Dijkstra and Bidirectional Dijkstra on Determining Evacuation Routes

I G S Rahayuda\(^1\) and N P L Santiari\(^2\)

\(^{1, 2}\) Program Study of Information Systems, Faculty of Informatics and Computers, Institute of Technology and Business STIKOM Bali, Denpasar, Bali, Indonesia

Email: surya_rahayuda@stikom-bali.ac.id

Abstract. Determination of the best path or often called the shortest path finding is a method that has many benefits and can be applied in various cases and fields of science. In previous studies, authors have developed the shortest path finding program using various algorithms, such as held-karp, iterative deepening search, bidirectional depth first search and depth limited search. In this research author implements shortest path finding on information system for determining fire disaster evacuation route using Dijkstra and bidirectional Dijkstra. The program was developed based on web using vuetify.js framework. Both algorithms were tested on several datasets and the results will be compared to find out the strengths and weaknesses. Based on the research, the algorithm was successfully developed based on the web using the vuetify.js framework and successfully tested on several datasets, it is known that 40% of the test results show that bidirectional Dijkstra is better than Dijkstra, 10% vice versa and 50% get the same results from both algorithms. From this comparison it can be concluded that the bidirectional Dijkstra algorithm is better than Dijkstra. This research succeeded in proving that the bidirectional Dijkstra algorithm can produce a better route than the standard or classic Dijkstra algorithm.

1. Introduction

Determination of the best path or often called shortest path finding is a method that has many benefits and can be applied in various cases and fields of science. In previous studies authors have developed desktop program using visual basic programming language, where program has succeeded in determining the best route in case of products delivery. In the first year program was developed using two algorithms, namely Iterative Deepening Search and Held-Karp algorithm. From the results of research conducted it is known that Held-Karp algorithm gets better results compared to Iterative Deepening Search algorithm. From comparison with other algorithm, Bidirectional Depth First Search and Depth Limited Search get some better results than other algorithms. According to authors Bidirectional Depth First Search algorithm would be better if applied to cases that do not require a trip back in starting point and will get optimal results if used in a case of delivery that has data points close together at the beginning and ending levels of matrix graph, and has far points in middle level of matrix graph. Depth Limited Search method is best applied to several product shipments that have a lot of points or points of delivery and need to be limited to different shipping activities. From the results obtained by Depth Limited Search algorithm has similarities with Iterative Deepening Search algorithm, this is because both algorithms are variant from Depth First Search algorithm. Based on the
comparison of results obtained, if all methods are compared then Held-Karp algorithm gets the best results. In the third year research, author used Dijkstra algorithm. In contrast to previous studies, where the shortest path finding algorithm was applied to product delivery case, in third year of study, author tried to apply Dijkstra algorithm to a research case regarding determining the best route for fire fighter team when evacuating victims in a fire disaster [1]. In accordance with planning of previous research, this year author will develop both algorithms using web framework. Of the several existing web frameworks, author’s chose to use vue.js web framework, because vue.js framework, has advantages regarding appearance design, vue.js is frontend framework that is ranked at the top of several top rank’s web framework sites today. Vue.js has advantage of a reactive process, where the results of process are carried out directly, such as when switching pages, browser no longer reloads all page elements, so that it will get faster response, browser also does not need to re-download existing elements, so users can save more bandwidth. Two ways data binding facility can provide immediate results when a text is changed, this allows users to give messages to other users without pausing to reload or save. Vue.js also provides drag and drop facility that allows group of posts to be moved to other places in real time [2][3]. There are many advantages of vue.js, with some of these advantages, author will use vue.js framework in third year research. Author hopes that a good program implementation will be produced later, and can provide some new knowledge for writers, readers and sciences. In the current study or fourth year of research, authors add a new algorithm that namely bidirectional Dijkstra, bidirectional Dijkstra algorithm is variant of Dijkstra algorithm, different from Dijkstra algorithm which starts searching from beginning of graph matrix to the end of graph matrix level, while bidirectional Dijkstra algorithm starts search for Dijkstra algorithm. From beginning and end of graph matrix and ending at meeting between search that was carried out at beginning and end earlier or can be said to end in the middle of graph matrix level. According to several bidirectional Dijkstra sources, search process can be better than Dijkstra algorithm or often referred to as classic or standard Dijkstra. Dijkstra parallel or bidirectional Dijkstra algorithm, be based on two-way Dijkstra algorithm and popular multi core technology. Such technology makes it is possible to create parallel algorithms based on shared memory and most time for data synchronization. Author uses bidirectional Dijkstra's algorithm, because based on the references, bidirectional Dijkstra's is said to have a faster search execution process than Dijkstra's algorithm. Just likes research in previous year bidirectional Dijkstra's algorithm will be implemented based on web framework vue.js, more specifically on material design vuetify.js. The process and results of the two algorithms will be compared. Author hopes that in the future better route search results will be obtained, and results of the comparison will be known to the advantages and disadvantages of each algorithm [4]. The novelty of the current research is the use of a new algorithm, namely the bidirectional Dijkstra and the use of the vuetify.js framework. The purpose of this research is to find a search solution that is faster than the previous algorithm. So that it can overcome the problems that occurred in previous studies when using the Dijkstra algorithm.

2. Literature Review
This chapter will explain briefly about the literature study that used in this research, some of the literature is:

2.1. Dijkstra
Dijkstra Algorithm or Dijkstra First Shortest Path Algorithm (SPF) is an algorithm for finding the shortest path between nodes in a graph, which can represent, for example road network. Discovered by computer scientist Edsger W. Dijkstra in 1956 and published three years later [5]. Dijkstra algorithm exists in many variants, the original Dijkstra variant finding the shortest path between two nodes, but the more common variant fixes one node as the "source" node and finds the shortest path from the source to all other vertices in the graph, resulting in the shortest path tree. Dijkstra algorithm can be used to find the shortest route between one city and all other cities. The shortest path algorithm applications that are widely used are network routing protocols, especially Intermediate System to Intermediate System and Open Shortest Path First (OSPF) [6][1].
2.2. Bidirectional Dijkstra
Bidirectional Dijkstra is variant of Dijkstra algorithm, where this algorithm works by dividing two graphs matrix, then searching using Dijkstra algorithm separately. Then combine the results of two searches. The final results will be obtained in form of combined route or route results. Bidirectional Dijkstra can process searches better than Dijkstra algorithm or often called the classic or standard Dijkstra. Dijkstra parallel or bidirectional Dijkstra algorithm is based on two-way Dijkstra algorithm and popular multi core technology. Such technology makes it is possible to create parallel algorithms based on shared memory and most of the time for data synchronization [7]. Dijkstra parallel or bidirectional Dijkstra algorithm is almost 3 times faster than standard Dijkstra algorithm. Bidirectional Dijkstra or so-called advanced shortest paths can work in practice 1000s (up to 25000) times faster than the classic Dijkstra algorithm on real road networks and social network graphics [4][8].

2.3. Framework Vuetify.js
Vuetify.js is a semantic development framework for vue.js. Built with the material design, aims to provide all tools needed to create beautiful and rich content. Vuetify.js is a material design framework component that can be used for both mobile and desktop web applications and is perhaps the richest feature with server-side rendering, PWA and CLI template support [9][1].

3. Research Method
Author uses research method SDLC (System Development Life Cycle), SDLC is a software development cycle consisting of several stages. There are four methodologies in SDLC software development, namely: Waterfall, Prototype, RAD (Rapid Application Development) and Agile Software Development. In this study authors used the SDLC Waterfall methods. Waterfall or Classic Life Cycle is a method that is widely used in Software Engineering, this method takes a systematic and orderly approach from the level of system requirements to the analysis, design, implementation, and testing of the system. It is called a waterfall because the step by step, it must wait for the completion of the previous stage and proceed in sequence. The stages in this study are: requirement analysis, design system, implementation, comparison and testing [1][10].

4. Result and Discussion
This chapter will explain the result of the research that has been carried out, the author explains them in accordance with the order of the research methods used, requirement analysis (dataset), design system (data flow diagram), implementation, comparison and testing (black box). The following is the discussion:

4.1. Dataset
Here are some data that will use in research. The data is an open data set shared by the United States government, more specifically in city of Naperville. Based on the source, type of data that used is secondary data because data collected is obtained from open dataset. The data are official data shared by Naperville Fire Department, United States of America. Until now there were 149,864 fire disasters that occurred in the city from 2010 to 2019. Author uses as many as 80 most recent datasets of the whole data, here are some datasets that will use in research as shown in table 1 [1][6].

| No | Address            | Coordinate           |
|----|--------------------|----------------------|
| 1  | Normandy           | (41.711936, -88.222768) |
| 2  | Audubon            | (41.811179, -88.200837) |
| 3  | Rosecroft          | (41.711787, -88.210966) |
| 4  | Naperville Wheaton | (41.799269, -88.122684) |
| 5  | Bradford           | (41.785456, -88.206206) |
| 6  | Royal St George    | (41.781942, -88.168579) |
| 7  | Water              | (41.770828, -88.150572) |
4.2. Data Flow Diagram
To explain the flow and concepts of research, writing uses Data Flow Diagrams (DFD) to model the flow of research conducted. DFD is depicted starting from the context diagram up to level 3. DFD of Information Systems can be seen in figure 1 to 7 [1].

Figure 1. Context Diagram
Figure 2. Diagram Level 0
Figure 3. Diagram Level 1
Figure 4. Diagram Level 2 Process 1
Figure 5. Diagram Level 2 Process 2
Figure 6. Diagram Level 3 Process 1
Figure 7. Diagram Level 3 Process 2

4.3. Implementation
Dijkstra and bidi-Dijkstra algorithm have been successfully developed based on the vuetify.js web framework. Bidi-Dijkstra page can be seen in figure 8 and 9. In addition to the Dijkstra and Bidi-Dijkstra algorithm, several other pages were also added to complement the fire disaster evacuation information system. Some of these pages are: home, fire incident, fire stations, contact us and questions [1][6].
**Figure 8.** Bidirectional Dijkstra Submit Page

**Figure 9.** Bidirectional Dijkstra Result Page
The Dijkstra and Bidi-Dijkstra algorithm are placed in the shortest path finding menu. Users or admins can choose which method to use. On this page the user or admin can choose the several random locations of fire, and then press the shortest path button. The system will direct all data entered into the matrix pages and bidi-Dijkstra or Dijkstra pages where the data will be processed using matrix graph and Bidi-Dijkstra or Dijkstra method. You will see the resulting of matrix graph, the closest distance to each matrix level and the closest combined path line formed. Author will focus the explanation on the bidi-Dijkstra algorithm, regarding the explanation of the Dijkstra algorithm can be seen at the previous author's article. In this article author will use the same dataset for Dijkstra and bidi-Dijkstra, so that the results can be compared. Bidi-Dijkstra algorithm starts searching from first levelled of graph matrix, namely FD001, in addition to searching from beginning, Bidi-Dijkstra also searches from the end of graph matrix level, starting from FI046, as shown in first level graph matrix (see table 2).

Table 2. First level graph matrix

| First: 1 | FD001 | FI047 | FI048 | FI043 | FI004 | FI046 |
|----------|-------|-------|-------|-------|-------|-------|
| FD001    | 0.0   | 6.8   | 6.7   | 6.3   | 5.5   | 4.7   |

| End: 1   | FD001 | FI047 | FI048 | FI043 | FI004 | FI046 |
|----------|-------|-------|-------|-------|-------|-------|
| FI046    | 4.7   | 2.7   | 2.1   | 1.9   | 3.2   | 0.0   |

The minimum distance search for first 1 is done by ignoring the 3 block graph matrix values on the right, and vice versa with end 1, this is done to avoid selecting the same block from the first and end searches. From the entire data contained in the first graph, it was found that the nearest route was FD001 to FI048 with a distance of 6.7 miles, and on the end graph 1 the nearest route was FI046 to FI043 of 1.9 miles. Then proceed with a search on the FI048 matrix graph for first 2 and FI043 for end 2 (see table 3).

Table 3. Second level graph matrix

| First: 2 | FD001 | FI047 | FI048 | FI043 | FI004 | FI046 |
|----------|-------|-------|-------|-------|-------|-------|
| FI048    | 6.7   | 1.9   | 0.0   | 1.7   | 4.1   | 2.1   |

| End: 2   | FD001 | FI047 | FI048 | FI043 | FI004 | FI046 |
|----------|-------|-------|-------|-------|-------|-------|
| FI043    | 6.7   | 1.9   | 0.0   | 1.7   | 4.1   | 2.1   |

At the first graph level 2, found FI048 to FI047 has the closest route of 1.9 miles, and at end 2 found the route FI043 to FI004 of 4.1 miles, the search continues on the matrix graph FI048 for first 3 and FI004 for end 2 (see table 4).

Table 4. Third level graph matrix

| First: 3 | FD001 | FI047 | FI048 | FI043 | FI004 | FI046 |
|----------|-------|-------|-------|-------|-------|-------|
| FI047    | 6.8   | 0.0   | 1.9   | 1.4   | 2.4   | 2.7   |

| End: 3   | FD001 | FI047 | FI048 | FI043 | FI004 | FI046 |
|----------|-------|-------|-------|-------|-------|-------|
| FD004    | 5.5   | 2.4   | 4.1   | 2.3   | 0.0   | 3.2   |

The search has reached the end of the matrix graph level limit, the closest route chosen in first 3 is the route to the last matrix graph level, FI046 of 2.7 miles and at end 3 the route chosen to the matrix graph's starting point is FD001 of 5.5 miles. All routes are then combined into series of travel routes: FD001 → FI048 → FI047 → FI046 → FI043 → FI004 → FD001, with total distance of 21 miles and travel time of 42 minutes.
4.4. Comparison

All data that has been collected and stored is then tested on web-based program [11]. From the trials conducted evacuation routes can be generated in each case of fire disaster, as shown in table 5.

| Best Route Bidi-Dijkstra | Output | Distance |
|--------------------------|--------|----------|
| **Case 1, 22 april 2019** |
| Naperville F.D. | Naperville F.D. → Water | 6.7 mil |
| Royal St George | Water → Royal St George | 1.9 mil |
| Water | Royal St George → Oswego | 2.7 mil |
| Hidden Spring | Oswego → Hidden Spring | 1.9 mil |
| Bradford | Hidden Spring → Bradford St 1 | 2.3 mil |
| Oswego | Bradford St 1 → Naperville F.D. | 5.5 mil |
| **Result** | FD001→FI048→FI047→FI046→FI043→FI004→FD001 | **21 mil** |
| **Case 2, 22 april 2019** |
| Naperville F.D. | Naperville F.D. → Rosecroft | 1 mil |
| Rosecroft | Rosecroft → Normandy | 1.1 mil |
| Normandy | Normandy → Bradford | 6.5 mil |
| Audubon | Bradford St 1 → Audubon | 2.3 mil |
| Naperville Wheaton | Audubon → Naperville Wheaton | 6.1 mil |
| Bradford St 1 | Naperville Wheaton → Naperville F.D. | 9.6 mil |
| **Result** | FD001→FI031→FI025→FI004→FI001→FI024→FD001 | **26.6 mil** |
| **Case 3, 21 april 2019** |
| Naperville F.D. | Naperville F.D. → Fulham | 1.6 mil |
| Fulham | Fulham → Main | 6.3 mil |
| Main | Main → Iroquois | 2.5 mil |
| Diehl St 1 | Iroquois → Ogden | 1.4 mil |
| Ogden | Ogden → Diehl St 1 | 3.2 mil |
| Iroquois | Diehl St 1 → Naperville F.D. | 6.8 mil |
| **Result** | FD001→FI042→FI045→FI044→FI027→FI013→FD001 | **21.8 mil** |
| **Case 4, 21 april 2019** |
| Naperville F.D. | Naperville F. D. → Grommon | 2.5 mil |
| Grommon | Grommon → Flat Rock | 1.6 mil |
| Flat Rock | Flat Rock → Cantera | 10.4 mil |
| Naper | Cantera → Naper | 3.7 mil |
| Ogden | Naper → Ogden | 4.2 mil |
| Cantera | Ogden → Naperville F. D. | 8.6 mil |
| **Result** | FD001→FI019→FI016→FI008→FI023→FI027→FD001 | **31 mil** |
| **Case 5, 20 april 2019** |
| Naperville F.D. | Naperville F. D. → Willow Ridge | 1.9 mil |
| Willow Ridge | Willow Ridge → Buttonwood | 4 mil |
| Buttonwood | Buttonwood → Candlenut | 1.4 mil |
| Best Route Bidi-Dijkstra | Distance |
|--------------------------|----------|
| Diehl St 1               | Candlenut → Westminster 2.4 mil |
| Westminster              | Westminster → Diehl 1.2 mil |
| Candlenut                | Diehl → Naperville F. D. 6.8 mil |
| **Result**               | FD001→FI038→FI006→FI007→FI036→FI013→FD001 17.7 mil |

6 **Input Output**: Case 6, 19 april 2019

| Naperville F. D.         | Naperville F. D. → Stockton 1.7 mil |
| Stockton                 | Stockton → White Eagle 1.1 mil |
| White Eagle              | White Eagle → Country Lakes 6.2 mil |
| Redstart                 | Country Lakes → Centre Point 4.5 mil |
| Center Point             | Centre Point → Redstart 5.8 mil |
| Country Lakes            | Redstart → Naperville F. D. 5.5 mil |
| **Result**               | FD001→FI034→FI037→FI011→FI010→FI030→FD001 24.8 mil |

7 **Input Output**: Case 7, 18 april 2019

| Naperville F.D.          | Naperville F. D. → Aurora 4.4 mil |
| Aurora                   | Aurora → Ellsworth 3.3 mil |
| Ellsworth                | Ellsworth → Hobson Mill 2.7 mil |
| Russet                   | Hobson Mill → Russet 2.2 mil |
| Woodview                 | Russet → Woodview 2.2 mil |
| Hobson Mill              | Woodview → Naperville F. D. 5 mil |
| **Result**               | FD001→FI002→FI015→FI022→FI032→FI039→FD001 19.8 mil |

8 **Input Output**: Case 8, 17 april 2019

| Naperville F.D.          | Naperville F. D. → Patriots 5.4 mil |
| Patriots                 | Patriots → Genesee 1.4 mil |
| Genesee                  | Genesee → North 5.0 mil |
| Washington               | North → Dorval 4.1 mil |
| Dorval                   | Dorval → Washington 1.5 mil |
| North                    | Washington → Naperville F. D. 4.7 mil |
| **Result**               | FD001→FI028→FI018→FI026→FI014→FI035→FD001 22.1 mil |

9 **Input Output**: Case 9, 16 april 2019

| Naperville F.D.          | Naperville F. D. → Gypsum 1 mil |
| Garnette                 | Gypsum → Garnette 3.7 mil |
| Gypsum                   | Garnette → Raintree 4.5 mil |
| Brookdale                | Raintree → Brookdale 3.8 mil |
| Naperville Wheaton       | Brookdale → Naperville Wheaton 5.5 mil |
| Raintree                 | Naperville Wheaton → Naperville F. D. 9.6 mil |
| **Result**               | FD001→FI020→FI017→FI029→FI005→FI024→FD001 28.1 mil |

10 **Input Output**: Case 10, 15 april 2019

| Naperville F.D.          | Naperville F. D. → Spartina 3.3 mil |
| Harbor                   | Spartina → Harbor 7.8 mil |
| Spartina                 | Harbor → Center 3.4 mil |
In previous studies author have tested Dijkstra algorithm on the same dataset, table 6 will display comparison of the results of trials conducted on Dijkstra algorithm and bidi-Dijkstra.

| Dataset          | Dijkstra | Bidi-Dijkstra | Difference |
|------------------|----------|---------------|------------|
| Case 1           | 19.5 mil | 21 mil        | -1.5 mil   |
| Case 2           | 26.6 mil | 26.6 mil      | 0 mil      |
| Case 3           | 23.1 mil | **21.8 mil**  | **1.3 mil**|
| Case 4           | 31.3 mil | **31 mil**    | **0.3 mil**|
| Case 5           | 17.7 mil | 17.7 mil      | 0 mil      |
| Case 6           | 25.6 mil | **24.8 mil**  | **1.2 mil**|
| Case 7           | 19.8 mil | 19.8 mil      | 0 mil      |
| Case 8           | 22.4 mil | **22.1 mil**  | **0.3 mil**|
| Case 9           | 28.1 mil | 28.1 mil      | 0 mil      |
| Case 10          | 25.6 mil | 25.6 mil      | 0 mil      |

From the table, it is known that in some cases the Dijkstra algorithm tested had the same results as bidi-Dijkstra. In case 1, Dijkstra algorithm gets better results, whereas in cases 3, 4, 6 and 8 the bidi-Dijkstra algorithm gets better results, so it can be concluded that bidi-Dijkstra algorithm gets better results than Dijkstra algorithm.

4.5. **Blackbox Testing**

In this research, black box testing method is used. Testing is done by testing the interface part of the information system, each part of the interface is tested to determine whether the system is running in accordance with the expected function. Home Page: Admin or user can access home page, on the home page will display some news content that is displayed in form of timeline, each column of news can be pressed so that later can be accessed pages that display news details. Fire Incident Page: The fire incident page can be accessed by the admin and user, the page will display list of all fires disaster locations in the form of table, there is column in the form of a percentage of damage in the form of a bar line, where if the cursor is directed at the bar line will display details of the damage in the form of tooltips. Fire Stations Page: The fires stations page can be accessed by users and administrator [12], page will display all existing fire stations in form of card column accompanied by an image. Dijkstra Page: Dijkstra page is one of main pages of the system, where admin can access and can input scene of fire disaster, if pressed submit button system will process graph matrix and Dijkstra method to get best route solution. Bidi-Dijkstra Page: Bidi-Dijkstra page is one of main pages of the system, where admin can access and can input scene of fire disaster, if you press submit button system will process graph matrix and Bidi-Dijkstra method to get best route solution. Contact Us Page: Contact us page can be accessed by user and administrator, user can input identity and can write suggestions or questions then, data will be stored in system database. Questions Page: The questions page can be accessed by user and admin, on this page real-time logging technology is used from vueify.js, where if the view questions button be pressed, all questions in the database will be displayed in the form of animated slide vertical transitions in rotation and change colour. Based on the screening scenario on several test classes, namely: home, fire incident, fire stations, Dijkstra, bidi-Dijkstra, contact us and questions. Obtained results in accordance with expectations [1][12].
5. Conclusions
The information system for determining fire disaster evacuation route has been successfully developed using vuetify.js framework. The flow and design of the system was successfully illustrated using data flow diagrams, where data flow diagrams were developed to level 3. Dijkstra and bidi-Dijkstra were successfully applied to the system and successfully tested on several datasets. From 10 fire incidents, there were 5 evacuation cases which had the same results between the Dijkstra and bidi-Dijkstra methods and there were 4 evacuation cases, where bidi-Dijkstra method had better results compared to Dijkstra method. Limited number of datasets can also affect results. System testing is done using black box testing method, based on the results of test obtained good results and in accordance with the wishes. This research succeeded in proving that the bidirectional Dijkstra algorithm can produce a better route than the standard or classic Dijkstra algorithm. We hope that further research will be carried out on bidirectional Dijkstra's algorithm, development and implementation in various fields.

6. References
[1] I. G. S. Rahayuda and N. P. L. Santiari, “Penentuan Rute Evakuasi Bencana Kebakaran Menggunakan Algoritma Dijkstra berbasis Web Framework Vue.js,” J. Sist. dan Inform., vol. 14, no. 2, pp. 80–88, 2020, doi: https://doi.org/10.30864/jsi.v14i2.252.
[2] A. Tarasyev, V. Turygina, M. Pavlov, Y. Kharitonov, and D. Solty, “Development of client-server technology for access to the database of algorithms on the VUE.js platform in,” in CEUR Workshop Proceedings, 2019.
[3] Y. Deng, Y. Chen, Y. Zhang, and S. Mahadevan, “Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment,” Appl. Soft Comput. J., 2012, doi: 10.1016/j.asoc.2011.11.011.
[4] G. Vaira and O. Kurasova, “Parallel Bidirectional Dijkstra’s Shortest Path Algorithm,” Front. Artif. Intell. Appl., vol. 224, pp. 422–435, 2010.
[5] T. J. Misa, “Interview: An interview with Edsger W. Dijkstra,” Communications of the ACM. 2010, doi: 10.1145/1787234.1787249.
[6] I. G. S. Rahayuda and N. P. L. Santiari, “Fire Incident Emergency Response Plan using Hybrid Fuzzy Dijkstra,” 2019 1st Int. Conf. Cybern. Intel. Syst., 2019, doi: 10.1109/ICORIS.2019.8874927.
[7] C. T. Gelareh B. Sanjabi, Duc T. Nguyen, “Heuristic Bidirectional Dijkstra Algorithm Using Piece-Wise Linear Function,” in International Conference on Transportation and Development, 2018.
[8] M. Levin, “Advanced Shortest Paths Project,” in Algorithms on Graphs, 2020.
[9] I. G. S. Rahayuda and N. P. L. Santiari, “Crawling and Cluster Hidden Web using Crawler Framework and Fuzzy-KNN,” in 2017 5th International Conference on Cyber and IT Service Management, CITSIM 2017, 2017, doi: 10.1109/CITSM.2017.8089225.
[10] M. Bulman, “SDLC - Waterfall Model,” Indep., 2017.
[11] H. Vamja, A. Shaikh, and S. Rami, “Comparative Analysis of Different Path Finding Algorithms to Study Limitations and Progress,” in The Fourteenth International Conference on Networks, 2017, vol. 7, no. 9, pp. 68–75.
[12] T. S. Jaya, “Pengujuan Aplikasi dengan Metode Blackbox Testing Boundary Value Analysis (Studi Kasus: Kantor Digital Politeknik Negeri Lampung),” J. Inform. Pengemb. IT, 2018, doi: 10.30591/jpit.v3i1.647.

Acknowledgments
This study is funded by the Institute of Technology and Business STIKOM Bali, Indonesia under the individual research scheme.