Mobile and Forensics
Vol. 5, No. 1, March 2023, pp. 12-22
ISSN: 2714-6685 (online), 2656-6257 (print)
DOI: http://dx.doi.org/10.12928/mf.v5i1.7906

Improved Breadth First Search for Public Transit Line Search Optimization

Suprihatin[a,1*], Imam Riadi[a,2], Furizal[b,3] Ahmad Azhari[c,4]
[a]Department of Information System, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[b]Master Program of Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[c]Department of Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia
[1,*]suprihatin@uad.ac.id, [2]imam.riadi@is.uad.ac.id, [3]2108048014@webmail.uad.ac.id,
[4]ahmad.azhari@tif.uad.ac.id
*Corresponding email

Abstract

People in general find it difficult to determine the transportation route, because to get to one destination there are many alternative paths that must be passed. This study aims to model the search for alternative bus route routes that are faster to produce routes that must be passed. The method used in this study is Improved Breadth first search by modifying BFS so that its performance is improved in producing route search completion. The improved BFS method is basically the same as BFS doing a level-by-level search stop if a false finish point is found. As the experiment above with a starting point of 175 and an end point of 54 the BFS algorithm takes 27 seconds 564 milliseconds, while the Improve BFS algorithm takes 171 milliseconds. The results showed that improved BFS can improve the performance of the BFS method. Research can be a model to be applied to other optimal route-finding cases.

Keywords: BFS, Route search, Optimization, Improved BFS, Path

INTRODUCTION

Transportation is a service that is useful for moving or carrying people or goods from one place to another [1][2]. Transportation is a facility that is often used by the community to support all their activities that cannot be separated from their daily lives [3]. Public transportation requires information on the path of travel and is part of the needs for the people of the city [4]. These lines intersect each other at a stop or terminal to form a transportation network [5][6]. A passenger traveling from one place to another can ride one vehicle lane by changing to another vehicle lane [7]. It is not an easy problem for passengers to be able to change vehicle lanes so that they will arrive at the intended place [8][9]. The route formed by the vehicle path is a collection of stops or terminals that are intertwined, can form a graph Euler circuit, that is, from one stop / terminal will return to that stop / terminal again [10].
The search for public transport lines is one of the complex problems in the field of transport [11][12]. The need for optimization of the search for public transportation routes is increasingly important given the growth in population and mobility in big cities [13][14]. In order to speed up the search for public transit lines, many methods have been developed, including the Breadth First Search (BFS) method [15][16]. Although BFS has proven effective in the search for public transit lines, there are still some disadvantages that need to be addressed, such as high time complexity and inefficient search on very large graphs.

Therefore, in this study, an advanced method of BFS called Improved Breadth First Search (IBFS) was introduced to optimize the search for public transportation lines. IBFS combines BFS principles with new techniques to improve search speed [17]. The results of this research can contribute to the development of more efficient and effective transportation systems in large cities.

To support this research, there have been several related previous studies. One of them is research on optimizing the route of waste transport trucks using the BFS algorithm [18]. Based on the existing test results, comparison results were obtained with BFS as the optimal algorithm in the process of mileage and processing time, while DFS is the optimal algorithm in minimizing the total volume of trips. In addition, BFS has also been applied to determining the shortest route for disaster evacuation shelters in the Purus area of Padang City [19]. The results of this study show that the BFS algorithm can be used in assisting communities in finding the most optimal path to reach the nearest and safest shelter when a tsunami disaster occurs, thereby reducing the risk of more victims.

In other studies, BFS has also been applied in the geographic information system of bus and travel counters in Padang City [20]. The results of his research showed that the application of the shortest path search with the BFS method can display the optimal shortest path, so that the people of Padang City can find out the location points of bus and shuttle counters in Padang City and can see bus and shuttle departure schedules.

METHODS

The transport network is basically a directional and weighted graph with the points being stops/terminals [21][22], a line is two consecutive points on its path, weights are its path. There are two blind search methods to obtain path search completion, namely Depth First Search (DFS) [23][24] and Breadth First Search (BFS) [25]. The DFS process will trace the node from the lowest level to the next level to get the sought-after node, otherwise get backtracking to the previous level [26]. The BFS search method by visiting nodes at the lowest level until the searched node is found, if not found, it will proceed to the next level [27], [28]. DFS produces more nodes than BFS, for this reason BFS was chosen for this study. BFS takes a long time because you have to go through the levels. The problem will increase if the BFS method is applied to transportation routes consisting of many stops and lanes [29]. This method will be modified by changing the finish point to a pseudo-finish point [30]. A pseudo-finish point is a point that can deliver to the finish point with a certain path.

A graph is a collection of points and lines connecting [31]–[33]. The following algorithm 1 is the BFS pseudocode to find the path from Start point to Finish point.

Algorithm 1. BFS Pseudocode

```
var
    path : list of Node;
    found : boolean;
Begin
  queue ← null;
  GetInToQueue(Start);
  queue position ← head of queue ;
  found ← false;
  while (not found) do
  Begin
    path ← element of queue position
    found ← (last element of path = Finish);
     if not found then
```

```
14          Begin
15          node ← the last node of the path
16          for all points: k
17            Begin
18              exist ← move(node,k);
19              if (exist and not in(k,path)) then
20                  Begin
21                    newpath← path + k;
22                    GetInToQueue(newpath);
23                  end;
24            end;
25        End;
26          queue position ← queue position next
27        end;
28      end;
```

The algorithm starts with an empty queue on line 5, line 6 enters the start point into the queue. The position of the queue at the beginning of the queue is as shown on line 7. Line 8 initialization meets the price false. Line 9 loops i.e. while not yet met do line 10 to line 27. Line 11 shows the path searched filled with queue position elements. If the path is the last element is the finish point, then the path sought has been found shown on line 12. If you do not find it, the path is deepened, which is shown on lines 14 to 25. Line 15 is taking the node from the last element of the path. Line 16 applies to all k points: if there is a successor k and k is not in the path, then line 21 creates a new path by adding the k element to the path, and line 22 inserts the new path into the queue. Figure 1 is an example showing the BFS method search process.

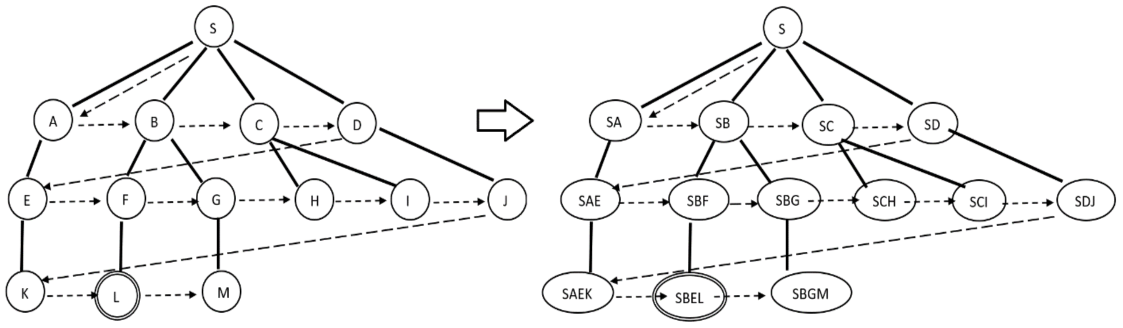

Figure 1. BFS Tree and Path Tree

Figure 1 starts with an empty queue line 5 in the BFS pseudocode, line 6 contains the string S, the position of the queue is on the element S. Point S is not the finish, points A, B, C, and D successor to point S, then the String S is developed into SA, SB, SC, and SD. Add the development result to the queue so that it becomes S, SA, SB, SC, SD. Henceforth, each completed development will enter the queue. The queue position advances to the next element SA. Point A on SA is not the finish, point E is the successor to point A. Therefore, the SA element is developed into SAE and enters the queue. The queue position advances to element SB, Point B instead of finish, points F and G successors point B. SB elements are developed into SBF and SBG. The queue position advances to the SC element. Point C is not the finish, points H and I successor points C. SC elements are developed into SCH, SCI. The queue position advances to the SD element, Point D instead of finish, point J successor point

D. The SD element is developed into SDJ. The queue position advances to the SAEK element, Point K is not the finish, Point K has no successor. The queue position advances to the SBEL element, the L finish point of the process is completed, so the result is SBEL.

Proposed Method

The transport path is a weighted graph, for example the search tree as Figure 2.

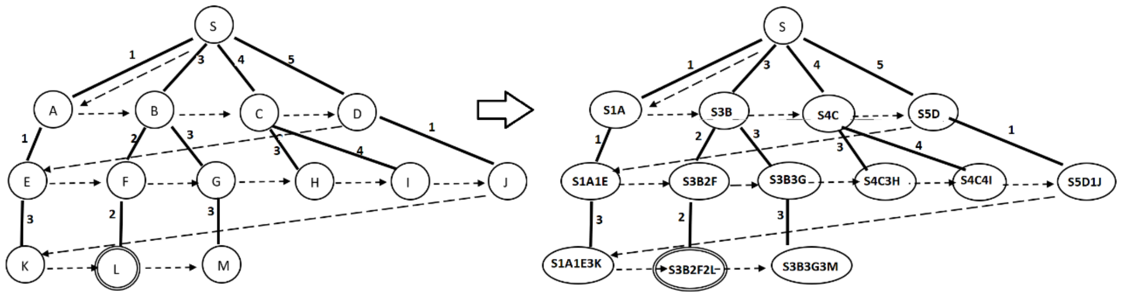

Figure 2. Weighted BFS graph search and Tree Path

The BSF search process is added with its weight so that the search result is S1A1E3K. The BSF process takes a long time, this method will be improved by cutting the nodes on the search graph if a pseudo-finish node is found (i.e. the node that delivers to the finish node with a certain path) [20]. The picture below shows node B is the pseudo-finish point because there is path 2 that delivers to the finish point (L). Figure 3 represents the BFS search tree adding pseudo-finish point B.

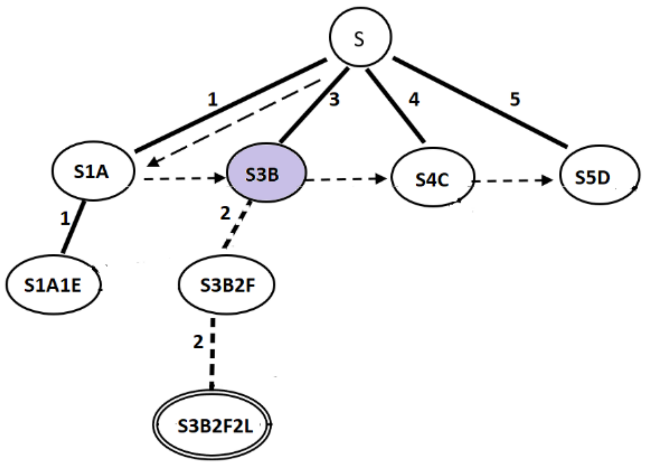

Figure 3. BFS search freight line updates

The search process will be improved again by stopping path development if a false finish point is found, as shown in Figure 4. The final fix in figure 4 will be used to resolve this pathfinding issue.

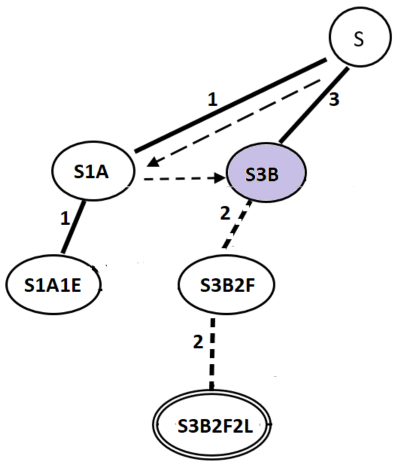

Figure 4. BFS search for updates to both freight lines

RESULT AND DISCUSSIONS

The final fix in figure 4 will be used to resolve this pathfinding issue.

Implementation

To facilitate the implementation, algorithms and accompanying functions will be formed.

Algorithm

The following is a pseudocode improve BFS as an implementation of figure 4 aimed at speeding up the BFS process:

Algorithm 2. Improve BFS Pseudocode

```
var
   path : list of Node;
   found : boolean;
   stop  : boolean;
Begin
  queue ← null;
  stop ← false;
  GetInToQueue(Start);
  queue position ▯ head of queue ;
  found ← false;
  while (not found) and (not stop ) do
  Begin
    path ← element of queue position
    found ← (speudofinish(last element of path));
     if not found then
     Begin
     node ← the last node of the path
     for all points: k and not stop
       Begin
         exist ← move(node,j,k);
         if (exist and not in(k,path)) then
           Begin
             newpath ← path + j + k;
             GetInToQueue(newpath);
```

```
25                stop ← speudofinish(k);
26               end;
27           end;
28         End;
29             If(stop) then
30               Begin
31                Jump to last queue position ;
32           found ← true;
33              end else  queue position ← queue position next
34        end;
35      end;
```

Algorithm 2 is an update of BFS by adding a stop variable to stop the search process. The stop variable is initialized with false on line 7. The while loop is updated with no encounters and no stops on line 11. The meet variable is populated with the pseudo-finish point on line 14. Line 18 has additional states and does not stop. Line 25 pricing the stop of the apparent finish point. Lines 29-33 are if you meet a pseudo-finish point, do jump to the end of the queue and have found the path at the last path of the queue, otherwise the queue advances to the next queue.
The object of the study will be carried out with bus lines that require many nodes and many lines. As a node, a byte type (char) will be selected, and a path of string type to facilitate implementation in the algorithm or programming. A transportation route such as Figure 5 will be taken.

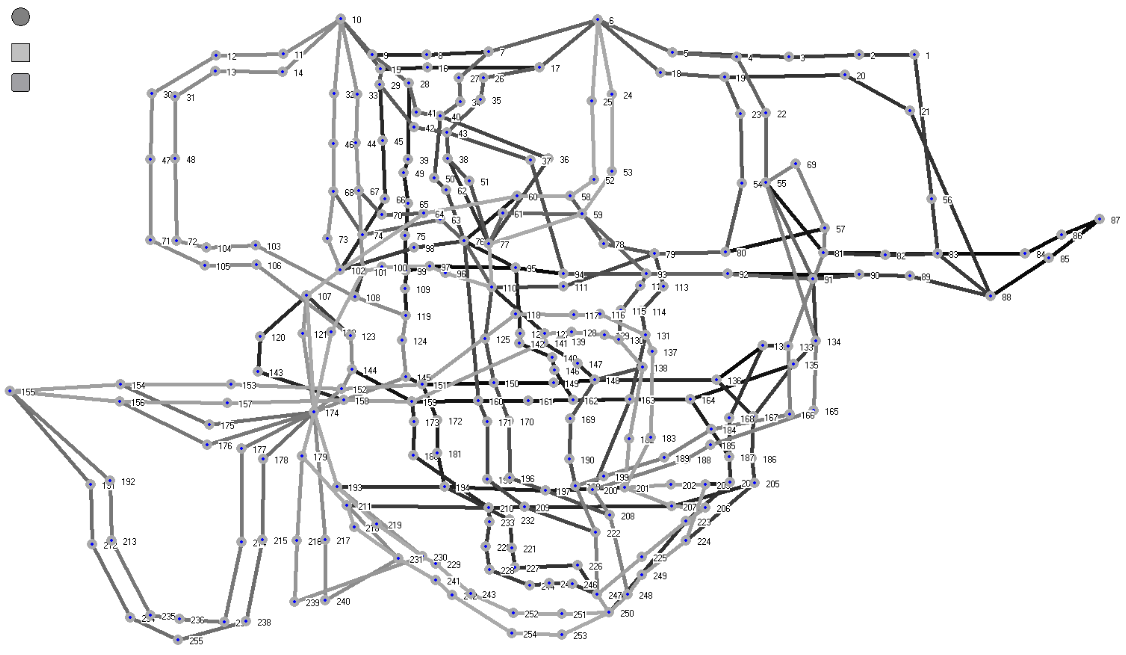

Figure 5. Graph of freight line implementations

Move Function

The path is represented by a string, that is, if the string s is known, it can be formed a move function (a, i, b), where i is the path number a and b is the character in the consecutive s. Element a comes before element b. If described as follows:

Line i: -------------------------ab------------------------

If point a is known and any point is sought after it, not all paths contain the function of moving from point a to another point, only part of it, for example, the following paths are known:

Line 1: ------------------------------ab------------------
Line 2: ----------------------------------ac-----------------
Line 3: ----------------------------b-----------------
Line 4: ----------------------------ax----------

Then the moving function can be taken as follows: move(a,1,b), move(a,2,c), move(a,4,x).

Pseudo Endpoint Functions
If known the paths that make the cycle as follows:
Line 1: abcdefghijka
Line 2: defgbacjklimnopd
For example, the search to point c and the finish is k, then c is called the pseudo-finish point because with path 1 from point c can be delivered to point k with points passed: cdefghijk, with path 2 points are cjk. There is more than one then it will be taken whose minimum length is: cjk.
For example, the search to point k and the finish is c then to make it easier can be combined the path as follows:
Line 1: abcdefghijkabcdefghijka
Line 2: defgbacjklimnopdefgbacjklimnopd
So that the points passed: with line 1: kabc, with line 2: klimnopdefgbac, there is more than one choice then a minimum is taken. This method will be used to find the pseudo-finish point.

Improve BFS Algorithm Experiment
For example, the start and finish are selected and aimed at with the circle as the start and the square as the finish point. The BFS tree and its search results are as shown in Figure 6.

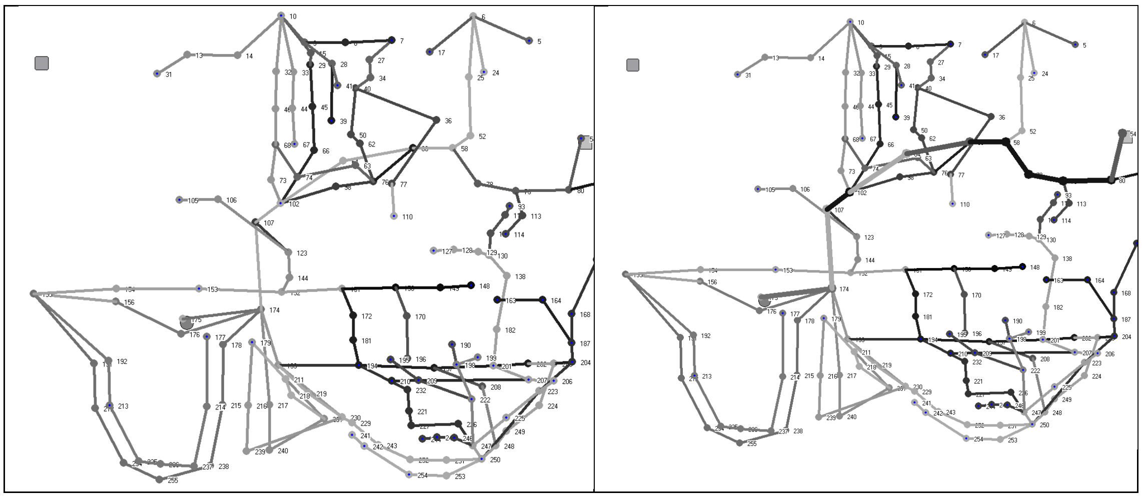

Figure 6. BFS tree freight line and its search results

Figure 6 is the BFS tree formation from the graff figure 5 with a start of 175 finish points of 54. Based on the experiment takes time: 27 seconds 564 milliseconds. The route is 175 =11=> 174 =17=> 107 =2=> 102 =17=> 64 =9=> 60 =2=> 58 =2=> 78 =2=> 79 =2=> 80 =9=> 54 meaning that from point 175 go up line 11 and get off at 174, take line 17 and get off at 107, go up line 2 and get off at 102, and so on ending at point 54.
Using Improve BFS with the same starting point 175 and the same finish point also 54, the tree and search results are formed as Figure 7.

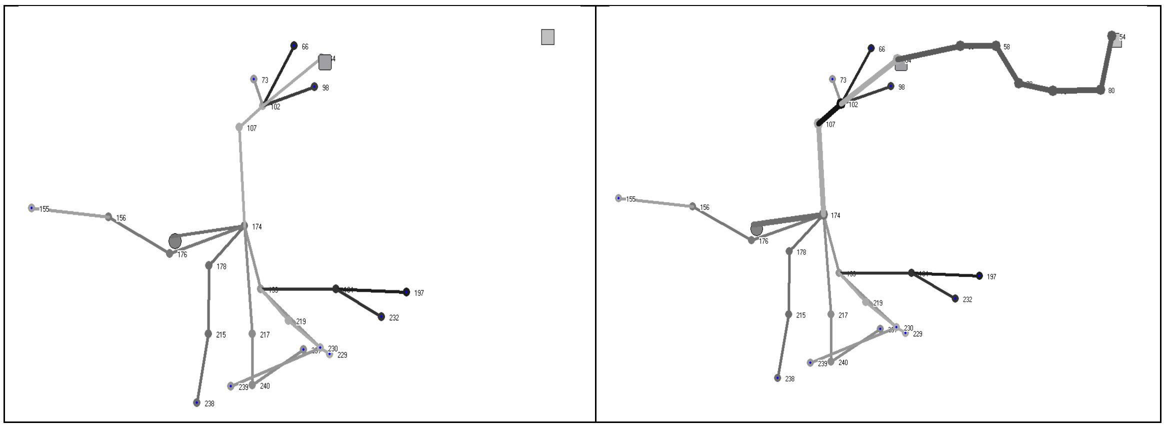

Figure 7. Tree Improve BFS freight lines and their search results

In Figure 7, many deleted dots will speed up the search process. Based on the experiment takes time: 0 seconds 171 milliseconds. The route is 175 =11=> 174 =17=> 107 =2=> 102 =17=> 64. Point 64 is the apparent finish point depicted with rounded square dots. After getting to this pseudo-finish point, continue up line 9 stop if you have reached the destination point 54, with the points passed, namely: 64==>60==>58==>78==> 79==>80==>54.

CONCLUSIONS

This study explored several public transportation routes that are often used. The route is obtained from the output process of the shortest route search algorithm. The shortest route algorithm used uses BFS. As the experiment above with a starting point of 175 and an end point of 54 the BFS algorithm takes 27 seconds 564 milliseconds, while the Improve BFS algorithm takes 171 milliseconds. The results of the comparison of the BFS and Improve BFS methods show that the Improve method provides better performance, which is much faster to produce search results.

REFERENCES

[1] E. Mogaji, I. Adekunle, S. Aririguzoh, and A. Oginni, "Dealing with impact of COVID-19 on transportation in a developing country: Insights and policy recommendations," *Transp Policy (Oxf)*, vol. 116, pp. 304–314, Feb. 2022, doi: 10.1016/j.tranpol.2021.12.002.
[2] A. Mourad, J. Puchinger, and C. Chu, "A survey of models and algorithms for optimizing shared mobility," *Transportation Research Part B: Methodological*, vol. 123, pp. 323–346, May 2019, doi: 10.1016/j.trb.2019.02.003.

[3] D. Dwi Rita Nova and N. Widiastuti, “Pembentukan Karakter Mandiri Anak Melalui Kegiatan Naik Transportasi Umum,” *Comm-Edu (Community Education Journal)*, vol. 2, no. 2, p. 113, May 2019, doi: 10.22460/comm-edu.v2i2.2515.

[4] M. Agustien, E. Buchari, and dkk, “Sosialisasi Pelayanan Teman Bus Sebagai Upaya Meningkatkan Minat Masyarakat Menggunakan Layanan Angkutan Umum Di Kota Palembang,” *Community Jurnal Pengabdian*, vol. 4, no. 1, pp. 29–38, 2022.

[5] A. Dardas, B. Hall, J. Salter, and H. Hosseini, “A geospatial workflow for the assessment of public transit system performance using near real-time data,” *Transactions in GIS*, vol. 26, no. 4, pp. 1642–1664, Jun. 2022, doi: 10.1111/tgis.12942.

[6] F. Wang, M. Ye, H. Zhu, and D. Gu, “Optimization Method for Conventional Bus Stop Placement and the Bus Line Network Based on the Voronoi Diagram,” *Sustainability*, vol. 14, no. 13, p. 7918, Jun. 2022, doi: 10.3390/su14137918.

[7] St. M. Hafran, M. T. Syarkawi, I. Syafei, I. Munsyir, and S. Saleh, “Analisis Kinerja Angkutan Umum BMA (Studi Kasus Rute Pinrang – Makassar PP),” *PENA TEKNIK: Jurnal Ilmiah Ilmu-Ilmu Teknik*, vol. 4, no. 2, p. 111, Jan. 2021, doi: 10.51557/pt_jiit.v4i2.590.

[8] K. Meneses-Cime, B. Aksun Guvenc, and L. Guvenc, “Optimization of On-Demand Shared Autonomous Vehicle Deployments Utilizing Reinforcement Learning,” *Sensors*, vol. 22, no. 21, p. 8317, Oct. 2022, doi: 10.3390/s22218317.

[9] C. Yang, X. Chen, X. Lin, and M. Li, “Coordinated trajectory planning for lane-changing in the weaving areas of dedicated lanes for connected and automated vehicles,” *Transp Res Part C Emerg Technol*, vol. 144, p. 103864, Nov. 2022, doi: 10.1016/j.trc.2022.103864.

[10] Z. Buako, L. Yahya, and N. Achmad, “Aplikasi Algoritma Floyd-Warshall Dengan Pendekatan Madm Dalam Menentukan Rute Terpendek Pengangkutan Sampah,” *Euler : Jurnal Ilmiah Matematika, Sains dan Teknologi*, vol. 9, no. 2, pp. 62–70, Oct. 2021, doi: 10.34312/euler.v9i2.10979.

[11] A. Ibraeva, G. H. de A. Correia, C. Silva, and A. P. Antunes, “Transit-oriented development: A review of research achievements and challenges,” *Transp Res Part A Policy Pract*, vol. 132, pp. 110–130, Feb. 2020, doi: 10.1016/j.tra.2019.10.018.

[12] C. Iliopoulou and K. Kepaptsoglou, “Integrated transit route network design and infrastructure planning for on-line electric vehicles,” *Transp Res D Transp Environ*, vol. 77, pp. 178–197, Dec. 2019, doi: 10.1016/j.trd.2019.10.016.

[13] N. Agatz, M. Hewitt, and B. W. Thomas, “‘Make no little plans’: Impactful research to solve the next generation of transportation problems,” *Networks*, vol. 77, no. 2, pp. 269–286, Mar. 2021, doi: 10.1002/net.22002.

[14] P. A. Maldonado Silveira Alonso Munhoz *et al.*, “Smart Mobility: The Main Drivers for Increasing the Intelligence of Urban Mobility,” *Sustainability*, vol. 12, no. 24, p. 10675, Dec. 2020, doi: 10.3390/su122410675.

[15] S. L. Pardede, F. R. Athallah, Y. N. Huda, and F. D. Zain, “A Review of Pathfinding in Game Development,” *[CEPAT] Journal of Computer Engineering: Progress, Application and Technology*, vol. 1, no. 01, p. 47, May 2022, doi: 10.25124/cepat.v1i01.4863.

[16] L. Hao, Y. Wang, Y. Bai, and Q. Zhou, “Energy management strategy on a parallel mild hybrid electric vehicle based on breadth first search algorithm,” *Energy Convers Manag*, vol. 243, p. 114408, Sep. 2021, doi: 10.1016/j.enconman.2021.114408.

[17] Y. Wang and L. Feng, “An adaptive boosting algorithm based on weighted feature selection and category classification confidence,” *Applied Intelligence*, vol. 51, no. 10, pp. 6837–6858, Oct. 2021, doi: 10.1007/s10489-020-02184-3.

[18] M. F. Bernov, A. D. Rahajoe, and B. M. Mulyo, “Route Optimization of Waste Carrier Truck using Breadth First Search (BFS) Algorithm,” *JEECS (Journal of Electrical Engineering and Computer Sciences)*, vol. 7, no. 2, pp. 1293–1304, 2023, doi: 10.54732/jeecs.v7i2.23.

[19] S. Sularno, D. P. Mulya, R. Astri, and D. Mulya, “Determination of The Shortest Route Based on BFS Algorithm for Purpose to Disaster Evacuation Shelter,” *Scientific Journal of Informatics*, vol. 8, no. 1, pp. 33–42, May 2021, doi: 10.15294/sji.v8i1.27863.

[20] S. Sularno, R. Astri, P. Anggraini, D. Prima Mulya, and D. Mulya, “Geographical Information System of Bus and Travel Counter in Padang City Using BFS Method Based on Mobile Web,” *Scientific Journal of Informatics*, vol. 8, no. 2, pp. 304–313, Nov. 2021, doi: 10.15294/sji.v8i2.33117.

[21] E. Wang *et al.*, "Double Graph Attention Actor-Critic Framework for Urban Bus-Pooling System," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2023, doi: 10.1109/TITS.2023.3238055.

[22] H. Behrooz and Y. M. Hayeri, "Machine Learning Applications in Surface Transportation Systems: A Literature Review," *Applied Sciences*, vol. 12, no. 18, p. 9156, Sep. 2022, doi: 10.3390/app12189156.

[23] J. Wang, Y. Xie, S. Xie, and X. Chen, "Cooperative particle swarm optimizer with depth first search strategy for global optimization of multimodal functions," *Applied Intelligence*, vol. 52, no. 9, pp. 10161–10180, Jul. 2022, doi: 10.1007/s10489-021-03005-x.

[24] Y. Du, F. Li, T. Zheng, and J. Li, "Fast Cascading Outage Screening Based on Deep Convolutional Neural Network and Depth-First Search," *IEEE Transactions on Power Systems*, vol. 35, no. 4, pp. 2704–2715, Jul. 2020, doi: 10.1109/TPWRS.2020.2969956.

[25] O. Riansanti, M. Ihsan, and D. Suhaimi, "Connectivity algorithm with depth first search (DFS) on simple graphs," *J Phys Conf Ser*, vol. 948, p. 012065, Jan. 2018, doi: 10.1088/1742-6596/948/1/012065.

[26] A. Gaihre, Z. Wu, F. Yao, and H. Liu, "XBFS: eXploring Runtime Optimizations for Breadth-First Search on GPUs," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, New York, NY, USA: ACM, Jun. 2019, pp. 121–131. doi: 10.1145/3307681.3326606.

[27] T. N. Lina and M. S. Rumetna, "Comparison Analysis of Breadth First Search and Depth Limited Search Algorithms in Sudoku Game," *Bulletin of Computer Science and Electrical Engineering*, vol. 2, no. 2, pp. 74–83, Dec. 2021, doi: 10.25008/bcsee.v2i2.1146.

[28] Bonifacius Vicky Indriyono and Widyatmoko, "Optimization of Breadth-First Search Algorithm for Path Solutions in Mazyin Games," *International Journal of Artificial Intelligence & Robotics (IJAIR)*, vol. 3, no. 2, pp. 58–66, Nov. 2021, doi: 10.25139/ijair.v3i2.4256.

[29] Z. Guo, C. S. Lai, P. Luk, and X. Zhang, "Techno-economic assessment of wireless charging systems for airport electric shuttle buses," *J Energy Storage*, vol. 64, p. 107123, Aug. 2023, doi: 10.1016/j.est.2023.107123.

[30] L. Miteva, K. Yovchev, and I. Chavdarov, "Point-to-point Motion Planning with Obstacle Avoidance for Hard Constrained Redundant Robotic Manipulators," in *2021 XXX International Scientific Conference Electronics (ET)*, IEEE, Sep. 2021, pp. 1–5. doi: 10.1109/ET52713.2021.9579820.

[31] Q. Li *et al.*, "FPS: Fast Path Planner Algorithm Based on Sparse Visibility Graph and Bidirectional Breadth-First Search," *Remote Sens (Basel)*, vol. 14, no. 15, p. 3720, Aug. 2022, doi: 10.3390/rs14153720.

[32] K. Zhao, G. Yang, and S. Zhu, "Fire Site Map Construction Method Based on UWB Connectivity Fusion," in *2022 4th International Conference on Natural Language Processing (ICNLP)*, IEEE, Mar. 2022, pp. 575–579. doi: 10.1109/ICNLP55136.2022.00105.

[33] T. Liu *et al.*, "Towards Indoor Temporal-variation aware Shortest Path Query," *IEEE Trans Knowl Data Eng*, pp. 1–1, 2021, doi: 10.1109/TKDE.2021.3076144.

AUTHORS BIBLIOGRAPHY

SUPRIHATIN holds the academic title Assistant Professor in the field of Information Systems. He earned his Master degree from Gadjah Mada University in 2003. He holds Bachelor's degree in Computer Science from Gadjah Mada University in 1993. He has been a permanent lecturer at Universitas Ahmad Dahlan (UAD) since 1996. His courses are introduction artificial intelligence, visual programming and data structure.

IMAM RIADI holds the academic title of Professor in the field of Information Systems. He earned his Doctorate degree from Gadjah Mada University in 2014. He holds a Master's degree in Computer Science from Gadjah Mada University in 2004 and a Bachelor's degree in Electrical Engineering from Yogyakarta State University (UNY) in 2001. He has been a permanent lecturer at Universitas Ahmad Dahlan (UAD) since 2002. His courses are Information Security, Computer Networking, Cyber Security and Digital Forensics.

FURIZAL was born in Kesra, Riau, Indonesia, in 1999. He received the bachelor's degree from the Department of Informatics Engineering, Universitas Islam Riau, in 2022, and the master's degree from the Department of Informatics, Universitas Ahmad Dahlan, in 2023. From 2017 to 2018, he was an instructor of computer courses at Yuri Computer. Since 2021, he has been a web developer at Faculty of Engineering, Universitas Islam Riau. His research interests include artificial intelligence, fuzzy sets, control system, Internet of Things, and web development.

AHMAD AZHARI (Member, IAENG) received his bachelor's degree from the Department of Informatics from Universitas Islam Indonesia, Indonesia in 2011 and received Master of Engineering (M.Eng.) in Electrical Engineering from Universitas Gadjah Mada in 2015. He is currently an Assistant Professor in Department of Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia. Member of Association for Scientific Computing and Electronics, Engineering (ASCEE). His current research interests include Brain Science, Pattern Recognition, Signal Processing, and Perseptual Psychology.