study of maltepe, Istanbul. In *2021 13th International Conference on Electronics, Computers and Artificial Intelligence* (pp. 1-6). IEEE. Pitesti, Romania.

Akram, M., Habib, A., & Alcantud, J.C.R. (2021). An optimization study based on Dijkstra algorithm for a network with trapezoidal picture fuzzy numbers. *Neural Computing and Applications*, *33*(4), 1329-1342. https://doi.org/10.1007/s00521-020-05034-y.

Carvalho, I.A., Noronha, T.F., Duhamel, C., Vieira, L.F., & Santos, V.F.D. (2023). A fix-and-optimize heuristic for the minmax regret shortest path arborescence problem under interval uncertainty. *International Transactions in Operational Research*, *30*(2), 1120-1143.

D’Emidio, M. (2020). Faster algorithms for mining shortest-path distances from massive time-evolving graphs. *Algorithms*, *13*(8), 191.

Deng, Y., Chen, Y., Zhang, Y., & Mahadevan, S. (2012). Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment. *Applied Soft Computing*, *12*(3), 1231-1237.

Di Caprio, D., Ebrahimnejad, A., Alrezaamiri, H., & Santos-Arteaga, F.J. (2022). A novel ant colony algorithm for solving shortest path problems with fuzzy arc weights. *Alexandria Engineering Journal*, *61*(5), 3403-3415.

Edmonds, N., Breuer, A., Gregor, D.P., & Lumsdaine, A. (2006). Single-source shortest paths with the parallel boost graph library. In *The Shortest Path Problem* (pp. 219-248). Piscataway, NJ.

Hasan, B.S., Khamees, M.A., & Mahmoud, A.S.H. (2007). A heuristic genetic algorithm for the single source shortest path problem. In *2007 IEEE/ACS International Conference on Computer Systems and Applications* (pp. 187-194). IEEE. Amman, Jordan.

Henzinger, M., Krinninger, S., & Nanongkai, D. (2014). A subquadratic-time algorithm for decremental single-source shortest paths. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (pp. 1053-1072). Society for Industrial and Applied Mathematics.

Kumar, S., Munapo, E., Ncube, O., Sigauke, C., & Nyamugure, P. (2013). A minimum weight labelling method for determination of a shortest route in a non-directed network. *International Journal of System Assurance Engineering and Management*, *4*, 13-18.

Kumar, S., Munapo, E., Nyamugure, P., & Tawanda, T. (2022). Mathematics of OR: Significance and applications of virtual directions in reducing computational complexity in network optimization. In: Chauhan, I.S. (ed) *Emerging Trends in Applied Research, Integrated Publications* (pp. 33-48), Delhi, India.

Lewis, R. (2020). Algorithms for finding shortest paths in networks with vertex transfer penalties. *Algorithms*, *13*(11), 269.

Liang, S., Jiao, T., Du, W., & Qu, S. (2021). An improved ant colony optimization algorithm based on context for tourism route planning. *PLoS One*, *16*(9), e0257317. https://doi.org/10.1371/journal.pone.0257317.

Maposa, D. Mupondo, N.C., & Tawanda, T. (2014). Non-iterative algorithm for finding shortest route. *International Journal of Logistics Economics and Globalisation*, *6*(1), 56-77.

Munapo, E., Jones, B.C., & Kumar, S. (2008). A minimum incoming weight label method and its application in CPM networks. *ORiON*, *24*(1), 37-48.

Nguyen, D.V.A., Szewczyk, J., & Rabenorosoa, K. (2022). an effective algorithm for finding shortest paths in tubular spaces. *Algorithms*, *15*(3), 79. https://doi.org/10.3390/a15030079.

Ojekudo, N.A., & Akpan, N.P. (2017). Anapplication of Dijkstra’s algorithm to shortest route problem. *IOSR Journal of Mathematics*, *13*(3), 20-32.

Petrovan, A., Pop, P., Sabo, C., & Zelina, I. (2023). Novel two-level hybrid genetic algorithms based on different Cayley-type encodings for solving the clustered shortest-path tree problem. *Expert Systems with Applications*, *215*, 119372. https://doi.org/10.1016/j.eswa.2022.119372.

Rosita, Y.D., Rosyida, E.E., & Rudiyanto, M.A. (2019). Implementation of Dijkstra algorithm and multi-criteria decision-making for optimal route distribution. *Procedia Computer Science*, *161*, 378-385.

Saksena, J.P., & Kumar, S. (1966). The routing problem with “K” specified nodes. *Operations Research*, *14*(5), 909-913. https://doi.org/10.1287/opre.14.5.909.

Salem, I.E., Mijwil, M.M., Abdulqader, A.W., & Ismaeel, M.M. (2022). Flight-schedule using Dijkstra's algorithm with comparison of routes findings. *International Journal of Electrical and Computer Engineering*, *12*(2), 1675.

Srinivasan, G. (2017). *Operations research: Principles and applications*. 3rd Edition, PHI Learning Pvt. Ltd.

Tawanda, T. (2013). Tawanda's non-iterative optimal tree algorithm for shortest route problems. *Scientific Journal of Pure and Applied Science*, *2*(2), 87-94.

Tawanda, T. (2018). Determining k-possible critical paths using Tawanda's non-iterative optimal tree algorithm for shortest route problems. *International Journal of Operational Research*, *32*(3), 313-328.

Tawanda, T., Nyamugure, P., Kumar, S., Munapo, E. (2023). Modified TANYAKUMU labelling method to solve equality generalized travelling salesman problem. In: Vasant, P., Weber, GW., Marmolejo-Saucedo, J.A., Munapo, E., Thomas, J.J. (eds) *Intelligent Computing & Optimization. ICO 2022*. Lecture Notes in Networks and Systems, vol 569. Springer, Cham. https://doi.org/10.1007/978-3-031-19958-5_88.

Thorup, M. (1999). Undirected single-source shortest paths with positive integer weights in linear time. *Journal of the ACM*, *46*(3), 362-394.

Wayahdi, M.R., Ginting, S.H.N., & Syahputra, D. (2021). Greedy, A-Star, and Dijkstra's algorithms in finding shortest path. *International Journal of Advances in Data and Information Systems*, *2*(1), 45-52.

Wu, W., Hayashi, T., Haruyasu, K., & Tang, L. (2023). Exact algorithms based on a constrained shortest path model for robust serial-batch and parallel-batch scheduling problems. *European Journal of Operational Research*, *307*(1), 82-102.

Zhang, G., Wang, H., Zhao, W., Guan, Z., & Li, P. (2021). Application of improved multi-objective ant colony optimization algorithm in ship weather routing. *Journal of Ocean University of China*, *20*, 45-55.