Journal of Information Technology and Computer Science
Volume 4, Number 3, Desember 2019, pp. 253-266
Journal Homepage: www.jitecs.ub.ac.id

Prototype of REST Location-based Service for Trip Planning by Paratransit (Mikrolet) in Malang City

Agi Putra Kharisma[1], Aryo Pinandito[2]

Universitas Brawijaya, Fakultas Ilmu Komputer, Malang, Indonesia
{[1]agi, [2]aryo}@ub.ac.id

Received 31 July 2019; accepted 29 November 2019

Abstract. Paratransit (mikrolet) was very popular in Malang city. However, their popularity is reduced by the availability of wide variety of more modern and online public transportations. One of the obstacles commonly faced by passengers of mikrolet is the limited information about the routes taken by mikrolet. In this paper, we proposed a prototype of service that can generate trip planning by mikrolet. We use depth-first search (DFS) algorithm as the basis to build our solution because it is relatively simple to modify. The service for trip planning available through REST web service and utilize location-based service style. Based on functionality testing result, the proposed system successfully generate all trip plans as expected. From performance perspective, when the number of routes and plans increased, the overall system performance degrades, so in the future the proposed prototype and its algorithm may needs further optimization.

1 Introduction

Paratransit in Malang city is also known as mikrolet. Most of mikrolet are owned and operated by individuals. Along with the times, paratransit should be able to satisfy the passengers because more and more alternative modes of public transportation are available. The availability of information is among the most significant factor affecting the quality of paratransit services [1]. One of the obstacles commonly faced by passengers, especially inexperienced passengers, of mikrolet is the limited information about the routes taken by mikrolet. As a result, inexperienced passengers often experience difficulties and eventually switch to another mode of transportation. The availability of route information is very important as one of factors that affects the quality of public transportation [2]. Research in Bandung city revealed that the most needed requirement by passenger was information on the paratransit route [3]. With the availability of travel route information, passengers are expected to make trip plans before taking an actual trip with mikrolet.

Mikrolet operate via a predetermined and fixed route called lyn. An example of lyn is AH which stands for Arjosari - Hamid Rusdi. Arjosari and Hamid Rusdi are the starting and ending locations of the mikrolet trips. Like the characteristics of paratransit in general, mikrolet can pick up or drop off their passengers in almost anywhere in the route they take and are not required to stop at bus stop or station. This characteristic distinguishes between mikrolet and buses or trains that are regularly scheduled and only stop at certain places.

In our previous study, a representational state transfer application programming interfaces (REST APIs) was proposed to serve information about local public transportation in Malang city [4]. These REST APIs was designed to accommodate all kind of local public transportation in Malang city, including paratransit (mikrolet), school bus, rickshaw (becak) and motorcycle taxi (ojek). One of the main features of that REST API is to provide trip planning. The APIs is designed and implemented based on REST architectural style because REST is scalable, using general interfaces [5] and can be implemented on hypertext transfer protocol version 1.1 (HTTP/1.1) seamlessly. So, the clients can consume the services easily. So, our solution in this study is based on that API design with slight modification.

In order to generate trip plans, the REST API that will be developed should accepts input in the form of origin and destination location by utilizing location-based services. Location-based Services are mobile services for providing information that has been created, compiled, selected or filtered under consideration of the user's current locations or those of other persons or mobile devices [6]. Location-based services can improve public transportation services for people with or without impairments [7]. In this study, location-based services are used in the process of identifying the origin location automatically, while the destination location is determined manually by the passenger. The client applications of REST API can utilize GPS sensors on their device to produce origin location coordinates that are quite accurate. Nowadays GPS sensors on smartphones are getting more accurate and smartphone technology has been widely used in public transportation. Smartphone has been used as trip planning system on smart tourism technologies to enhance travel satisfaction and it has been used exploratively and exploitatively [8].

Among the opportunities and challenges in the use of smartphones in public transportation is route planning. The smartphones application can be used to optimize routes and select modes of transportation in certain areas. Mobile applications can also be used to assist multi destination trip planning by public transportation [9]. The route and trip planning calculation process can be done on a smartphone or done on a server. This study chose to put the calculation process on the server that can be accessed via REST API to reduce the computational load on smartphones because the resources on smartphones are relatively limited.

This study focuses on the problem of how to implement a location-based trip planning method by utilizing REST APIs technology and certain path finding algorithm. The problem in this study is not to find the closest route or shortest path, but to look for all possible and reasonable routes or trip plans available. To generate all possible routes or trip plans, we modify depth-first search (DFS) algorithm for traversing graph data structures. DFS has the advantage of being easily modified because this algorithm is quite generic [10]. DFS has also been used as a method in planning trips by public transportation [11].

DFS algorithm is not the fastest algorithm for graph traversing. In terms of performance, DFS algorithms tend not to be linear. As the number of vertices in graph increases, the time needed for the DFS algorithm increases higher and resembles an exponential growth [12]. In this study, we will also measure the performance of our modified DFS algorithm.

2 Data Analysis and Modeling

Basically, mikrolet travel based on fixed and pre-determined routes according to its lyn. In its journey, one lyn may intersect or pass several segments of the same road with another lyn. In this condition, passengers can usually transit to move from one lyn to

another lyn. But sometimes passengers need to walk when transiting to move to another lyn if the lyn don't meet each other on the same road. In this study, we did not take into account the transit process which involved walking by passengers.

In reality, mikrolet can pick up or drop off their passengers in almost anywhere in the route they take and are not required to stop at bus stop or station. In our proposed model, we represent mikrolet travel routes in the form of directed graphs (digraph), where the edges represent roads and vertices represent places that can be used for pick up or drop off passengers. The more the number of vertices, the more precise the location of pick up or drop off passengers because the more the number of vertices it will increasingly resemble a line (road) as in Figure 1. On graphs with fewer vertices, passengers may be directed to locations farther than they should be as shown in right side example of Figure 1. However, as a tradeoff, the greater number of vertices will require higher computation requirements.

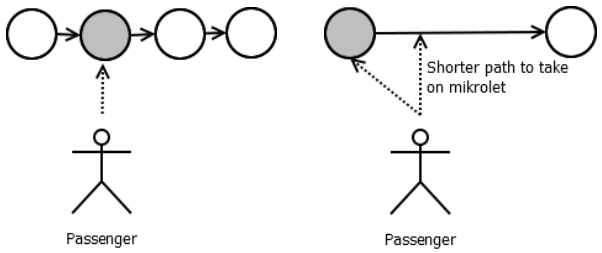


Fig. 1. The comparation of greater number of vertices (left) and lesser number (right) of vertices

Generally, the process of planning a travel with mikrolet can be seen in Figure 2. First of all the passenger determines the origin and destination locations along with the maximum tolerance distance to reach the mikrolet and reach the destination since passengers may have to walk to reach the mikrolet and reach the destination. The proposed system records all locations (origin and destination) that are within the reach of passengers. In Figure 2, gray nodes are all origin locations and black nodes are all destination origin within passenger range, while the labels A, B, C and D are lyn. Then the system will process these inputs with a special algorithm described in the algorithm design section of this paper to generate trip plans.

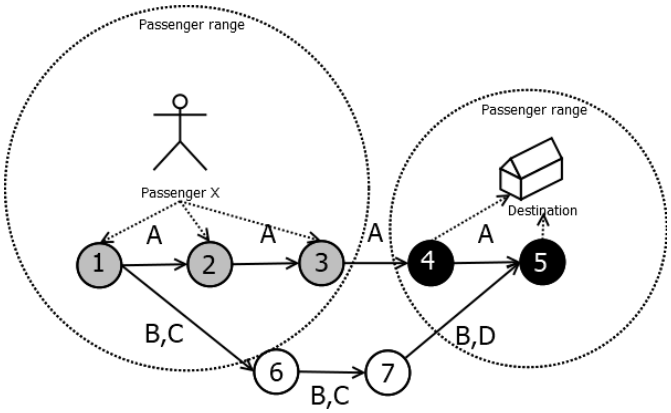


Fig. 2. General process of trip planning with mikrolet

In order to implement the proposed system, object-oriented development is used to produce prototypes. So that all entities involved in the trip planning process are modeled in the form of classes and their relationships (UML class diagram) as shown in Figure 3.

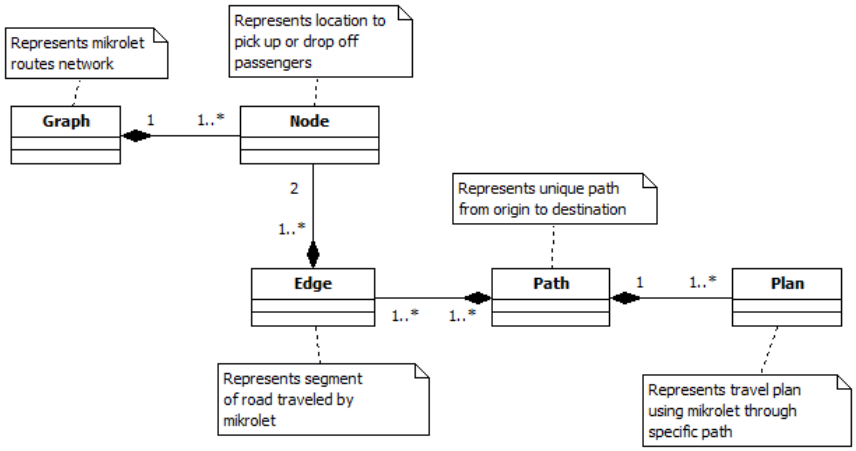


Fig. 3. Structural model of prototype

3 Design of Algorithm

The problems to be solved in this study is modeled in the form of graphs. In Figure 3 it has been stated that there is a Graph class which is a representation of a mikrolet routes network. Mikrolet route network contains all of the stops for mikrolet to pick up or drop off their passengers (represented by vertices of graph) along with all lanes that are passed by lyn (represented by edges of graph). To simplify the explanation, we use an example based on the hypothetical conditions in Figure 2. Based on Figure 2 we can illustrate the mikrolet route network as in Figure 4.

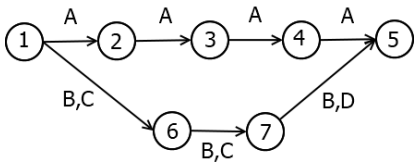


Fig. 4. Example of mikrolet route network (based on Figure 2)

The first step of the trip planning algorithm is to determine all origin locations and destination locations based on the passenger range. In the example in Figure 2, the origin locations are nodes 1,2 and 3 while the destination locations are nodes 4 and 5. Measurement of the distance between the passenger and the location of origin / destination is done using the Haversine formula, as in equation 1 [13]:

$$d = 2r\ arcsin\sqrt{sin^2\left(\frac{\varphi_2-\varphi_1}{2}\right) + cos(\varphi_1).\cos(\varphi_2).sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)} \quad (1)$$

Where:

d	= distance between two locations (in kilometers)
r	= the radius of earth (6371 kilometers)
φ1 dan φ2	= latitude 1 and latitude 2 (in radians)
λ1 dan λ2	= longitude 1 and longitude 2 (in radians)

The second step of the trip planning algorithm is to generate all path from all origin locations to all destination locations. In the example in Figure 2, there are 3 origin locations and 2 destination locations, so there are 6 pairs (3*2) of origin and destination locations. Each pair will generate all possible paths using the "FindAllPath" algorithm in the following pseudocode:

```
Algorithm FindAllPaths(G, o, d):
Input: A graph G,
       an ordered list of visited node o in G,
       a destination node d in G
Output: All paths p from o to d

for each adjacent nodes n in last element of o
  if o contains n then
    continue
  end if
  if n = a then
    add a to last element of o
    add o to p
    break;
  end if
end for

for each adjacent nodes n in last element of o
  if (o contains n) or (n = d) then
    continue
  end if
  add n to last element of o
  FindAllPaths(G, o, d);
  remove last element of o;
end for
```

The “FindAllPaths” algorithm is a modification of the DFS algorithm by adding the recording process to the history of graph traversing. From the example in Figure 2, all the generated paths can be seen in Table 1.

Table 1. All generated paths (based on Figure 2)

No	Origin Node	Destination Node	Path
1	1	4	12 – 23 – 34
2	2	4	23 – 34
3	3	4	34
4	1	5	12 – 23 – 34 – 45
5	1	5	16 – 67 – 75
6	2	5	23 – 34 – 45
7	3	5	34 – 45

The third step of the trip planning algorithm is to generate all of possible plans. All

possible trip plans are the result of cartesian products from all lyn on all paths. Cartesian product is a set of all ordered pairs of elements from two sets [14]. In this case, the ordered pairs consist of lyn. Then the cartesian product results are filtered to produce a unique trip plans. The detailed step-by-step processes to generate all of possible plans can be seen in the "GenerateAllPlansInPath" algorithm in the following pseudocode:

```
Algorithm GenerateAllPlansInPath(H):
Input: A path H
Output: A set of plans P

create a cartesian product P of all lyn in all edges in H

for each plan p1 in P
  for each plan p2 in P
    if cost of p1 > cost of p2
      if all lyn in p1 are also in p2
        remove p2 from P
      end if
    else if cost of p1 = cost of p2
        if distance of mikrolet stops from passenger in p2
        is farther than in p1
          remove p2 from P
    end if
  end for
end for
```

When the passenger transit to exchange to another lyn, the cost of plan will be incremented. To calculate the cost of plan, we use following algorithm named “SetCost”:

```
Algorithm SetCost(p)
Input: A plan p
Output: A cost c

set c := 1
for each lyn l1 in p
  if exist next lyn l2 after l1
    if l2 <> l1 then
      increment c by 1
    end if
  end if
end for
```

The example results of cartesian products with their cost from all lyn on all paths based on Figure 2 can be seen on Table 2.

Table 2. Result of all plans generated by cartesian products (based on Figure 2)

No	Path	Plan	Cost
1	12 – 23 – 34	A, A, A	1
2	23 – 34	A, A	1
3	34	A	1
5	12 – 23 – 34 – 45	A, A, A, A, A	1
6	16 – 67 – 75	B, B, B	1
7	16 – 67 – 75	B, C, B	3
8	16 – 67 – 75	B, B, D	2

9	16 – 67 – 75	B, C, D	3
10	16 – 67 – 75	C, B, B	2
11	16 – 67 – 75	C, B, D	3
12	16 – 67 – 75	C, C, B	2
13	16 – 67 – 75	C, C, D	2
14	23 – 34 – 45	A, A, A	1
15	34 – 45	A, A	1

The final step of "GenerateAllPlansInPath" algorithm is filtering all plans by distance of mikrolet stops from passenger and plan cost. Some plans may be discarded in this step. The example of final results of all generated plans based on Figure 2 can be seen in Table 3.

Table 3. Final result of all generated plans (based on Figure 2)

No	Origin Node	Destination Node	Plan
1	2	5	A
2	1	5	B
3	1	5	C, D

4 Development of Prototype

All trip plans generated from the computational process based on the proposed algorithm is published in the form of a REST web service so that it can be consumed easily by various computers, smartphones, or other devices. The general description of the proposed Web Service system architecture can be seen in Figure 5. All data communication between clients and web service use HTTP GET.

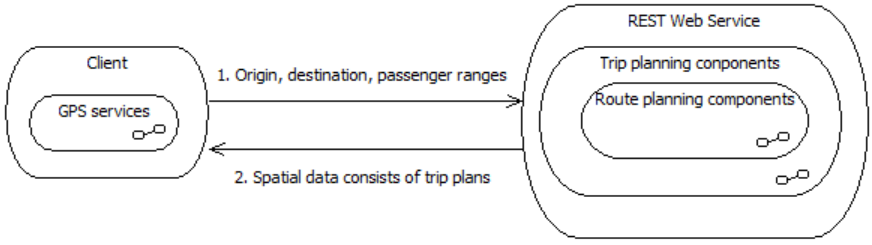


Fig. 5. Proposed system architecture

The prototype of REST webservice service developed in this study has only one functionality, which is to produce all alternative trip plans with mikrolet. When requesting services, the client must provide four input parameters as follows:

1. Coordinate of origin
2. Coordinate of destination
3. The passenger range in origin
4. The passenger range in destination

The client makes a request by sending an HTTP GET request, then the system will return an XML document. In detail the prototype of web service specifications developed can be seen in Table 4.

Table 4. Specification of REST web service prototype

Service name	Trip plan with mikrolet

Service URL template	`/tripplan?originlat=[originlat]`
	`&originlng=[originlng]`
	`&destinationlat=[destinationlat]`
	`&destinationlng=[destinationlng]`
	`&originrange=[originrange]`
	`&destinationrange=[destinationrange]`

Service URL example	`/travelplan?originlat=-7.93621921539307`
	`&originlng=112.65914916992188`
	`&destinationlat=-7.93516683578491`
	`&destinationlng=112.65661621093750`
	`&originrange=100`
	`&destinationrange=100`

Request method	`GET`
Response media type	`application/xml`

Response representation template	`<plans>`
	`  <plan distance="[distance]"`
	`  cost="[cost]"`
	`  distance-to-origin="[distance-to-origin]"`
	`  distance-to-destination=`
	`  "[distance-to-destination]">`
	`  <routes>`
	`  <route`
	`  originlat="[originlat]"`
	`  originlng="[originlng]"`
	`  destinationlat="[destinationlat]"`
	`  destinationlng="[destinationlng]"`
	`  lyn="[lyn]"/>`
	`  </routes>`
	`  </plan>`
	`</plans>`

Response representation example	`<plans>`
	`  <plan distance="303.38717738776126"`
	`  cost="1"`
	`  distance-to-origin="0.0"`
	`  distance-to-destination="0.0">`
	`  <routes>`
	`  <route originlat="-7.93621921539307"`
	`  originlng="112.65914916992188"`
	`  destinationlat="-7.93599605560303"`
	`  destinationlng="112.658447265625"`
	`  lyn="AG-O"/>`
	`  <route originlat="-7.93599605560303"`
	`  originlng="112.658447265625"`
	`  destinationlat="-7.93576192855835"`
	`  destinationlng="112.65779876708984"`
	`  lyn="AG-O"/>`
	`  <route originlat="-7.93576192855835"`
	`  originlng="112.65779876708984"`
	`  destinationlat="-7.93544387817383"`
	`  destinationlng="112.65711212158203"`
	`  lyn="AG-O"/>`
	`  <route originlat="-7.93544387817383"`

```
      originlng="112.65711212158203"
      destinationlat="-7.93516683578491"
      destinationlng="112.6566162109375"
      lyn="AG-O"/>
  </routes>
</plan>
<plan distance="303.38717738776126"
  cost="2"
  distance-to-origin="0.0"
  distance-to-destination="0.0">
  <routes>
    <route originlat="-7.93621921539307"
      originlng="112.65914916992188"
      destinationlat="-7.93599605560303"
      destinationlng="112.658447265625"
      lyn="GA-O"/>
    <route originlat="-7.93599605560303"
      originlng="112.658447265625"
      destinationlat="-7.93576192855835"
      destinationlng="112.65779876708984"
      lyn="GA-O"/>
    <route originlat="-7.93576192855835"
      originlng="112.65779876708984"
      destinationlat="-7.93544387817383"
      destinationlng="112.65711212158203"
      lyn="AL-O"/>
    <route originlat="-7.93544387817383"
      originlng="112.65711212158203"
      destinationlat="-7.93516683578491"
      destinationlng="112.6566162109375"
      lyn="AL-O"/>
  </routes>
</plan>
<plan distance="303.38717738776126"
  cost="2"
  distance-to-origin="0.0"
  distance-to-destination="0.0">
  <routes>
    <route originlat="-7.93621921539307"
      originlng="112.65914916992188"
      destinationlat="-7.93599605560303"
      destinationlng="112.658447265625"
      lyn="GA-O"/>
    <route originlat="-7.93599605560303"
      originlng="112.658447265625"
      destinationlat="-7.93576192855835"
      destinationlng="112.65779876708984"
      lyn="GA-O"/>
    <route originlat="-7.93576192855835"
      originlng="112.65779876708984"
      destinationlat="-7.93544387817383"
      destinationlng="112.65711212158203"
      lyn="AT-O"/>
    <route originlat="-7.93544387817383"
```

```
                originlng="112.65711212158203"
                destinationlat="-7.93516683578491"
                destinationlng="112.6566162109375"
                lyn="AT-O"/>
            </routes>
        </plan>
    </plans>
```

In response representation example in Table 4, there are fields with name "distance-to-origin" and "distance-to-destination". These two fields are the distance that the passenger should travel in order to get on the mikrolet from passenger's initial location ("distance-to-origin") or to arrive on destination from the place the passenger gets off the mikrolet ("distance-to-destination"). From example in Figure 2, the "distance-to-origin" is the distance between Passenger X and node 1, node 2 or node 3 and the "distance-to-destination" is the distance between node 4 or node 5 and destination. The generated trip plans in form of XML response from REST web service developed in this study can be visualized on map as shown in Figure 6. Each lyn is given a different color to illustrate the trip plan that passengers can follow.

Fig. 6. Visualization example of trip plan by lyn A, B and C

5 Testing and Evaluation

Testing and evaluation are carried out to determine the correctness of prototype functionality and the characteristics of prototype performance. The correctness of prototype functionality is determined by experimentation with example data as in Figure 2. We test the system as a black-box. All test cases and results in our functionality testing can be seen on Table 5.

Table 5. Functionality test results

No	Input	Expected Result	Result	Status
1	Location and range of Passenger X	Plan 1: A Plan 2: B Plan 3: C,D	Plan 1: A Plan 2: B Plan 3: C,D	Passed

Performance testing is done with dummy data to obtain an initial description of the

performance characteristics of the prototype developed. The test case is taken based on the combination of the original location and destination location as follows:

1. The origin node at the end of the graph, the destination node on the other end of the graph
2. The origin node in the middle of the graph, the destination node at the end of the graph
3. The origin node at the end of the graph, the destination location in the middle of the graph
4. The origin node at the end of the graph, the destination node at the same end of the graph

Each test case will be executed 3 times and measured the estimated execution time needed for the prototype to produce a route plan. Performance is measured from the time needed when the REST Web Service gets a request to produce a response in milliseconds (ms).

In order to determine the initial performance of data processing time, testing and evaluation are done through a simulation process with dummy data. The environment for testing is as follows:

1. Hardware specifications:
   a. Processor : Intel Core i5 3320M
   b. RAM : 8GB DDR3 800MHz
   c. Hard Disk : 5400 rpm SATAIII
2. Software specifications:
   a. Operating System : Microsoft Windows 7 64-bit
   b. Java Environment : JDK 10 64-bit
   c. Web Server : Apache Tomcat 9 64-bit
3. Test Data:
   a. Number of graph node adjacency lists : 1096
   b. Number of lyn : 4

Based on the results of performance testing which can be seen in Figure 7, it appears that there is a degradation in performance along with the increasing number of routes found. With more routes being found, the greater the computational process for determining trip plans. This indicates that the proposed algorithm needs to get further optimization in terms of efficiency.

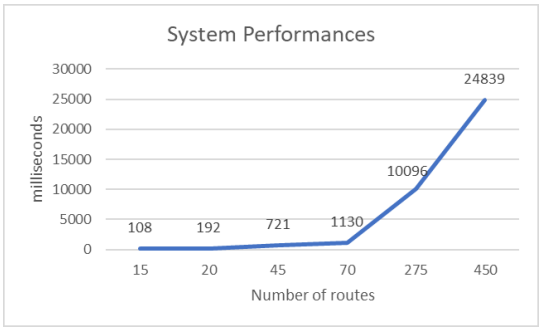


Fig. 7. Performance testing results of proposed prototype

As a comparation, those results showed in Figure 7 resemble the performance characteristics of the DFS algorithm as shown in Figure 8. However, overall performance of proposed trip plans algorithm doesn't scale as well as DFS due to additional processes (modifications) that are not exist in the original DFS. Those processes include recording visited node and edges, generating cross product of all lyn

and generating trip plans.

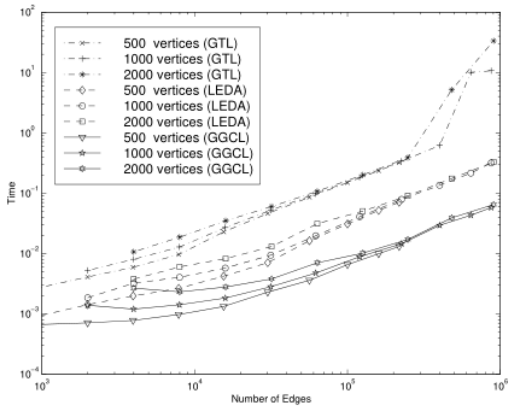
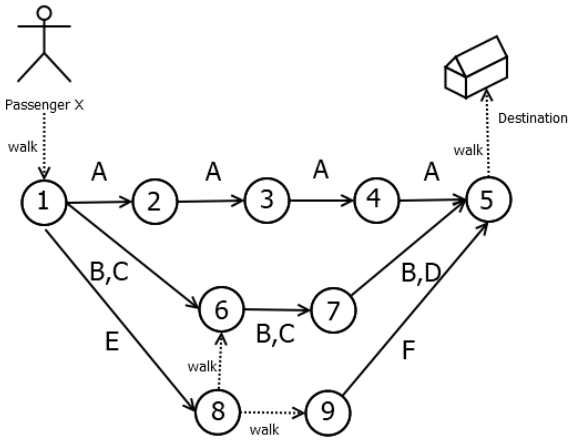


Fig. 8. Performance of DFS algorithm on various library [15]

6 Discussions

Based on the results of performance tests, it is known that performance has decreased dramatically when the number of routes produced increases. Among the causes are the characteristics of the DFS algorithm. Other studies on ride-sharing and fixed-route taxi algorithms also experience performance problems and require an optimization in algorithm performances [16] [17]. An adaptation or modification of dynamic programming (DP) algorithm can be used to improve the performance of the trip planning algorithm [18].

The trip planning algorithm proposed in this study only supports direct transfers from one lyn to another lyn. In other words, transfers between lyn only occur when both



lyn meet at a same location. However, in the real world scenario, transfers may be made indirectly. Passengers may have to walk before moving to another lyn. Illustration of transfers between lyn indirectly can be seen in Figure 9.

Fig. 9. Trip planning with indirect transfers

In order to travel to destination, passenger X should following one of these plans:

1. Walk to take on A on node 1, take off from A on node 5, walk to destination
2. Walk to take on B on node 1, take off from B on node 5, walk to destination
3. Walk to take on C on node 1, take off from C and transfer to D on node 7, take off from D on node 5, walk to destination
4. Walk to take on E on node 1, take off from E on node 8, walk to take on B on node 6, take off from B on node 5, walk to destination
5. Walk to take on E on node 1, take off from E on node 8, walk to take on C on node 6, take off from C on node 7 and transfer to D, take off from D on node 5, walk to destination
6. Walk to take on E on node 1, take off from E on node 8, walk to take on F on node 9, take off from F on node 5, walk to destination

The process of transfers from E to B, E to C and E to F is indirect transfer.

Yet another issue is in the process of determining trip plans. The parameters used in trip planning algorithm includes the cost and distance, but do not include the total travel time. From the results of observations in the field it is known that there is no definite relationship between time and distance. Long distances do not necessarily result in long travel times and short distances do not necessarily result in short travel times. Thus, in the future it is necessary to develop a method to estimate the total travel time to make a better trip planning system.

7 Conclusion and Future Work

Based on this study, it can be concluded that the prototype of trip planning services can be realized in the form of REST web services and location-based services. Based on experiment on functionality testing, the proposed algorithm has succeeded in producing all possible routes and plans that connect origin locations to destination locations. The XML output of REST web service can be visualized on map. However, from the results of performance testing it can be concluded that there is significant performance degradation with the increasing number of generated routes. Future works need to deal with open issues described in discussion section. The open issues include development of optimized algorithm to produce higher performance, development of plans with indirect transfer and development of method to estimate travel time by paratransit.

References

1. Joewono TB, Kubota H.: User satisfaction with paratransit in competition with motorization in indonesia: anticipation of future implications. In: Transportation (2007) 34(3) pp. 337–354.
2. Siuhi, Saidi and Mwakalonge, Judith.: Opportunities and challenges of smart mobile applications in transportation. In: Journal of Traffic and Transportation Engineering (English Edition) (2016) Vol. 3, Issue 6, pp. 582-592
3. Priscilia, Prisca. and Octavia, Johanna Renny.: Pengembangan Aplikasi Mobile untuk Mempermudah Pencarian Informasi Rute Angkutan Kota Di Bandung. In: PERFORMA Vol. 16, No 1 Maret (2017) pp. 62-71
4. Kharisma, Agi Putra and Pinandito, Aryo.: Design of REST API for Local Public Transportation Information Services in Malang City. In: Journal of Information Technology and Computer Science, Vol. 2, No. 2, pp 92-102. Universitas Brawijaya (2017)
5. Fielding, Roy Thomas.: Architectural Styles and the Design of Network-based Software Architectures. Ph.D. dissertation, University of California, Irvine. Available: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.
6. Kupper, Axel.: Location-Based Services: Fundamentals and Operation. John Wiley & Sons, Inc., , USA (2005)

7. Chang, Shwu-Jing., Hsu, Gong-Ying., Huang, Shian-Jia.: Location-aware mobile transportation information service. In: Mobile Technology, Applications and Systems (2005) 2nd International Conference (2005), S. 5 pp. -5
8. Huang, C. Derrick., Goo, Jahyun., Nam, Kichan., Yoo, Chul Woo.: Smart tourism technologies in trip planning: The role of exploration and exploitation.. In: Information & Management, 54 (2017), Nr. 6, S. pp. 757-770
9. Chowdhuri, Subeh. and Giacaman, Nasser.: En-Route Planning of Multi-Destination Public-Transport Trips Using Smartphones. In: Journal of Public Transportation, Vol. 18, No. 4 (2015) pp. 31-45
10. Goodrich, Michael T. and Tamassia, Roberto.: Algorithm Design and Applications (1st ed.). Wiley Publishing (2014).
11. Song, Cuiying., Guan, Wei., Ma, Jihui.: Potential travel cost saving in urban public transport networks using smartphone guidance. In: PLoS One. (2018);13(5):e0197181. doi:10.1371/journal.pone.0197181
12. Lee, Lie-Quan, Jeremy G. Siek, and Andrew Lumsdaine.: The generic graph component library. In: ACM SIGPLAN Notices (1999), vol. 34, no. 10, pp. 399-414.
13. Smart, William Marshall.: Textbook on Spherical Astronomy, 6th ed. Cambridge University Press (1977).
14. Rosen, Kenneth H.: Discrete Mathematics and Its Applications, 7th ed. McGraw-Hill (2012).
15. Lee, Lie-Quan., Siek, Jeremy G., Lumsdaine, Andrew.: The generic graph component library. In: ACM SIGPLAN Notices, vol. 34, no. 10, pp. 399-414. (1999)
16. Babaei, Mohsen., Schmöcker, Jan-Dirk., Khademi, Navid., Reza Ghaffari, Ahmad., Naderan, Ali.: Fixed-route taxi system: route network design and fleet size minimization problems. In: Journal of Advanced Transportation 50, no. 6 (2016) pp. 1252-1271
17. Bei, Xiaohui, and Zhang, Shengyu.: Algorithms for trip-vehicle assignment in ride-sharing. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018).
18. Tong, Yongxin., Zeng, Yuxiang., Zhou, Zimu., Chen, Lei, Ye, Jieping., Xu, Ke.: A unified approach to route planning for shared mobility. In: Proceedings of the VLDB Endowment 11, no. 11 (2018) pp. 1633-1646