Locating Nearest Public Facility using IDA* Algorithm

Esther Irawati Setiawan
Departement of Computer Science
Sekolah Tinggi Teknik Surabaya
Surabaya, Indonesia
esther@stts.edu

Joan Santoso
Departement of Computer Science
Sekolah Tinggi Teknik Surabaya
Surabaya, Indonesia
joan@stts.edu

Indra Maryati
Departement of Computer Science
Sekolah Tinggi Teknik Surabaya
Surabaya, Indonesia
maryati@stts.edu

***Abstract* – Nowadays public transportation is needed given the current increasing transportation needs in the community because of the density of private vehicles that causes congestion everywhere. Given these reasons, currently people prefer to use public transportation to get to a location or perform daily life activities. But the problem is lack of information of public facilities and routes of available public transportation. This research will propose a system for searching public facility locations with implementation of IDA* algorithm. The input of this system is user's location from a mobile device and the output will be a list of routes that can be viewed in a map.**

Keywords - IDA*; Pencarian Lokasi Terdekat; Kendaraan Umum; Lyn; Lokasi Fasilitas Umum;

I. INTRODUCTION

Currently people's mobility is increasing thus the need of personal transport vehicle to support daily activity is crucial. This causes congestion and density of traffic. Society then turns to public transportation as the solution. Effective time management is important as well. During busy work period, people can't afford to waste time in the traffic jam so they need to be able to travel from work to home in most minimum time.

Lyn is one of the most popular public transportation in the Indonesian community, especially Surabaya. Almost every people have used this kind of public transportation. Though Lyn is popular, user doesn't have sufficient information on all available routes. User only knows Lyn routes near their work and home area.

Based on this definition, in this research we will propose a system for solving these issues. A search algorithm in artificial intelligence will be implemented in the system. In this section we have described the background and objective of this research. In the second section, we will describe the proposed system architecture and detailed input and output of the system. Description of the system proposed especially IDA* algorithm for shortest route problem will be outlined in the third section. The fourth and fifth section will elaborate methods of trial and conclusion of the given proposed system.

II. SYSTEM ARCHITECTURE

This section describe the system architecture that will be created and proposed as a research prototype used to address some issues that have been discussed previously. Details of the proposed system architecture can be seen in figure 1.

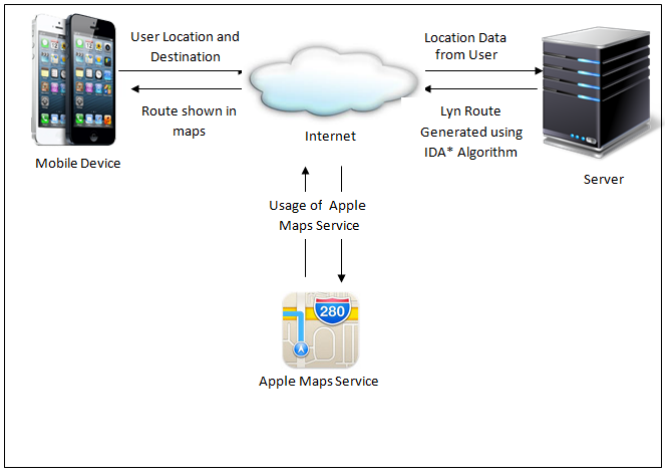

Figure 1. System architecture.

In this system, the input is user's location and user's destination. User's location can be obtained from user's mobile device. These data will be transferred to the service by internet. The service will be developed in PHP [1]. The server will process the data and generate a list of routes using ID* Algorithm [2]. User can then chose a preferred route. The result will also be displayed in an Apple Map [3] on a mobile device. Example of input and output from this system is displayed on figure 2.

978-1-4673-2317-8/12/$31.00©2013 IEEE

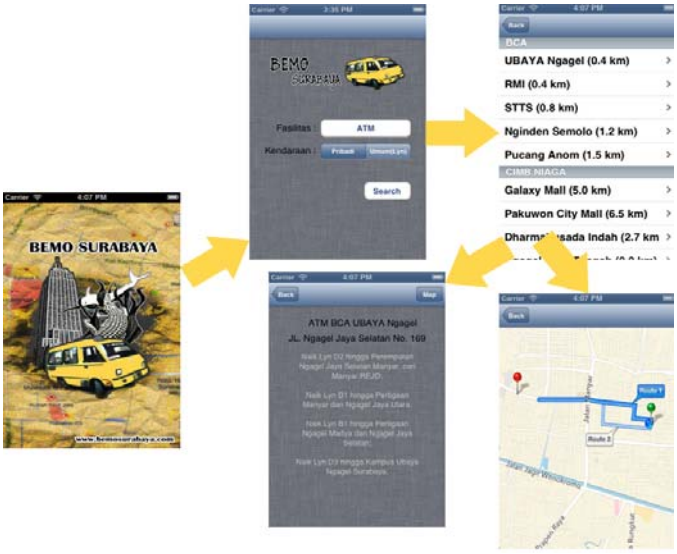

Figure 2. Input and output example.

In the server web application, there are four subsystems that will handle the request from mobile device application. The first subsystem is finding the nearest location which is used to find the nearest public facilities location, such as ATM, hospital, etc. The second subsystem is finding the nearest route which is used to find the route from user location to public facilities that have been searched before. The third subsystem is finding the public transportation Lyn that can be used to cover the route. The last subsystem is for data processing. Data that will be used in this application are public facilities information, public transportation Lyn and available routes. Server side application architecture is displayed in figure 3.

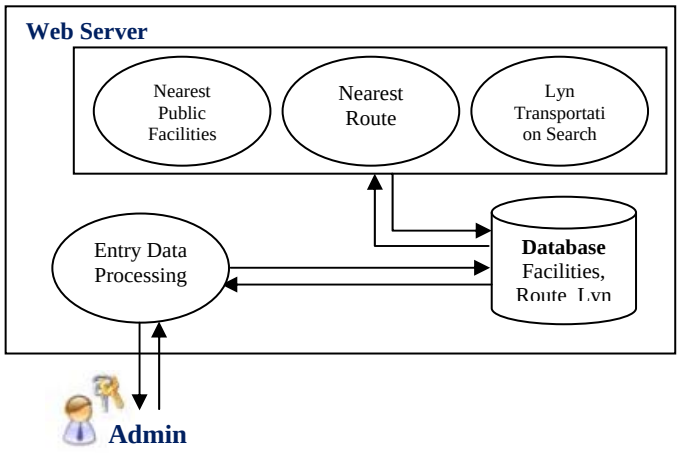

Figure 3. Web server architecture.

III. SHORTEST PATH ALGORITHM

In prior research [4], A* algorithm was choose to find a route from user location to destination location for public transportation Lyn. The proposed change is into IDA* algorithm [5], shown in figure 4, which is a development from A* algorithm.

A* algorithm has complete and optimal characteristic, similarly with IDA* algorithm. IDA* algorithm has advantage in memory usage, it requires less memory than A* algorithm. One of the problems in the previous research is limited memory availability so that system cannot run in real time. With the IDA* algorithm, it is expected that this proposed system can be the solution to this problem.

Though IDA* algorithm has an advantage on the amount of memory being used but there are also disadvantages. One of the disadvantage is due to the iterative characteristic of the IDA* algorithm that has been described in [6]. Due to the iterative characteristics, IDA* algorithm can generate the same nodes repeatedly so that the time complexity of this algorithm is quite high. However this is not a significant shortcoming because of this weakness can be covered by various advantages of this algorithm.

```
node              current node
g                 the cost to reach current node
f                 estimated cost of the cheapest path (root..node..goal)
h(node)           estimated cost of the cheapest path (node..goal)
cost(node, succ)  path cost function
is_goal(node)     goal test
successors(node)  node expanding function

procedure ida_star(root, cost(), is_goal(), h())
  bound := h(root)
  loop
    t := search(root, 0, bound)
    if t = FOUND then return FOUND
    if t = ∞ then return NOT_FOUND
    bound := t
  end loop
end procedure

function search(node, g, bound)
  f := g + h(node)
  if f > bound then return f
  if is_goal(node) then return FOUND
  min := ∞
  for succ in successors(node) do
    t := search(succ, g + cost(node, succ), bound)
    if t = FOUND then return FOUND
    if t < min then min := t
  end for
  return min
end function
```

Figure 4. IDA* algorithm.

IDA * algorithm in figure 4 will be executed in real time so that users can search for locations quickly and accurately. The result of this algorithm will be sorted by Merge Sort algorithm [7] so the user can choose from several routes provided. In the implementation, some threshold value can be given to determine the cost limit as a result of the algorithm used.

The reason for using Merge Sort algorithm is defined in an existing literature [9]. Merge Sort algorithm is stable and when implemented requires only a small amount of available storage memory. After the overall process is completed, the result route will be displayed on user's mobile device and the route is represented in map services available from the mobile device. In this system, we displayed the result on Apple Maps.

```
function merge_sort(list m)
    // if list size is 0 (empty) or 1, consider it sorted and return it
    // (using less than or equal prevents infinite recursion for a zero
    length m)
    if length(m) <= 1
        return m
    // else list size is > 1, so split the list into two sublists
    // 1. DIVIDE Part...
    var list left, right
    var integer middle = length(m) / 2
    for each x in m before middle
         add x to left
    for each x in m after or equal middle
         add x to right
    // recursively call merge_sort() to further split each sublist
    // until sublist size is 1
    left = merge_sort(left)
    right = merge_sort(right)
    // merge the sublists returned from prior calls to merge_sort()
    // and return the resulting merged sublist
    // 2. CONQUER Part...
    return merge(left, right)
```

```
function merge(left, right)
    // receive the left and right sublist as arguments.
    // 'result' variable for the merged result of two sublists.
    var list result
    // assign the element of the sublists to 'result' variable until there
    is no element to merge.
    while length(left) > 0 or length(right) > 0
        if length(left) > 0 and length(right) > 0
           // compare the first two element, which is the small one, of
           each two sublists.
            if first(left) <= first(right)
                // the small element is copied to 'result' variable.
                // delete the copied one(a first element) in the sublist.
                append first(left) to result
                left = rest(left)
            else
                // same operation as the above(in the right sublist).
                append first(right) to result
                right = rest(right)
        else if length(left) > 0
            // copy all of remaining elements from the sublist to 'result'
            variable, when there is no more element to compare with.
            append first(left) to result
            left = rest(left)
        else if length(right) > 0
            // same operation as the above(in the right sublist).
            append first(right) to result
            right = rest(right)
    end while
    // return the result of the merged sublists(or completed one, finally).
    // the length of the left and right sublists will grow bigger and
    bigger, after the next call of this function.
    return result
```

Figure 5. Merge Sort Algorithm

IV. TESTING

Testing need to be done to make an application run well and test the performance of the proposed system. Testing will be done on two different cases, one is on server side application and the other one will be on client side application.

Input	Output
Jenis kendaraan = umum	1.ATM BCA RMI – 0.449 km
Lokasi asal = Jalan Manyar Rejo	2.ATM BCASTTS – 0.691 km
Fasilitas yang dicari = ATM BCA	3.ATM BCA Ubaya Ngagel – 0.897 km
Aplikasi yang digunakan = Server	4.ATM BCA Pucang – 2.02 km

Figure 6. First testing.

In the first test case, the application will be tested in server side and use 10 queries to find the result route. From the queries will be seen the result given by the application and see how well the system can provide the results. In addition to tests carried out on the server side, some of the other tests will also be done on mobile device application. Test on mobile device application will use iPhone as the device.

Input for the testing is type of vehicle using by user, current user location, search facilities and application that being used. Current user location is obtained from device GPS system or user can enter street names. As the output user will provide the nearest search facilities and the route shown in Apple Maps.

Input	Output
Jenis kendaraan = umum	1.ATM Mandiri RMI – 1.263 km
Lokasi asal = Jalan Nginden Semolo	2.ATM Mandiri Ngagel Jaya Selatan – 2.028 km
Fasilitas yang dicari = ATM Mandiri	
Aplikasi yang digunakan = iPhone	

Figure 7. Second testing.

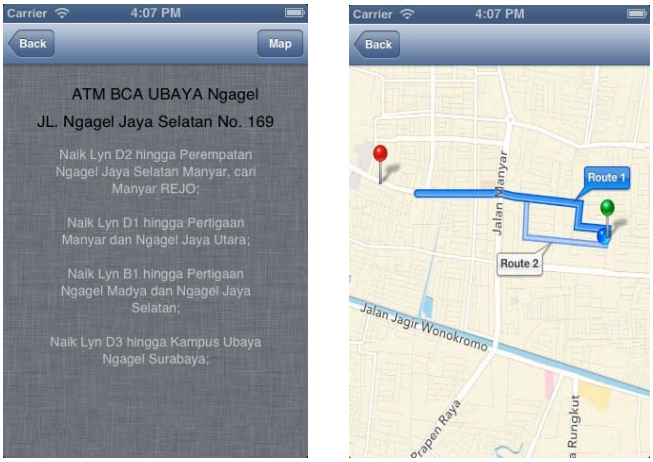

Figure 8. Result Lyn and Route in Apple Maps.

The other test is on Apple mobile device with iOS operating system. Factors that will be considered are:

1. Time taken by application to process a request service.
2. Accuracy of the results which are routes and GPS accuracy on the mobile device that can help improve the system overall.

From the two test cases, test results will also consider the relevance of the results by comparing the real time usage of this application on several users. This is crucial because there are several other factors that currently cannot be identified, and later maybe used as the input of the system. This relevance can be used as a benchmark of the system and

provide optimal solution of the system. One of the experiments that can be conducted is testing on *n* devices simultaneously to search 10 locations of the same public facilities from 2 different user locations.

V. Conclusion and Further Research

The developed system will be able to facilitate Surabaya citizen in using public transportation Lyn. Although the developed system is the improved from prior research, the system also have some limitations that can be improved in future research. Limitation that found in the current system is time factor and traffic jam that currently cannot be used as input and processing factor in the system.

For further research, the system can be developed by adding components that currently is not identified yet, so that the developed system can respond to the specific needs for Surabaya citizen and work out the limitation.

Acknowledgment

In this research the authors are very grateful for help and support given by fellow community members. Also this work was supported and fully funded by Directorate General of Higher Education, Ministry of Education and Culture of Indonesia.

References

[1] Robert Richards,“ Pro PHP XML and Web Services (Books for Professionals by Professionals)”,Apress, 2006.

[2] Stuart Russel and Peter Norvig, “Artificial Intelligence: A Modern Approach”, Upper Saddle River, N.J.: Prentice Hall, 2003.

[3] Apple, Maps, http://www.apple.com/ios/maps/, (Accessed 07 August 2013)

[4] Esther Irawati Setiawan, Gunawan, Indra Maryati, Joan Santoso, Rossy Prabowo Chandra. Shortest Path Problem for Public Transportation Using GPS and Map Service. Procedia – Social and Behavioral Sciences Volume 57(9 October 2012), Pages 426-431, 2012.

[5] Wikipedia, IDA*, http://en.wikipedia.org/wiki/IDA*. (Accessed 07 August 2013).

[6] Suyanto, Artificial Intelligence : Searching - Reasoning - Planning - Learning (Edisi Revisi), Penerbit Informatika, 2011.

[7] Donald Knuth,” Section 5.2.4: Sorting by Merging". *Sorting and Searching*.” The Art of Computer Programming. 3 (2nd ed.). Addison-Wesley, 1998, pp.158-168.

[8] Wikipedia, Merge Sort, http://en.wikipedia.org/wiki/Merge_sort, (Accessed 07 August 2013).

[9] Wikipedia, QuickSort, http://en.wikipedia.org/wiki/Quicksort, (Accessed 07 August 2013).