Journal of Applied Engineering and Technological Science
Vol 5(2) 2024: 1182-1198

MODIFICATION OF LOAD CALCULATION IN THE DIJKSTRA ALGORITHM TO ACHIEVE HIGH THROUGHPUT AND LOW LATENCY ON 5G NETWORKS

Eko Kuncoro Adiyanto[*1], Sri Wahjuni[2], Hendra Rahmawan[3]
Department of Computer Science, IPB University, Bogor, Indonesia[1,2,3]
ekokuncoroadiyanto@apps.ipb.ac.id*, my_juni04@apps.ipb.ac.id, hrahmawan@apps.ipb.ac.id

Received : 22 March 2024, Revised: 25 May 2024, Accepted : 05 June 2024
**Corresponding Author*

ABSTRACT
throughput for high-resolution remote video surveillance. 5G cellular network as today's most advanced wireless technology will be the perfect match for Agriculture 4.0 requirements. In its maturity process, the 5G network requires various optimizations, one of which is by making route algorithm calculation modifications in terms of determining the best route for a data packet from a data source to a data destination. To achieve this goal, it requires research in the form of experiments using network simulator. Software Define Network (SDN) as network programmability is used to modify route in Dijkstra algorithm calculation, and run several use case that simulate 5G network characteristic. By adding bandwidth utilization and latency parameters into the routing algorithm calculations, 5G requirements such as packet loss below 1% and latency below 5ms are successfully achieved. These positive results may be further tested on real 5G networks, if in the future this research also gets positive results in testing on a real 5G network, then cellular network customers will be able to experience an increase in service quality.
***Keywords** : 5G, Bandwidth, Dijkstra, Latency, Software Defined Network*

1. Introduction

The world population is expected to reach 8 billion people in 2025, and 10 billion people in 2050 (Araujo *et al.*, 2021). This increment will have a huge impact on the need for food. The food producers, namely farmers and livestock breeders, are expected to focus more on developing agriculture that is smarter, more efficient, and more productive so that it can meet the future of human food needs. Indonesia, as the country with the 4th largest population in the world, has designed an agricultural strategy that can meet food sustainability for the future.

According to data from the Indonesian National Food Agency 2023, food imports in Indonesia are still quite high. For example, even though as an agricultural tropical country, Indonesia still has to import 3.5 million rice throughout 2023 to meet national food needs. To overcome this, modern agriculture is needed through agriculture 4.0, so that production can increase so that it can meet the food needs of the Indonesian people. KPPIP, as a part of Indonesia's Government, has several National Strategic Projects, including a program named Agriculture Digital Maturity, that is powered by agriculture 4.0 technology (KPPIP, 2007). There are three main components in agriculture digital maturity strategy:

1. Agricultural Information System
2. Agriculture e-Commerce Platforms
3. Precision Agriculture

AGRICULTURE SECTOR DIGITAL GOALS	CONNECTED Broadband, IoT	INFORMATION ACCESS Big Data, IoT, Cloud, Data Center	APPLICATIONS & PROCESSES IoT, Cloud, Data Center
AGRICULTURE INFORMATION	Farms & Fisheries have access to smart devices that are connected to the Internet	All Agriculture Data (Pests, Marketing Pricing, Farming Methods, Weather) is digitized and available online	Business & Farm Management Apps are available online as a services through Cloud SaaS
AGRICULTURE E-COMMERCE PLATFORMS	Farms and Fisheries are connected and have access to e-Commerce Platforms	Farms & Fisheries have Online Web Presence/Site connected to e-Commerce Marketplace. Access to Information about Transport Schedules	Digital Marketing Tools E-Commerce Marketplace Logistics & Support Systems
PRECISION AGRICULTURE	Smart Sensors (Environmental, Livestock, Geographic) and Machinery are connected to IoT Network	Sensor and Machinery Information is captured and analyzed through Big Data Cloud based PaaS Platform	SaaS Cloud based Farming Applications associated with the data collected from sensors e.g. Soil Sensors provide Information to Soil Salinity App

Fig. 1. Agriculture Digital Maturity Framework

These three components will then run well if they are supported by three main pillars (according to Figure 1), namely connectivity, information access, and applications and processes. In Figure 1, at the top left, broadband connectivity is the first thing mentioned, because broadband connectivity is the basis of a country's infrastructure to achieve a digital economy. This broadband connectivity can be likened to road, electricity, or canal infrastructure that must be built to connect various regions and strategic locations in a country.

Cellular-based broadband wireless connectivity is the most effective solution for broadband connectivity that offers the widest coverage and delivers very high throughput. Currently, in many countries, cellular technology operates on 2G, 4G, and 5G network generations. While the 4G network is still dominating the cellular network landscape, 5G is waiting for the moment to explode in the future. In its maturity process, the 5G network requires various optimizations, one of which is by making route algorithm calculation modifications in terms of determining the best route for a data packet from a data source to a data destination. In Agriculture 4.0, there are several different network service requirements that require different 5G network specifications. Each of different services in 5G uses different resources on the network, from radio network, to transport to the core network. This resource difference uses the network slicing method, where the configuration is done on SDN (Barakabitze *et al*., 2020).

A reliable network system layer is required, to enable the optimum 5G network system that meets the necessary specifications. The Internet Protocol (IP) transport layer is one of the critical components of this network system, as it plays an essential role in carrying Internet traffic from one BTS point to another until it reaches the core network that connects it to the world Internet network. Routing efficiency is one of the critical parameters in an IP transport network, as it determines the method for finding the most efficient path or route so that data packets sent on a network can arrive more quickly.

There is one routing algorithms are widely used today, namely Dijkstra algorithm. Dijkstra algorithm, as the shortest path algorithm to find the destination route path from the source point (Dijkstra, 1959), is the base algorithm for the OSPF network routing protocol in the Interior Gateway Protocol (IGP). Dijkstra find the shortest path to destination from a single source, not multi source (Cormen, 2009). These type of algorithm match with the network routing requirement from one single source to one single destination. Dijkstra's algorithm use general graphs with non-negative edge costs. The efficiency of Dijkstra's algorithm heavily relies on efficient priority queues (Melhorn & Sanders, 2007).

OSPF is widely used in 4G and 5G cellular networks, including LAN, WAN, and DC networks. OSPF needs to achieve fast convergence to topology changes, it requires highly scalable operation on part of OSPF to avoid routing instability (Goyal *et al.*, 2012). For serving 5G network, it's a must to improve OSPF's convergence speed as well as scalability of the 5G Networks. Software Defined Network (SDN) in 5G networks has become a new standard as it provides a platform for automation, programmability, independent devices, and intelligent and controlled distributed networks (Goransson & Black, 2016). Therefore, the use of OSPF on SDN is commonplace. However, one of the limitations of OSPF is that the Dijkstra algorithm's load calculation still only uses bandwidth configuration or reference bandwidth (Akhtarkavan & Karami, 2015), which is less relevant to 5G network needs.

There are many research that related to modification or extend the dijkstra algorithm, in order to get better performance or applying it in many different aspect of life. One studies used the latency parameter as a substitute for the bandwidth parameter in OSPF network routing load calculations (M Abdelghany *et al.*, 2022). In other research, the Dijkstra algorithm was used in a data center network load-balancing routing scheme using bandwidth utilization as a cost parameter (Adekokun *et al.*, 2017). Dijkstra's algorithm itself has been internally improved by developing a way to avoid heap in path calculation which is useful in the efficiency of sparse networks especially in road traffic networks (Huang *et al.*, 2013). Dijkstra's improvisation has also been applied to determine flexible weight values in path selection in the data storage structure (Zhang *et al.*, 2009).

Optimization of the Dijkstra algorithm is not only carried out on network routing but is also carried out on flood route routing during a disaster (Wang, 2017). Developed Dijkstra shortest path search algorithm can improve storage efficiency and reduce meaningless operation

(Chao, 2010). Extended Dijkstra algorithm for surface path planning of mobile robots improves the accuracy of the surface optimization path in single-robot single-target and multi-robot multi-target path planning tasks (Luo *et al.,* 2020). Implementation of extended Dijkstra's algorithm in SDN resulted that extended dijkstra outperforms the original dijkstra and other algorithms (Jiang *et al.,* 2014).

Dijkstra improvement in order to reduce the number of iterations and to find easily and quickly the shortest path (Kadry *et al.,* 2011). Dijkstra's improvisation on storage structure and searching area to make travel route planning more efficient (Fan & Shi 2010). Dijkstra's improvisation on the data storage structure, as well as ignoring reversed nodes and flexible weight values in path selection (Huang *et al.,* 2013). Improvisation in shortest-path determination using Node-Wise Limited Arc Interdiction (Khachiyan *et al.,* 2006). Discussion of the use of Dijkstra on Google maps with distance load calculations, traffic & delays (Lanning *et al.,* 2014). Extended Dijkstra Algorithm for Improvisation in shortest-path selection using the bidirectional search method (Noto & Sato 2000).

Engineering Fast Route Planning Algorithms for Improvisation in terms of shortest-path selection using the priority queues method (Sanders & Schultes 2007). Improvising dijkstra the exit mechanism to avoid loops and how to select vertices more optimally (Shu-Xi 2012). Dijkstra modifications to the path selection process to avoid loops (Wei *et al.,* 2019). Improvisation using value iteration and Q-learning methods for path planning 2D eight-neighbor grid map (Wenzheng *et al.,* 2019). Improvisation of Dijkstra algorithm processing using parallel computing with multi-core (Wu *et al.,* 2015). Improvisation by reforming the feature matrix of precursor node and adding a shortest path tree (Xiao & Lu 2010). Improvisation to avoid the heap process in Dijkstra, making it more suitable and efficient for use in large sparse networks (Xu *et al.,* 2007). Using the Dijkstra algorithm for path finding in spatial applications, the cost parameters used are distance, time, path capacity, and path type (Zhang *et al.,* 2009). Improvising dijkstra algorthm by the node search process using the heap pairing method (Zhang *et al.,* 2012).

Base on several research above, there are still a room for improvements to combine latency, bandwidth utilization, and bandwidth configuration into dijkstra routing algorithm load calculation. To implement this experiment, Software Defined Network (SDN) simulator is used to run the custom network programmability. It is expected that this research will be able to contribute to optimizing the 5G network. By this better network performance, it will deliver better real time monitoring sensor or high resolution remote video surveillance in agriculture 4.0 application.

2. Literature Review

2.1 Dijkstra Algorithm

The Dijkstra algorithm was introduced by Dutch computer scientist Edsger Wybe Dijkstra in 1959. Dijkstra's algorithm is a greedy algorithm that is used to solve the problem of finding the shortest graph (shortest path problem) from a route that has a direction with a different weight (edge weight) for each route. The input of this algorithm is a weighted directed graph G, and an origin s in a set of lines V. The output of this algorithm is the shortest path route from an origin point to a destination point. Dijkstra is the most popular algorithm used for many years in methods for finding the best path (shortest path) from a source to a destination. Dijkstra has become more popular than other shortestpath algorithms such as Floyd Warshal, Johnson and Bellman-Ford because Dijkstra has been used as the basis for the OSPF routing protocol on IP networks. The OSPF routing protocol has been used by almost all internet service provider networks in the world.

2.2 Software Defined Network

The 5G network is an evolution of a very complex wireless device network. The use of Software Defined Network (SDN) is a new way of how a 5G network can be managed centrally and scalably. On traditional networks, there is no freedom to customize the program of the network, because all routing protocols and configurations have been hard coded by the manufacturer. With SDN, every programmer has the freedom to improvise using the network

programmability available in SDN (Goransson & Black, 2014). This freedom makes the door for improvement and development wider open for anyone, not just network device manufacturers. Therefore, this research uses the SDN simulator, so that the Dijkstra algorithm, which has previously been hardcoded by router vendors, can now be easily modified to calculate the load load.

SDN can improve network performance in terms of network management, control and data handling (Hao *et al.,* 2014). Apart from SDN, there are several other Software Defined that can also improve the performance of other computing technologies such as Software Defined Security (SDSec), Software Defined Storage (SDS), Software Defined Infrastructure (SDI) (Goransson & Black, 2014). In general, SDN is divided into three parts, namely SDN devices or routers, SDN controllers, and SDN applications. The function of SDN Devices is only to forward packets, according to the direction of the flow table from the centralized controller. The SDN controller sends a flow table via the control plane to each SDN device using the openflow protocol (Kreutz *et al.,* 2014). SDN separates the control plane and data plane functions. The SDN application works on top of the SDN controller, to control the network using the northbound API. SDN applications can be programmed to suit the needs of how a data packet flows from one point to another. (Goransson & Black, 2014).

2.3 5G Network

There are three main services on the 5G network, where these three services can be differentiated in terms of resources according to the service needs requested by customers. These three main services are (Sutton, 2018)(Tang *et al.,* 2021) :

1. Enhanced Mobile Broadband (eMBB).
   This technology offers internet at very fast speeds, where in real conditions, the average throughput obtained by users can reach more than 100 Mbps.
2. Ultra-Reliable Low-Latency Communications (URLLC)
   For sensory devices or real-time applications, that may not require super-fast throughput, but prefer near real-time connection, 5G can provide a maximum latency or delay of up to 1 ms.
3. Massive Machine-Type Communications (mMTC)
   In the future, the number of communication devices, such as IoT, will far exceed the number of devices currently used by humans, such as cell phones or smartphones. In this case, 5G technology can process 1 million devices per 1 square km area.

2.4 IP Routing

There are various IP routing protocols that can be used, in general routing protocols are divided into two, namely static and dynamic. Dynamic routing is divided into two, namely Interior Gateway Protocol, with various types of routing protocols such as RIP, IGRP, EIGRP which are Distance based, and OSPF, IS-IS which are Link-State based. Then the second is the Exterior Gateway Protocol, the routing protocol that is popularly used is BGP (Sirika *et al.,* 2016)

The use of the OSPF IP routing protocol is widely used in current IP networks, both in 4G networks and datacentre networks. The OSPF protocol uses load parameters in the form of bandwidth configuration only. By using SDN, it is hoped that the Dijkstra algorithm, which is the basis of the OSPF protocol, can be modified by adding latency and bandwidth utilization parameters to the load calculation, so that the IP routing protocol can better suit the needs of the 5G network. (Karami & Akhtarkavan, 2015)(M Abdelghany *et al.,* 2022).

2.5 Quality of service

To get better quality experience, 5G technology increase the footprint of the underlying transport infrastructure by requiring new control plane facilities capable of supporting the specific needs characterizing end-to-end connections between mobile devices and their targets (Palmieri 2020). The fundamental point of network performance that can be used to measure the improvement of the network is the Quality of Service (QoS) parameter. Quality of Service aims to manage network resources so that the end-user experience of a session can be maximized because not all user sessions will have the same experience, therefore it is necessary to regulate

the quality of user sessions (Szigeti *et al.*, 2014). This network resource setting can be based on several parameters such as the minimum and maximum limits for delay or latency, the buffer size of a queuing method so that packet drops or packet loss do not occur, and the minimum and maximum limits for throughput speed (Bojovic *et al.*, 2022). The QoS parameters such as latency, throughput, and packet loss are obtained by a user session, whether they are in accordance with the expected limits or not. The limitations of these QoS parameters will refer to the 5G standards set by ITU-T, namely ITU-T Rec Y.3106.

3. Research Method

The method used in this research is an experimental method. Previously no one had modified the Dijkstra algorithm load calculations with a weighting combination of several network parameters.

3.1 System Design

The network topology used in this research is the simulation of SDN on 5G. The network will be controlled by ryu-controller software that acts as an SDN controller. Ryu-controller has the best SDN Controller simulator in term of latency & throughput good performance when it cover the number of routers or switches under 20 routers/switches (Albu-Salih 2022). While in this research, it used 12 routers, ryu-controller will bring best performance in testing scenario.

As seen in the Figure 2, this network topology separates the function of SDN devices into three parts, namely, Radio Access Network (RAN) access router (S11, S22, S77, S88) as a router connected to the BTS, Gateway router (S1,S2,S3,S4) as an aggregator of various routers, Core router (S33,S44,S55,S66) as a router connected to the core network device. As a control plane, the SDN controller provides a flow table to each connected RAN router, core, and gateway using a Dijkstra-based routing protocol that is modified in the load calculations. In this research, the best route between the source (H1) and destination (H3) points will be analyzed.

In this research, the load parameters calculation will use a combined weight of reference bandwidth, bandwidth utilization, and latency. The weights are adjusted to the services required in the 5G network slicing priority, namely eMBB & urLLC.

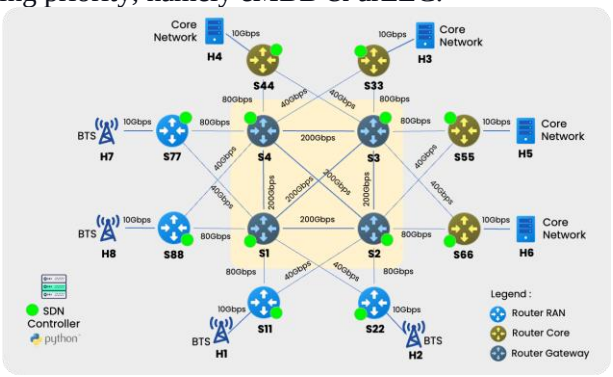

Fig. 2. SDN Topology For 5G Networks Simulation

As seen in Figure 3, higher weights are given to prioritized network slicing (An *et al.*, 2019). For load calculations, weighting will be carried out in 3 options. According to equation 1, the three load components are combined into one through a weighting scheme symbolized by x, y, and z.

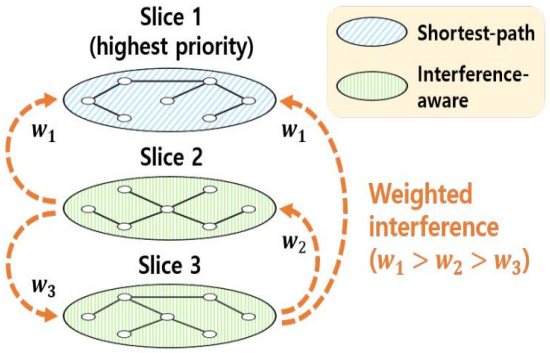

Fig. 3. Higher weights are given to prioritized network slicing

From this equation, the weighting will be carried out in 3 options. Option 1, equal weight means that each load has the same priority in network slicing. Option 2, In 5G eMBB network slicing for high throughput, bandwidth load has a higher weight than latency. Option 3, In urLLC 5G network slicing for low latency, the latency load has a higher weight than the bandwidth.

$$\text{Cost} = \left(\frac{\text{BW reference}}{\text{BW configuration}}\ (\text{x}) + \frac{\text{BW traffic}}{\text{BW configuration}}\ (\text{y}) + \text{Latency}\ (\text{z})\right) \quad (1)$$

Tabel 1 - Dijkstra Algorithm Weight Calculation Options In Equation 1

Options	x	y	z
1	33.334%	33.333%	33.333%
2	35%	35%	30%
3	30%	30%	40%

The x, y, and z values will change according to the needs of the traffic character that will be processed by network slicing. For load components with reference bandwidth, the reference bandwidth value used is 10 Tbps. This means that the larger the bandwidth configuration, the smaller the costs required.

3.2. Implementation

The implementation of the modified Dijkstra algorithm load calculation on the 5G network is carried out entirely in the form of network virtualization by SDN. The SDN emulator used in this research is Mininet version 2.3.0, and the SDN controller used Ryu Controller version 4.34, that support OpenFlow version 1.4. For further development, it is recommended to use the software version that is implemented in this research, so that the program can run well. Meanwhile, for the hardware environment, this research uses an Intel-based computer. If you want to run this program or develop it, on any type of computer as long as it is based on an Intel CPU, there should be no problem.

All of the modified Dijkstra algorithm load calculations were inserted into the Ryu controller as the main routing engine, then the best route calculation will be distributed to all network nodes in mininet. The smaller of load calculation result of the routing path, the more it will be chosen as the best routing path. Modification of the Dijkstra algorithm load calculation is carried out in a Python program running on the Ryu controller. In one of the functions in the Python program there is a function that contains how the program reads the bandwidth configured on each router interface, then this parameter is included in the calculation in the Dijkstra algorithm to find the best route. Load calculation modifications start from how to get data from bandwidth utilization on each interface, combined with the latency value obtained from the ping test sent by one router to another router.

3.3 Testing Scenarios

Testing will also be carried out with three scenarios; the first is when the network is in the best condition, where there are no links down. The second scenario is when the network is in a condition where there is one link is down. The third scenario is the worst scenario, where there

are three or more links that are down in the network. This entire scenario will be tested by two types of traffic: large and small packet size. The first stage of the test result is about how the selected paths differ, such as short length and number of hops, then continues with how long each algorithm takes (latency) to calculate.

The process required for the first test is a ping test and traceroute from source to destination. The second stage of testing results is measuring several QoS parameters, such as throughput and packet loss, using iperf. Each testing stage will be repeated ten times so that its characteristics can be analyzed.

3.4 Program Flow

Figure 4 illustrates that the program started from the SDN controller by ryu-controller. Ryu log file for routing and best path selection was starting recorded as well for every time changes happened. The program then continued to mininet as a network emulator that builds network topology simulation and then runs the iperf to send the traffic from all hosts in the network, in order to utilize the network links. At the same start time, the mininet log file for a ping test, link state, throughput iperf, and transaction duration was started as well.

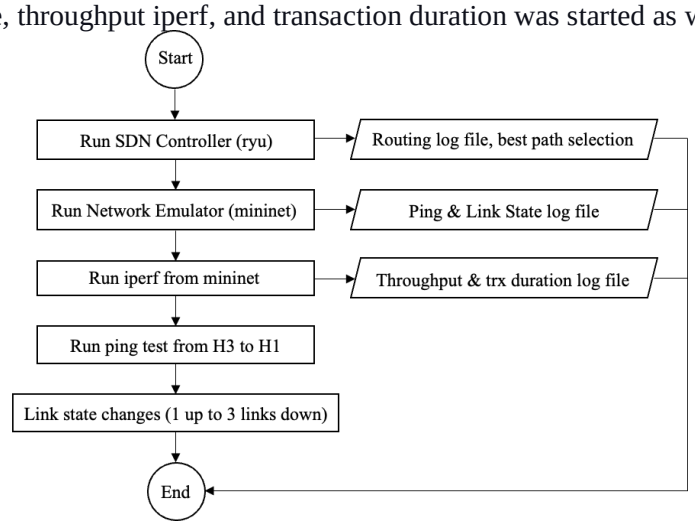

Fig. 4. Flowchart Program and Test

After the network had the traffic inside, the real test was started by running a ping test between H1 & H3, while the ping test was running, the dynamic routing test was run as well by changing the link state from 1 link, 2 links to 3 links down in the continuous time of execution. The program ended after three the links state was down, and all the logs have been captured and saved.

4. Results And Discussion

4.1 Routing Path Selection Test

The initial test of the program is to ensure that routing program can run well by carrying out a ping test from one host to another host. In each test, the host used as the source host is H1, and the host used as the destination host is H3. There are six programs that represent six testing scenarios. The six programs are :

1) Load calculation based on shortest distance
2) Load calculation based on bandwidth utilization
3) Load calculation based on latency
4) Load calculation based on a combination of bandwidth capacity/reference, bandwidth utilization & latency (all have equal weight load)
5) Load calculation based on a combination of bandwidth capacity/reference (30%), bandwidth utilization (30%) & latency (40%)
6) Load calculation based on a combination of bandwidth capacity/reference (35%), bandwidth utilization (35%) & latency (30%).

In the initial stage, a ping test is carried out on these six programs to ensure the routing function runs well. Each test will compare the number of hops, throughput/speed, latency, and

packet loss performance. Dynamic routing selection is also tested in each test by dropping the link status from 1 to 3 links to see whether the routing program still provides the best results.

The test on small packages was repeated five times, and for large packages, it was repeated five times. The following discussion will detail the results of each scenario and the testing stages. The total number of tests carried out was 60, with each duration ranging from 5 to 10 minutes. The total duration of the testing was around 600 minutes, which was executed at different times of testing. Before the routing test was recorded, every host in the network sent small and large packets to utilize the network with the traffic. This traffic occupied the network for around 10 minutes. The selection of hosts that will send packets to each other are hosts that are opposite each other, namely H1 to H3, H2 to H4, H5 to H8, and H6 to H7. After this selection, traffic can traverse various paths so that testing of routing selection can be more visible than if routing traversed adjacent hosts.

In order to deliver a more representative result, the program used in the explanation below was the load calculation based on a combination of bandwidth capacity/reference (30%), bandwidth utilization (30%) & latency (40%). In the initial test, all links were up, and all hosts then sent and received packets from H1-H3, H2-H4, H5-H8, and H6-H7. After the network is utilized with traffic, H3 performs a ping test to H1. From the test results, the best routing chosen has good routing path and performance results, because the chosen route is not circuitous and is quite effective.

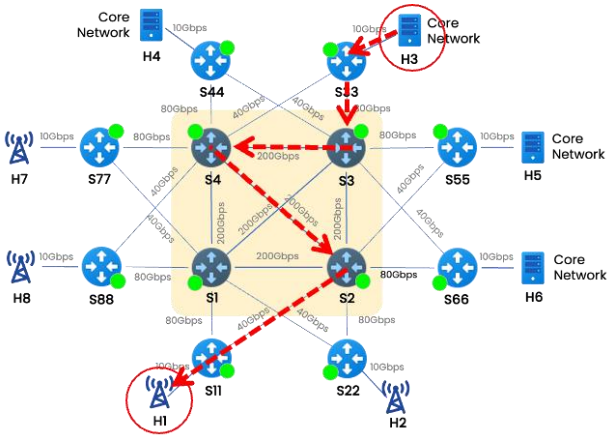

Fig. 5. Routing Path When All Link Up

Table 2 - Top 10 Best Path In All Link Up

Rank	Routing Path	Total load value
1	[33, 3, 4, 2, 11]	12.1
2	[33, 4, 2, 1, 11]	12.7
3	[33, 3, 4, 2, 1, 11]	12.7
4	[33, 3, 55, 2, 1, 11]	13.9
5	[33, 3, 4, 77, 1, 11]	14.0
6	[33, 4, 3, 55, 2, 1, 11]	14.2
7	[33, 3, 44, 4, 2, 1, 11]	14.2
8	[33, 3, 44, 4, 2, 11]	14.2
9	[33, 4, 3, 55, 2, 11]	14.6
10	[33, 3, 4, 77, 1, 22, 2, 11]	15.1

Figure 5 shows that the H3 to H1 route via S33-S3-S4-S2-S11 is the best route because, in terms of path costs, this route has the smallest load. The total number of routing combinations from H3 to H1 is 96 routes. If these 96 routes need to be displayed, then the entire network will be covered by the weight path number. As a general illustration, in Table 2, only the top 10 best routes are shown with their total path cost values when all links are in up condition. In the next routing test, three scenarios are in succession to test dynamic routing convergence: making one link down, two, and three. The test was run one by one, and every result was noted in a spreadsheet, so at the end, all the numbers could be summarized into charts.

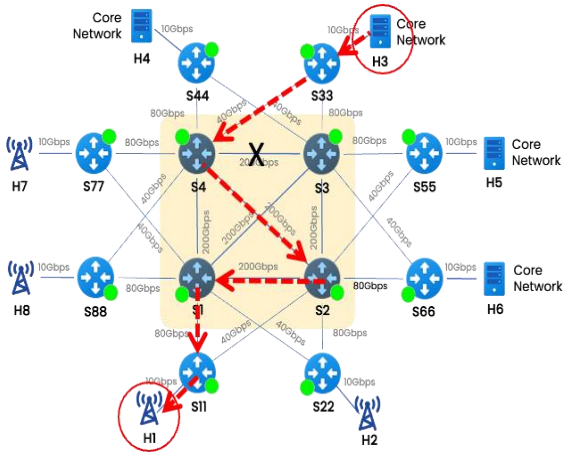

Fig. 6. Routing Path when one link down

Figure 6 shows that when the link between S3 and S4 is down, so dynamic routing will work to determine the best new path with the smallest cost. The convergence of the Dijkstra algorithm in finding this new routing takes around 20 to 25 seconds. After completing convergence, it was found that the new route with the lowest cost would be via route S33-S4-S2-S1-S11.

Table 3 - Top 10 Best Path In One Link Down

Rank	Routing Path	Total load value
1	[33, 4, 2, 1, 11]	13.77
2	[33, 3, 44, 4, 2, 11]	14.40
3	[33, 3, 55, 2, 1, 11]	14.58
4	[33, 3, 44, 4, 2, 1, 11]	15.33
5	[33, 4, 2, 11]	15.37
6	[33, 3, 44, 4, 77, 1, 11]	16.59
7	[33, 3, 44, 4, 77, 1, 22, 2, 11]	17.43
8	[33, 3, 55, 2, 11]	17.48
9	[33, 4, 77, 1, 22, 2, 11]	17.63
10	[33, 4, 77, 1, 11]	18.67

In Table 3, it is shown that the routing calculation results from H3 to H1 have a different routing table compared to routing table 1. This can happen because when the S3-S4 link is down, not only the H3 to H1 routing changes, but the entire routing between hosts also changes. Please remember that this network is sending and receiving packets from H1-H3, H2-H4, H5-H8, and H6-H7 all the time, so that if one of the links is down, then each routing will do the routing convergence to find the best path again.

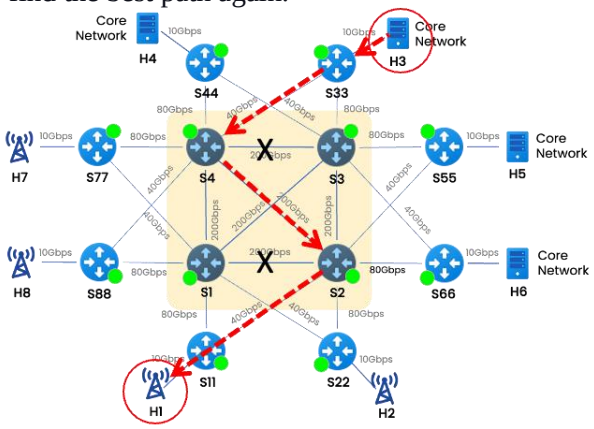

Fig. 7. Routing Path when two links down

After successfully carrying out routing convergence testing when one link is down, the next test is carried out with two links are down. Figure 7 shows that routing convergence can still be carried out successfully when two links down. It can even produce the fewest hop paths, namely only three hops. In Table 4, it can also be seen that the best path has a load of 15.32, which is greater than the load of the best path in previous Tables 1 and 2. The final basic test is

to create three downlinks, namely links S3-S4, S2-S4, and S1-S2. By the condition of the three links being down, the new path recalculated by routing convergence results in slightly circular routing to S22.

Table 4 - Top 10 Best Path In Two Links Down

Rank	Routing Path	Total load value
1	[33, 4, 2, 11]	15.32
2	[33, 4, 77, 1, 22, 2, 11]	17.19
3	[33, 3, 55, 2, 11]	18.26
4	[33, 4, 77, 1, 11]	18.37
5	[33, 4, 2, 3, 1, 11]	335.54
6	[33, 3, 1, 11]	336.41
7	[33, 3, 1, 22, 2, 11]	336.83
8	[33, 3, 1, 88, 4, 2, 11]	336.93
9	[33, 3, 2, 11]	337.11
10	[33, 4, 1, 11]	337.95

This happens because the S1-S3 link, which is shown in Figure 8, should have been selected but it was not selected. Most likely its condition has been maximally utilized by routing to and from hosts other than H1 and H3.

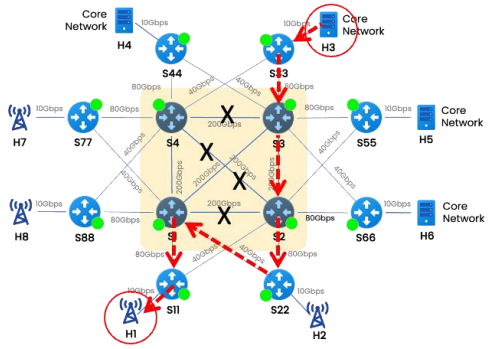

Fig. 8. Routing Path when 3 links down

This last scenario was considered the worst scenario regarding a limited number of best paths available when three links are down between H1 to H3. Overall, the testing result of these three scenarios produced good routing path results, as seen in Figure 8, the routing results when three links were down still showed that routing was quite optimal, with the total path cost on the best path still relatively small, as seen in table 5.

After going through various scenarios, both simulating conditions when the network is in good condition, and simulating when the network is experiencing a lot of interference, this routing program modification can provide the best routing selection performance. Best path selection is still produces an efficient path, it does not go round and round to all the router points in the network. The results of this good network performance will of course also have a good impact on improving user experience.

Table 5 - Top 10 Best Path 3 Links Down

Rank	Routing Path	Total load value
1	[33, 3, 2, 22, 1, 11]	12.03
2	[33, 3, 2, 11]	14.02
3	[33, 3, 55, 2, 22, 1, 11]	15.12
4	[33, 3, 55, 2, 11]	18.98
5	[33, 3, 1, 11]	334.47
6	[33, 3, 66, 2, 22, 1, 11]	336.18
7	[33, 3, 66, 2, 11]	340.05
8	[33, 3, 44, 4, 1, 11]	657.40
9	[33, 4, 1, 3, 2, 11]	658.31
10	[33, 4, 1, 11]	658.88

4.2 Performance Test

After carrying out basic routing testing, the network performance of the routing algorithm with load modifications will be tested. Testing was carried out with two types of packets, namely small packets with an IPerf throughput of 1 Mbps and an ICMP packet size of 64 bytes,

then big packets with an IPerf throughput of 1 Gbps and an ICMP packet size of 16 kilobytes. Determination of ICMP throughput and packet size is carried out based on the load test or stress test carried out at the beginning of the test. From this load test, the lower and upper limits of throughput and packet size are obtained, which are then called small packets and big packets.

These two types of traffic (small and large) describe the conditions of 5G network requirements in general. A throughput of 1gbps is needed for a 5G network to transmit high resolution video surveillance, while a throughput of 1 mbps is usually a small packet that requires super fast latency. With good results for performance testing on the traffic above, this Dijkstra routing load modification can be suitable for application on 5G networks.

4.3 Latency Performance Test

In the first test, the parameter tested was how the latency between H1-H3 was performed during five test repetitions. The latency measured is the average latency in one test, including when all links are up and when one to three are down. In the first performance test, the parameters tested were how the latency performed during five repetitions of the test. The latency measured is the average latency that occurs in one test, including when all links are up and when one to three links are down.

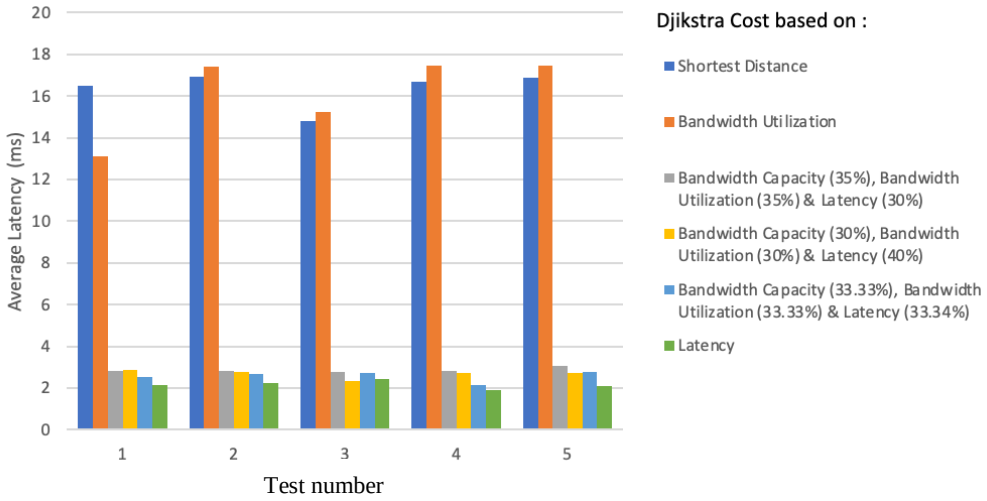

Fig. 9. Latency (Ms) Result Test Using Small Packet Size

Figures 9 and 10 show that the Dijkstra algorithm with the shortest distance load and bandwidth still delivers longer latency results compared to using other load calculations. This happens both when sending small packets and large packets. With these results it can be said that to achieve fast packet delivery with low latency, the Dijkstra algorithm cannot use load calculations with the shortest distance or bandwidth util. Load calculations must take into account latency conditions on the network when routing calculations are carried out, either using latency load calculations only or latency load calculations combined with bandwidth capacity and utilization loads.

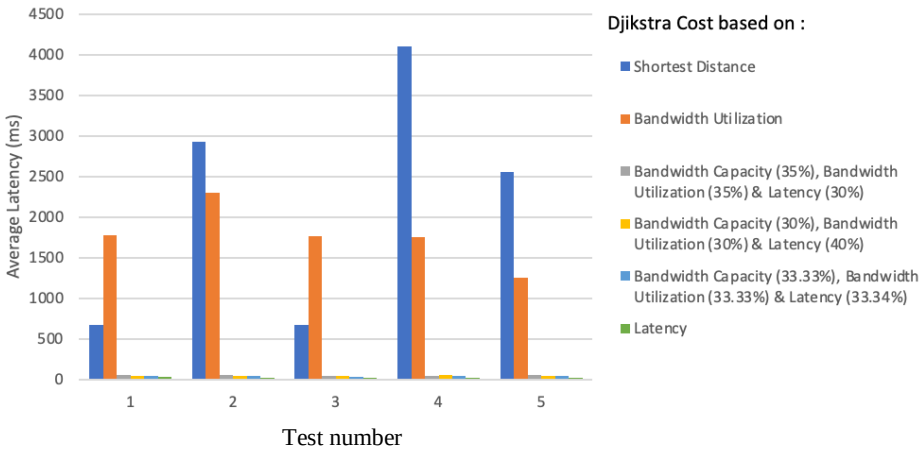

Fig. 10. Latency (ms) result test using big packet size

Two Figures 5 and 6 show the comparison amount of latency or time required between source and destination. A small packet requires an average of 3 milliseconds, and a large packet requires up to 4000 milliseconds. This happens in all experiments from the 1st to the 5th and also occurs in every scenario 1 to 3 link down. Figures 9 and 10 demonstrate that the general latency performance test results with load modifications to the Dijkstra algorithm using a combination of latency, bandwidth utilization, and reference bandwidth produce better performance than the Dijkstra algorithm with shortest distance load or bandwidth utilization alone. Even so, the results of the combination of latency load, bandwidth utilization, and reference bandwidth still cannot outperform the use of latency-only load calculation.

4.4 Hop Count Test

The second performance test tests the number of hops required for each Dijkstra algorithm load comparison. Figures 11 and 12 present that the routing load calculation based on the shortest distance and latency has the shortest number of hops. Meanwhile, it requires at minimum 4 hop counts, when the load routing path calculation is based on the load combination.

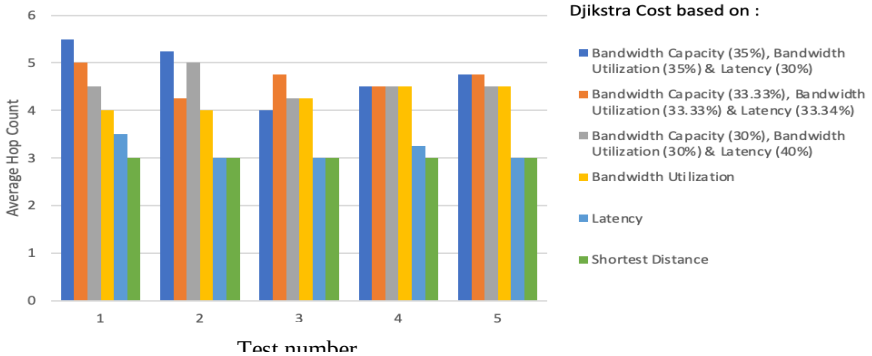

Fig. 11. Hop count result test using a small packet size

No number of hops exceeds 6 hops for both small packets and big packets. This proves that the entire routing path selection algorithm with various load modifications can successfully deliver packets smoothly, without having to circle the network which can increase delivery time. or latency also becomes long.

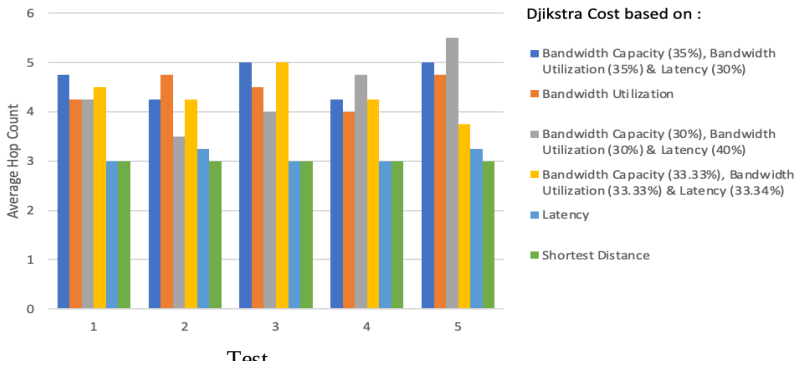

Fig. 12. Hop count result test using a big packet size

The general hop count performance test results in Figures 11 and 12 show that modification of the load on the Dijkstra algorithm using a combination of latency, bandwidth utilization, and bandwidth reference does not produce better performance than the Dijkstra algorithm with the shortest distance or latency load. This combination of latency load, bandwidth utilization, and reference bandwidth still cannot outperform bandwidth utilization load because the performance test results tend to be the same.

4.5. Packet Loss Performance Test

The final performance test is packet loss testing. In a real network, measuring packet loss is a very important parameter to be monitored and optimized regularly. Packet loss will greatly affect users' experience who expect the best network quality. In this research, the author uses a network emulator, where computational factors can also influence the packet loss performance results in this network simulation.

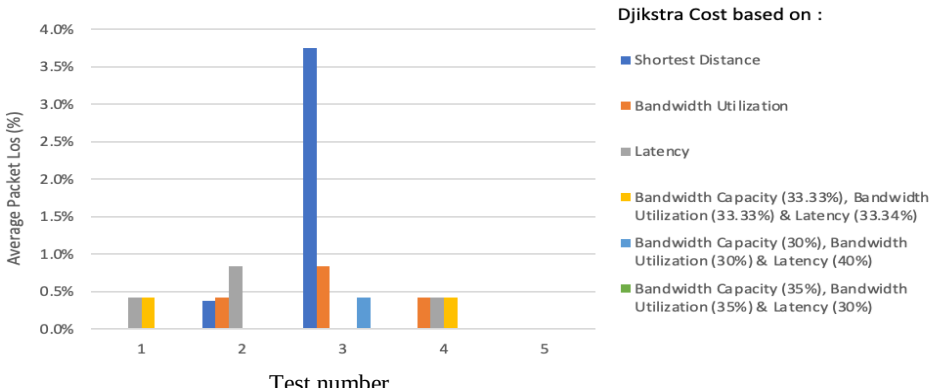

Fig. 13. Packet loss result test using a small packet size

Figures 13 and 14 show a fairly high increase in packet loss in the 3rd test, which reached 3.5% packet loss. This anomaly value is not caused by a bad network, but it is related to computational factors inside the emulator. There are some conditions when the ICMP test or ping test runs, the routing calculation has not been completed, so there are some packets that cannot get the best path information. These conditions cause the packet to drop and be lost.

In general, the packet loss performance results in Figure 6 do not reflect any significant differences from the modification of the Dijkstra algorithm load with a combination of latency, bandwidth utilization, and reference bandwidth. Apart from the computational factors in the network emulator, without any load modifications, the Dijkstra algorithm in this network emulator has shown good performance, namely 0% packet loss for various test scenarios.

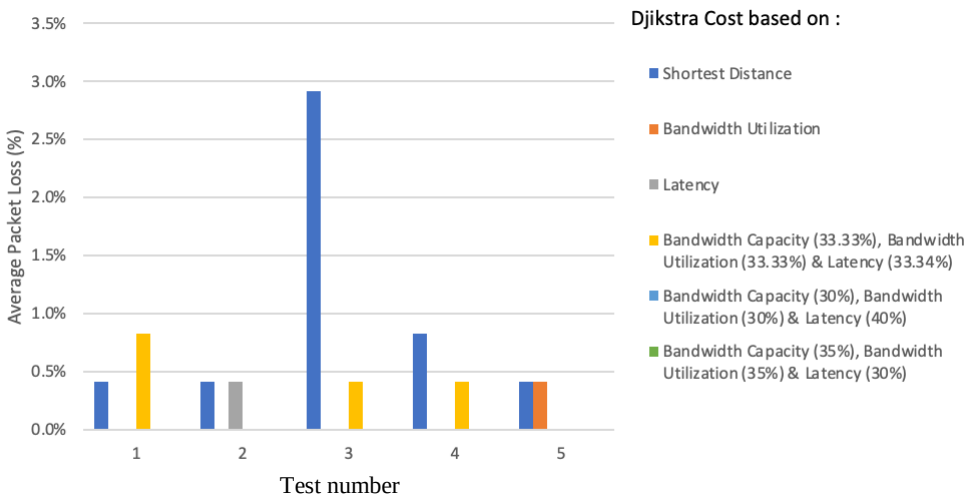

Fig. 14. Packet loss result test using a big packet size

However, this variation in load calculation parameters is what differentiates it from other research, because 5G network requirements are not only for ultra-high speed, but there are also ultra-low latency requirements, which can then only be met by using customize routing calculations that can be adjusted based on the use case.

4.6 Result Comparison

By the results above, there are some differences and similarities from the previous research. M Abdelghany *et al.,* (2022) has slightly difference latency achievement by getting 1 ms delay and 52 Mbps throughput, meanwhile our research reached 2 ms delay, but has better throughput delivery that reach 1 Gbps. Adedokun *et al.,* (2017) was not yet better, by only achieved 4.5 ms latency and 612 Mbps. Karami & Akhtarkavan (2015) has 0 ms delay when it only has under 30 events of routing exchange, after that it produced 90 ms delay. Jiang *et al.,* (2014) has 4-5 ms latency from the result. Fan & Shi (2010) achieved 90-300ms when defining new route from soruce to destination in road nework. Sander & Schultes (2007) improve dijkstra for fast route planning in map application, the query time need around 10-100ms delay.

5. Conclusion

This research can prove that load modification in the Dijkstra algorithm with a combination of latency load, bandwidth utilization and bandwidth capacity can work well. Increased network performance can be demonstrated by measuring packet loss of less than 1%, as well as latency performance of less than 5 milliseconds. This latency and packet loss performance gives the same performance results, there are even several tests that have better results compared to using the Dijkstra algorithm load without combination. With these results, the Dijkstra algorithm with load modification is expected to be able to answer packet routing needs on 5G networks, namely high throughput and low latency.

To get the optimal value, the load combination weights need to be tested with various scenarios. If in this study the experiment only used a weight figure of around 30% to 40%, then in future research it can be tested using a combination of numbers with a further polarization, for example the latency load weighs 70%, the bandwidth load weighs 30%, or vice versa. This is done so that the characteristics of the program can be more visible, so that it is hoped that the research results can be used as a comprehensive reference for its application in 5G network routing in the telecommunications industry.

References

Abdulaziz, A., Adedokun, E. A., & Man-Yahya, S. (2017). Improved extended Dijkstra's algorithm for software defined networks. *International Journal of Applied Information Systems (Online)/International Journal of Applied Information Systems*, *12*(8), 22–26. https://doi.org/10.5120/ijais2017451714

Albu-Salih, A. T. (2022). Performance evaluation of RYU Controller in software defined networks. *Magallaẗ Al-qādisiyyaẗ Li-ʿulūm Al-ḥāsibāt Wa-al-riyāḍiyyāt*, *14*(1). https://doi.org/10.29304/jqcm.2022.14.1.879

An, N., Kim, Y., Park, J., Kwon, D., & Lim, H. (2019). Slice management for quality of service differentiation in wireless network slicing. *Sensors*, *19*(12), 2745. https://doi.org/10.3390/s19122745

Araújo, S. O., Peres, R. S., Barata, J., Lidon, F., & Ramalho, J. C. (2021). Characterising the Agriculture 4.0 Landscape—Emerging Trends, challenges and opportunities. *Agronomy*, *11*(4), 667. https://doi.org/10.3390/agronomy11040667

Barakabitze, A. A., Ahmad, A., Mijumbi, R., & Hines, A. (2020). 5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges. *Computer Networks*, *167*, 106984. https://doi.org/10.1016/j.comnet.2019.106984

Bojović, P. D., Malbašić, T., Vujošević, D., Martić, G., & Bojović, Ž. (2022). Dynamic QoS Management for a Flexible 5G/6G Network Core: A Step toward a Higher Programmability. *Sensors*, *22*(8), 2849. https://doi.org/10.3390/s22082849

Chao, Y. (2010). A developed Dijkstra algorithm and simulation of urban path search. In *2010 International Conference on Crowd Science and Engineering*. https://doi.org/10.1109/iccse.2010.5593700

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms, third edition*. MIT Press.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, *1*(1), 269–271. https://doi.org/10.1007/bf01386390

Fan, D., & Shi, P. (2010). Improvement of Dijkstra's algorithm and its application in route planning. *2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery*. https://doi.org/10.1109/fskd.2010.5569452

Fuhao, Z., & Jiping, L. (2009). An algorithm of shortest path based on Dijkstra for huge data. In *2009 Sixth International Conference on Fuzzy Systems and Knowledge Discov*, *4*, 244–247. https://doi.org/10.1109/fskd.2009.848

Goransson, P., Black, C., & Culver, T. (2016). *Software defined networks: A Comprehensive Approach*. Morgan Kaufmann.

Goyal, M., Soperi, M., Baccelli, E., Choudhury, G., Shaikh, A., Hosseini, H., & Trivedi, K. (2012). Improving convergence speed and scalability in OSPF: a survey. *IEEE Communications Surveys and Tutorials/IEEE Communications Surveys and Tutorials*, *14*(2), 443–463. https://doi.org/10.1109/surv.2011.011411.00065

Hu, F., Hao, Q., & Bao, K. (2014). A survey on Software-Defined Network and OpenFlow: From Concept to implementation. *IEEE Communications Surveys and Tutorials/IEEE Communications Surveys and Tutorials*, *16*(4), 2181–2206. https://doi.org/10.1109/comst.2014.2326417

Huang, Y., Yi, Q., & Shi, M. (2013). An improved Dijkstra Shortest Path algorithm. *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering*. https://doi.org/10.2991/iccsee.2013.59

Jiang, J., Huang, H., Liao, J., & Chen, S. (2014). Extending Dijkstra's shortest path algorithm for software defined networking. In *2014 16th Asia-Pacific Network Operations and Management Symposium*. https://doi.org/10.1109/apnoms.2014.6996609

Kadry, S., Abdallah, A., & Joumaa, C. (2011). On the Optimization of Dijkstra's Algorithm. In *Lecture notes in electrical engineering* (pp. 393–397). https://doi.org/10.1007/978-3-642-25992-0_55

Karami, F., & Akhtarkavan, E. (2015). Improving OSPF Protocol based Latency : A new algorithm based on Dijkstra by using OSPF existing Metrics in SDN networks. *Ciência E Natura*, *37*, 344. https://doi.org/10.5902/2179460x20793

Khachiyan, L., Gurvich, V., & Zhao, J. (2006). Extending Dijkstra's algorithm to maximize the shortest path by Node-Wise limited arc interdiction. In *Lecture notes in computer science* (pp. 221–234). https://doi.org/10.1007/11753728_24

Komite Percepatan Penyediaan Infr,astruktur Prioritas (KPPIP). (2019). *Indonesia Digital for Future Economy and Inclusive Urban Transformation*. Deputy Ministry for Coordination of Infrastructure and Regional Development Acceleration.

Kreutz, D., Ramos, F. M. V., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., & Uhlig, S. (2015). Software-Defined Networking: A Comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14–76. https://doi.org/10.1109/jproc.2014.2371999

Lanning, D. R., Harrell, G. K., & Wang, J. (2014). Dijkstra's algorithm and Google maps. In *Proceedings of the 2014 ACM Southeast Regional Conference*. https://doi.org/10.1145/2638404.2638494

Luo, M., Hou, X., & Yang, J. (2020). Surface optimal path planning using an extended Dijkstra algorithm. *IEEE Access*, *8*, 147827–147838. https://doi.org/10.1109/access.2020.3015976

M Abdelghany, H., W Zaki, F., M Ashour, M. (2022). Modified Dijkstra Shortest Path Algorithm for SD Networks. *International Journal of Electrical and Computer Engineering Systems*, 13(3), 203-208. https://doi.org/10.32985/ijeces.13.3.5.

Mehlhorn, K., & Sanders, P. (2008). *Algorithms and data structures: The Basic Toolbox*. Springer Science & Business Media.

Noto, M., & Sato, H. (2000). A method for the shortest path search by extended Dijkstra algorithm. *In 2000 International Conference on Systems, Man and Cybernetics*, *3*, 2316–2320. https://doi.org/10.1109/icsmc.2000.886462

Palmieri, F. (2020). A Reliability and latency-aware routing framework for 5G transport infrastructures. *Computer Networks*, *179*, 107365. https://doi.org/10.1016/j.comnet.2020.107365

Sanders, P., & Schultes, D. (2007). Engineering fast route planning algorithms. In *Springer eBooks* (pp. 23–36). https://doi.org/10.1007/978-3-540-72845-0_2

Sirika, N. S., & Mahajan, N. S. (2016). Survey on Dynamic Routing Protocols. *International Journal of Engineering Research and Technology*, *V5*(01). https://doi.org/10.17577/ijertv5is010028

Shu-Xi, W. (2012). The improved Dijkstra's Shortest Path algorithm and its application. *Procedia Engineering*, *29*, 1186–1190. https://doi.org/10.1016/j.proeng.2012.01.110

Sutton, A. (2018). 5G Network Architecture: Enabling the future delivery and consumption of digital media. *The ITP (Institute of Telecommunications Professionals) Journal*, *12*, 9–15.

Szigeti, T., Hattingh, C., Barton, R., & Briley, K. (2013). *End-to-end QOS network design*. Pearson Education.

Tang, Y., Dananjayan, S., Hou, C., Guo, Q., Luo, S., & He, Y. (2021). A survey on the 5G network and its impact on agriculture: Challenges and opportunities. *Computers and Electronics in Agriculture*, *180*, 105895. https://doi.org/10.1016/j.compag.2020.105895

Wang, R. (2017). A research on the weighted improvement of Dijkstra algorithm in optimal path calculation. *In 2017 5th International Conference on Frontiers of Manufacturing Science and Measuring Technology*. https://doi.org/10.2991/fmsmt-17.2017.33

Wei, K., Gao, Y., Zhang, W., & Lin, S. (2019). A modified Dijkstra's algorithm for solving the problem of finding the maximum load path. *In 2019 Proceedings of the International Conference on Information and Communication Technology*. https://doi.org/10.1109/infoct.2019.8711024

Wenzheng, L., Junjun, L., & Shunli, Y. (2019). An improved Dijkstra's algorithm for shortest path planning on 2D grid maps. *In 2019 International Conference on Electronics Information and Emergency Communication*, 438–441. https://doi.org/10.1109/iceiec.2019.8784487

Wu, Q., Qin, G., & Li, H. (2015). An improved Dijkstra's algorithm application to multi-core processors. *Metal Journal*, *9*, 76–81. https://www.metaljournal.com.ua/assets/Journal/english-edition/MMI_2015_9/012_Qiong-Wu.pdf

Xiao, N. J., & Lu, N. F. (2010). An improvement of the shortest path algorithm based on Dijkstra algorithm. *In International Conference on Computer and Automation Engineering*, 383–385. https://doi.org/10.1109/iccae.2010.5451564

Xu, M., Liu, Y., Huang, Q., Zhang, Y., & Luan, G. (2007). An improved Dijkstra's shortest path algorithm for sparse network. *Applied Mathematics and Computation*, *185*(1), 247–254. https://doi.org/10.1016/j.amc.2006.06.094

Zhang, W., Jiang, C., & Ma, Y. (2012). An Improved Dijkstra Algorithm Based on Pairing Heap. *In 2012 Fifth International Symposium on Computational Intelligence and Design*, *2*, 419–422. https://doi.org/10.1109/iscid.2012.260