A Dijkstra’s Mobile Web Application Engine for Generating Integrated Light Rail Transit Route

A. M. Haziq Lim¹, N. S. Wan Sazli², B. Hussin³, A. A. Azlianor⁴, S. M. Suhaizan⁵, K. Massila⁶
Faculty of Information and Communication Technology
Universiti Teknikal Malaysia Melaka
Locked Bag 1752 Pejabat Pos Durian Tunggal
76109 Durian Tunggal, Melaka, Malaysia

Abstract- In the capital city of Malaysia, Integrated Light Rail Transit (LRT) System is one of the most importance public transportation as it connects some key districts to historical, interesting places, business areas and shopping malls are concentrated. The train services are running independently but have interchanges to integrate from one different LRT lines to the others. These interchanges could cause traveler a troublesome when they are choosing incorrect destination station especially on different LRT lines which contribute to time consuming and high costing. In previous research we are implemented the mobile web application using rules based algorithm where the destination-oriented routes need to be dynamically generated by determining the nearest station according to the specific places. Currently, our research is focusing on Dijkstra’s algorithms to provide more effective and intelligent shortest path system to provide the solution for traveler to reach the desired destination. In this paper, we discuss the results from our Dijkstra’s application prototype.

Key-Words: - Mobile application, Shortest path, Dijkstra, Dynamic route map

1.0 INTRODUCTION

At some of Asia's megacities, light railway systems (LRT) become crisscrossed including bustling Kuala Lumpur that allowing even harried business travelers to keep their trip on track and is said to be the most convenient public transport. In Kuala Lumpur, there are three lines operated by different companies which are part of Kuala Lumpur's Integrated Transit Network and it is a fully independent metro network. RapidKL operates the Kelana Jaya Rail Line and the Ampang Rail Line whereas KL Monorail operates Monorail Line. This integrated transportation network transports approximately 4 million passengers every week with 908 buses and 48 rail stations operating daily [10]. See Fig. 1.

Fig. 1 KL’s Integrated LRT Network [5]
Traveling from one appointment to another in this bustling metropolis can be difficult to the users. This research addresses the problem of selecting nearest destination station to a given places of interest such as shopping places, historical places and interesting places which may involve at least two LRT lines and need of interchange station. Thus, the traveler may face difficulty when they choose incorrect station especially on different LRT lines which led to time consuming and high cost [8].

Producing a mobile application that could use in handheld device such as Personal Digital Assistant (PDA) for traveler on-the-move has becoming wide spread. People percentage having access to mobile devices is rapidly increasing and need to have access to information anytime, anywhere. Therefore, we propose mobile web based application to assist travelers to travel via LRT.

2.0 RELATED WORK

With the rapid growth of the means of communication, mobile technologies have played an important role in the diffusion of information as well as business activity. People need to have access to information anytime, anywhere. Mobile services appear to be an obvious choice for travel and tourism as the travelers are on the move, which is the first criterion for mobile services to be relevant [9].

The travel and tourism industry have been undergoing many dramatic changes during the last decade, due to the possibilities offered by Internet technology. PDAs and smart phones are becoming increasingly capable and more affordable, and they provide an excellent platform for business Web applications running a variety of software applications from some mini version of management tools to text, document or even PDF readers [6].

The importance of supporting mobility of users has also been argued recently whereby there are many mobile frameworks/architectures has adopted technology based systems such as adaptive mobile applications [1], mobile commerce application [2], mobile web services [3], mobile tourist services [4] and mSpace Mobile [7]. These trends have created a growing need for applications that take advantage of mobile environments.

Many factors contribute to the success (or failure) of a mobile solution. These include the mobile device, wireless network connectivity, enterprise integration, and most important, the application architecture. Many people do not realize that several application models are available for mobile development, each with a different set of characteristics that make it appropriate for some applications and inappropriate for others [11].

Research [22], show that genetic algorithm demonstrates favorable performance on solving the combinatorial optimization problems and Dijkstra’s is an exact algorithm. It always determines the optimal route.

Other research [23], already apply the Dijkstra’s in to their system and from the testing, the system was able to provide a
path, an alternate path and map
directions to the users depending on the
requested service. The system will
respond to the users request using
mobile device and reply with a SMS or
MMS.

3.0 MOBILE WEB APPLICATION
ARCHITECTURE

In this section, we describe the mobile
architecture that is composed of a
number of components for developing
mobile application. Figure 3 gives an
overview of the relevant components
and how they interact. In this
architecture, we distinguish two sides, a
client-side and a server-side. They play
distinctive roles and require different
mechanisms for their implementation.

Client and Server Integration

In Fig. 3, each component integrates as
client-server application by
communicating between client-side and
server-side applications where consists
of 2 and 4 components respectively.
Client-side has User Interface and Java
Application components whereas server-
side has Process Control, Map
Generation, Optimization Route, and
Data Management components.

Mobile client using PDA will
communicate with a web server through
an HTTP request via mobile browser (or
micro browser). It is essential of using
mobile browser in the sense of its
capability like image and browser
capability and the types of markup
languages supported for web-based
application. Due to the existing
limitations of portable devices, the User
Interface component needs to consider
its screen size and resolution whereas the
Java Application Client able to execute,
if and only if the PDA has java-enabled
browser.

In order to enable mobile client to
communicate, the Apache is installed
with configured PHP as web application
server. Apache will serve as web server
whereas PHP will be used as application
server. Process Control component will
act as interface to process inputs and
organize it prior to respond to the mobile
clients. After it receives request from
mobile client, this component will send
to Optimization Route component to
obtain shortest, fastest, and cheapest
route to desired destination by accessing
it to database (MySQL) via Data
Management component. Next, it will
generate the map to visualize the route
and respond back to mobile client for
him/her to decide. This is how mobile web applications work across mobile clients.

### 4.0 OPTIMIZATION ROUTE ENGINE

A rule based classification method is used on the subset to identify new rules to describe the subset [19] and will result a tree like graphs. Each node represents an attribute test, while leaves represent conclusions. Therefore, each path from the root to a leaf can be interpreted as a decision rule [19].

In our previous work, we already implement the destination route system used rules based technique [20]. The system was developed in two versions, one for web browse and the second version is for mobile view. With this system, users can find a route to the selected destinations.

Three main properties for rules based methods are **locality**, **detachment** and **truth-functionality**. Locality in logical system, whenever we have a rule of the form $A \Rightarrow B$, we can consider $B$ given $A$, without worrying about any other rules [18]. Based on that, our rules are set for example from **Bangsar on Kelana Jaya Line** to **Plaza Rakyat on Ampang Line** we have 5 routes using our rules based system as show in Figure 5. It presents the total ticket price to represent the shortest path and in this case, route 1 and 2 is the chippers with only RM1.20.

In this system, we already set the rules which station the users need to change train. Using this method, if one of the stations in between the changing station is not operating because of technical problems, the system can not identify it while the rules still the same. So, to find the shortest path, rules based technique is not very practical as we need to consider all others path and station connected, either it operating or not. This is because, the connection of each station can be change or disconnect.

---

**Fig. 3: Interface of Destination Route System used Rule Based**
Fig. 4: Route using Rules Based

Based from this previous research [20], we successfully built a prototype for testing purpose of our rules based destination route by using an open source, cross-platform implementation and implement the Dijkstra’s algorithms. This prototype was built to enhance the route calculation and system for alpha testing [24][25]. The complete version will be developing for actual testing.

The used of algorithms help making the route performance more optimized and dynamic. If one or more station is not working, Dijkstra’s algorithms can performs new calculation and update the route. Not like the rule based technique where we found that it’s hard to detect if one or few station is out of service or not working.

**Dijkstra’s Algorithms**

There are others algorithm that can be used to find the shortest path. Dijkstra’s algorithm is perhaps one of the most efficient algorithms [12, 13] that can be used for shortest path problem. Dijkstra’s is modeled as finding the shortest path between two nodes in a weighted network \( G = (N,A) \) with node set \( N \) and arc set \( A \). In route finding, \( G \) is the road network [14].

Research [15] show that by combining Dijkstra’s shortest path algorithm with location-based services (LBS) result a highly efficient in route plan than previous used ways by only cumulative approaches.

Dijkstra’s algorithms can be applied on various platforms using deference technique. For examples by using AS3 [16] or Excel [17], Dijkstra’s algorithms still can be applied. In this study, we apply the Dijkstra’s algorithms to perform and find the shortest destination route of train transportation in Kuala Lumpur area. The used of algorithms basically to enhance the effectiveness and reliability of destination route. Dijkstra’s algorithms solves the single-source path and produce a graph with nonnegative edges path costs, producing the shortest path tree. Each route will be calculated to find the shortest path. Each path will have a weighted either the distance, cost or time take. We will use the train ticket price as the weighted in our research. For the prototype and testing purpose, we will used and modify
from our previous project. Figure 6 below show the 13 stations that involved in finding the shortest path from Bangsar to Plaza Rakyat. We used the ticket price as the path value to find the shortest path which we define by the cost saving route.

Fig. 5: Paths from Bangsar to Plaza Rakyat

Bangsar station is point 9 while Plaza Rakyat is point 40. Based on the point above, we create the metric as below to show the connection between stations that create a path:

<table>
<thead>
<tr>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>25</th>
<th>26</th>
<th>27</th>
<th>28</th>
<th>29</th>
<th>30</th>
<th>31</th>
<th>40</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Stations connection metric

Next, we will insert the weight for each point connection. The weight will be the ticket. Other weight option can also be used such as time taken, destination or any that suite. Below is the step by step operation of Dijkstra’s from [21]:

Step 1: point to nodes reachable from the startnode.
The distance to: b=10. There are no other arrows coming in to b.
Node b will be colored orange to indicate 10 is the length of the shortest path to b.

Step 2: point to nodes reachable from nodes that already have a final distance.
The distance to: c=20, f=22. Node c has the minimum distance.
There are no other arrows coming in to c.
Node c will be colored orange to indicate 20 is the length of the shortest path to c.

Step 3: point to nodes reachable from nodes that already have a final distance.
The distance to: d=30, f=22. Node f has the minimum distance.
There are no other arrows coming in to f.
Node f will be colored orange to indicate 22 is the length of the shortest path to f.

Step 4: point to nodes reachable from nodes that already have a final distance.
The distance to: e=40, g=34. Node d has the minimum distance.
There are no other arrows coming in to d.
Node d will be colored orange to indicate 30 is the length of the shortest path to d.

Step 5: point to nodes reachable from nodes that already have a final distance.
The distance to: e=40, g=34, m=42. Node g has the minimum distance.
There are no other arrows coming in to g.
Node g will be colored orange to indicate 34 is the length of the shortest path to g.

Step 6: point to nodes reachable from nodes that already have a final distance.
The distance to: e=40, h=46, m=42. Node e has the minimum distance.
There are no other arrows coming in to e.
Node e will be colored orange to indicate 40 is the length of the shortest path to e.

Step 7: point to nodes reachable from nodes that already have a final distance.
The distance to: h=46, l=42, m=42. Node m has the minimum distance.
Any other path to m visits another red node, and will be longer than 42.
Node m will be colored orange to indicate 42 is the length of the shortest path to m.
Step 8: point to nodes reachable from nodes that already have a final distance. The distance to: h=46, l=42. Node l has the minimum distance. There are no other arrows coming in to l. Node l will be colored orange to indicate 42 is the length of the shortest path to l.

Step 9: point to nodes reachable from nodes that already have a final distance. The distance to: h=46, k=54. Node h has the minimum distance. Any other path to h visits another red node, and will be longer than 46. Node h will be colored orange to indicate 46 is the length of the shortest path to h.

Step 10: point to nodes reachable from nodes that already have a final distance. The distance to: k=54. There are no other arrows coming in to k. Node k will be colored orange to indicate 54 is the length of the shortest path to k.

Step 11: point to nodes reachable from nodes that already have a final distance. The distance to: j=66. There are no other arrows coming in to j. Node j will be colored orange to indicate 66 is the length of the shortest path to j.

Step 12: point to nodes reachable from nodes that already have a final distance. The distance to: i=78. There are no other arrows coming in to i. Node i will be colored orange to indicate 78 is the length of the shortest path to i.

Algorithm has finished, follow orange arrows from startnode to any node to get the shortest path to the node.

As shown in Fig 6, the shortest route will be station 9, 10, 25, 26, 27 and 28 with a cost only RM7.00. Thus, if any station in between the route is not working, alternative shortest route will be proposed.

5.0 CONCLUSION AND FUTURE WORK

Mobile applications are indispensable for everyone who needs to deliver robust, high-value mobile solutions. This paper presents the results and comparisons of two our previous research regarding the proposed idea of mobile web application engine using Dijkstra’s algorithms. We have described the motivation for an intelligent route system. Next, we plan to build a complete version of our dynamic
LRT route system that used open source, cross-platform implementation of Dijkstra’s-DRM for mobile devices, and trial this implementation at actual environments. We also might consider the used of PHP and database for better optimizing in administration side of the system. GUI will be enhancing. It is hoped that it is useful for travelers and capable to help them to make their decision easily, directly and successfully ubiquitously.

Acknowledgement

This work has been supported by Ministry of Higher Education, Malaysia under the Fundamental Research Grant Scheme (FRGS), no. FRGS/2007/FTMK (6 – F0053 and by University Industry Center (UNIC), UTeM coordination action.

References


[23] Joel C. de Goma, Jose F. Abisado Jr. Paolo Alekseis P. Carolino, Juan Carlos L. Guverarra and Joemerick B. San Gabriel. Route Navigator: Path Finder Service For Mobile Phone. 2007

[24] A.M. Haziq Lim, N.S. Wan Sazli, B. Burairah, S.M. Suhaizan, A.A. Azlianor and K. Massila and (2009), A Mobile Web Application Engine for...
Generating Destination-oriented LRT Route, Proceeding of The 9th WSEAS International Conference on Applied Computer Science, WSEAS Genova, Italy