

FINDING THE SHORTEST PATHS AMONG CITIES IN JAVA ISLAND USING NODE COMBINATION BASED ON DIJKSTRA ALGORITHM

Bilqis Amaliah, Chastine Fatichah and Olyn Riptianingdyah
Informatics Engineering, Faculty of Information Technology,
Institut Teknologi Sepuluh Nopember
Jl. Raya ITS, Gedung Teknik Informatika, Surabaya 60111, Indonesia
Emails: bilqis@if.its.ac.id, chastine@cs.its.ac.id

Submitted: Aug. 16, 2016 ***Accepted: Nov. 14, 2016*** ***Published: Dec. 1, 2016***

Abstract- This study focuses on finding the shortest paths among cities in Java Island by repeatedly combining the start node's nearest neighbor to implement Dijkstra algorithm. Node combination is used to find the shortest path among cities in Java by deleting the node nearest to the start node. The use of memory by node combination is more efficient than the use of memory by the original Disjkstra algorithm. The 46 cities in Java Island will be used to evaluate the performance of finding shortest path. The experimental results show that the accuracy of node combination is 92.88% with the Google Map as the reference. The successful implementation of algorithm in finding the shortest path on the real problem is a good point; therefore, the algorithm can be developed to solve the transportation network problem.

Keywords*:* Shortest path problem, Dijkstra algorithm, Node combination, Transportation problem.

I. INTRODUCTION

In performing daily activities, transportation problem affects people's lives. To reach potential and remote areas, transportation network is needed in order to help develop the area. Therefore, many experts develop better methods to find the shortest paths which can help people reach their destinations faster and more efficiently. As the time passed by, and as the development of science and technology advances, there are a lot of algorithms used to solve the shortest path problem.

There are many researchers which try to find shortest path [1-28]. The first researchers propose Rapid Physarum Algorithm[1] to solve shortest path problem. The second researchers [2]propose a modified ant colony to find the shortest path in network and arc length used as probabilistic. The third researchers introduce a special case of Stochastic Shortest Path Problems to solve the shortest path problem [3]. The fourth researchers present the weighted region shortest path problem to find the shortest path in graph [4]. Researchers [5] design and implement a labeling technique to find shortest path trees.

Meanwhile, researchers [6] modify an original amoeba model to handle linear transportation problem. Authors [7] consider the problem of switching off network elements and adjust the weights to minimize the consumption of energy; this energy awareness is operated by shortest path routing. In the meantime, researchers [8] propose Fuzzy Physarum Algorithm, an algorithm which can handle the fuzzy shortest path problem effectively. Authors [9] present Neural Network model for solving the shortest path problems, which is replaced by Linear Programming. This network model is an improved version of the classic Dijkstra's algorithm by simple modification.

Researchers [10] propose fast algorithm which utilizes the previously-calculated results to accelerate the calculation. Meanwhile, a new technique for the addition of various fuzzy numbers in a path is devised by [11]. Since large, various fuzzy numbers can cause complexity, a genetic algorithm is presented to find the shortest path in the network.

There are three issues rise from [12]. In order to address the issues, the author uses Dijkstra algorithm. The experiment results show that the three issues have been effectively resolved to find the shortest path. One-way separators will be used in the graph to cross it only once. These separators give divide-and-conquer solutions to find the shortest paths between two vertices. This

purpose method is called parallel algorithm, which is used to find the Shortest Paths in Planar Layered Digraphs [13].

The shortest path algorithm is used to do texture analysis and to classify the texture in the literature [14]. The proposed method converts pixels of an image into verticesand finds the shortest paths between pairs of pixels. Literature [15] turns the bus stop matching problem into a shortest path problem. The bus stop was organized by route; and, the result transforms the stop matching problem into a shortest path problem. The improved branch-and-price algorithm can solve shortest path problem with resource constraints [16]. Multiobjective shortest path problems are more complex than the single objective one [17]. Pareto with label setting algorithm can make an optimal path in lexicographic goals. Private information Retrieval techniques is proposed by [18] to solve the shorthest path. The *k*-shortest paths algorithm is modified to the motion-based optimization of multiple-object tracking [19]. The dual graphs were built for acceleration-based and acceleration variation-based criteria. Incremental network design problems focused on optimal choice of network expansions [20] and with shortest paths. A fault toleran relay placement algorithm is used to find the shortest path in wireless sensor network[21]. This algorithm ensures *k* vertex-disjoint shortest paths in wireless sensor networks. The all-pairs shortest path problem adds crossover to the algorithm, this can significantly speed up a mutation in algorithm [22]. There are two concepts which are central in recombination, namely repair mechanisms and parent selection. Graph theory was used to analyze the pipeline optimal problem[23]. A Dijkstra algorithm was designed and analyzed to obtain N shortest paths. Author [25] propose Teaching-Learning-Based Optimization (TLBO) algorithm for unmanned aerial vehicle path planning. The experiment show that TLBO can find a better and shorter route. Vehicular ad hoc networks is a node with position and data transmission. Vehicular opportunity route based on real-time information (VORI) is proposed by [26] by using an energy efficient routing to collect real time data and help delivery message. Author [27] propose approximate path searching to get shortest path on road network. This method can find minimum path cost. Multi objective graph search is more complex than single objective. The researcher [28] proposed dimensionality reduction to find the shortest path in multi objective.

One of the notable algorithms and very commonly used in finding the shortest path is Dijkstra's algorithm. However, the use of Dijkstra's algorithm is not very simple and takes more memory. The Node Combination based on Dijkstra algorithm is proposed by [24]. First, the node

combination finds the nearest neighbor of the start node. Second, the node combination combines (deletes) the node with the start node, and then modifies the edge weights that connect to the nearest neighbor.

The purpose of this paper is to find the shortest path among cities in Java Island using the Node Combination which introduced by [24]. The Node Combination implement Dijkstra algorithm by deleting the node nearest to the start node. The use of memory is more efficient by deleting the node than the original Dijkstra algorithm. To evaluate the performance of the proposed method, 46 cities are used to find the shortest paths. The distances between the cities obtained from Google Maps on April 22, 2014.

This paper is organized as follows: the Node Combination based on Dijkstra algorithm is presented in Section 2, the use of Node Combination algorithm for finding the shortest paths by deleting the node nearest to the start node among cities in Java Island is introduced in Section 3, and the experimental results and conclusions the related problems are presented in Section 4 and 5 respectively.

II. THE NODE COMBINATION BASED ON DIJKSTRA ALGORITHM

Basic idea of the node combination is to combine nodes instead of maintaining the labeling sets in Dijkstra algorithm. The node combination method finds the shortest paths iteratively by finding the nearest neighbor of the start node, combining that node with the start and updates the edge weights connected to the nearest node.

The steps of the node combination method are as follows:

1. Determine the start node
2. Find the nearest node by looking at the smallest weights that are connected with the start node
3. Delete the node and update the edge weights that are connected with the start node. If there are two or more weights then choose the smallest one
4. Repeat the steps until only two nodes remains

The following is an example of node combination method that starts from Figure 1:

1. Node 1 is the start node. (See Figure 1)
2. The nearest node from 1 are 5 and 2. The nearest node from the start node is 2 with the weight of 2. (See Figure 2)

3. Delete node 2 and update the edge weight from node 1. The weight from node 1 to 3 is 2 + 5 = 7. There are two weights from node 1 to 5, which are 2+1 = 3 and 3. Because they have the same weights, then the weight from node 1 to 5 is 3. (See Figure 3)
4. The nearest node from 1 are 5 and 3. The nearest node from the start node is node 5 with the weight of 3. (See Figure 4)
5. Delete node 5 and update the edge weight from node 1. The weight from node 1 to 4 is 3+4=7. (See Figure 5)
6. The nearest node from 1 are 3 and 4. Because the two nodes has the same weight (7), the next node that is chosen is 3. The nearest node from the start node is node 3 with the weight of 7. (See Figure 6)
7. Delete node 3 and update the edge weight from node 1. There are two weights from node 1 to 4 that are 7+6 = 13 and 7. Then chose the minimum weight that is 7, so that the weight of the node 1 to 4 is 7. (See Figure 7)
8. The nearest node from node 1 is 4. (See Figure 8)
9. Delete node 4 and update the edge weight from node 1. The weight from node 1 to 6 is 7+7 = 14. (See Figure 9).

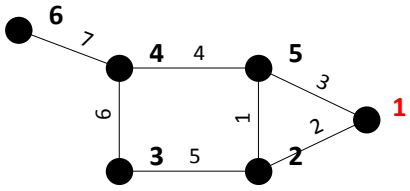

Figure 1. Start node: Node 1

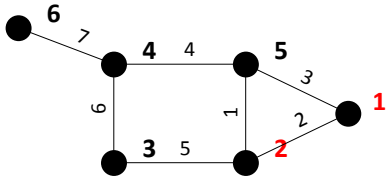

Figure 2. The nearest node of node 1: node 2

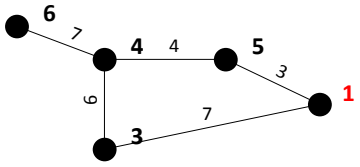

Figure 3. Deleting node 2

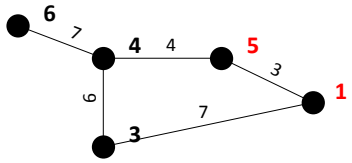

Figure 4. The nearest node of node 1: node 5

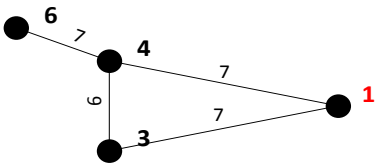

Figure 5. Deleting node 5

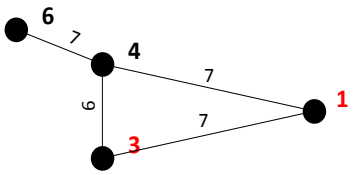

Figure 6. The nearest node of node 1: node 3

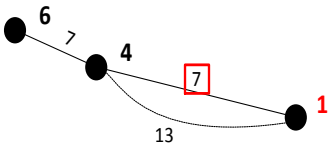

Figure. 7. Deleting node 3

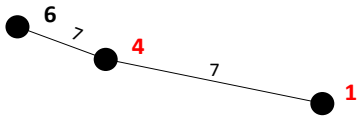

Figure 8. Finding the smallest weight and the nearest node of node 1: node 4

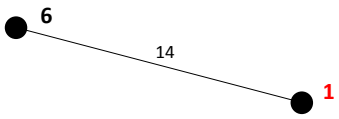

Figure 9. Deleting Node 4

III. FINDING THE SHORTEST PATHS AMONG CITIES IN JAVA ISLAND

Java Island is the fifth largest and the most populous island in Indonesia country. Transportation problems such as traffic jam, high shipping cost, finding the shortest path are commonly encountered in Java. This research aims to find the shortest path among cities in Java using the Node combination based on Dijkstra algorithm. There are 118 cities in Java; however, we use 46 cities which connect the eastern and western part of Java for evaluating the performance of this method. The visualization of connecting among 46 cities in Java is shown in Figure 10. And, the names of the 46 cities in Java we use for evaluating the shortest path results are shown in Table 1.

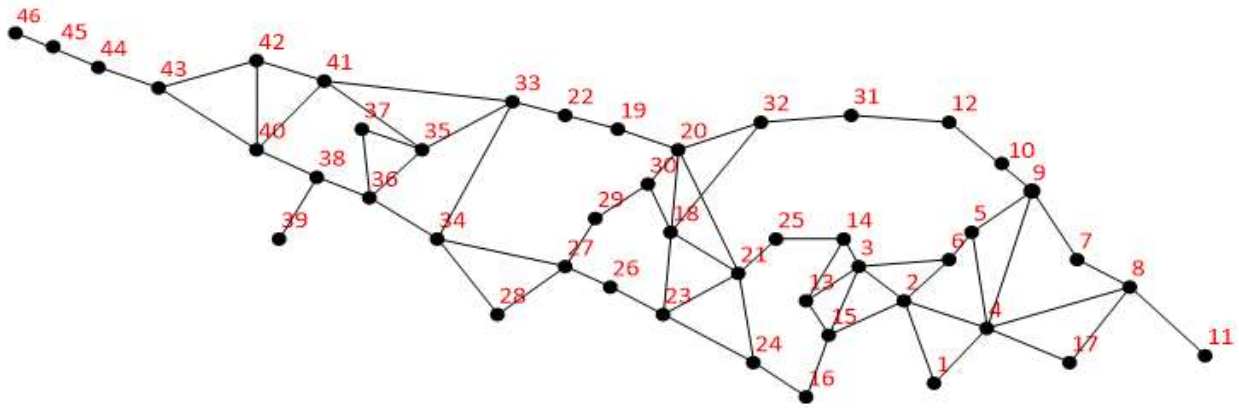

Figure 10. Map of 46 cities in Java Island

Table 1. The City names in Java Island

Node	City	Node	City	Node	City	Node	City	Node	City
1	Blitar	11	Banyuwangi	21	Surakarta	31	Rembang	41	Bekasi
2	Kediri	12	Tuban	22	Tegal	32	Kudus	42	Jakarta
3	Madiun	13	Magetan	23	Yogyakarta	33	Cirebon	43	Tanggerang
4	Malang	14	Ngawi	24	Wonosari	34	Tasikmalaya	44	Serang
5	Mojokerto	15	Ponorogo	25	Sragen	35	Subang	45	Cilegon
6	Jombang	16	Pacitan	26	Purworejo	36	Bandung	46	Merak
7	Pasuruan	17	Jember	27	Purwokerto	37	Purwakarta		
8	Probolinggo	18	Magelang	28	Cilacap	38	Cianjur		
9	Surabaya	19	Pekalongan	29	Wonosobo	39	Sukabumi		
10	Gresik	20	Semarang	30	Temanggung	40	Bogor		

The distance between start and destination node is taken from Google Maps on April 22, 2014, by the following steps:

1. Determine the start city and the destination city.
2. Obtain the route distance between two cities. This distance is achieved by activating the "Direction" feature on Google Maps. Names of the connected cities are typed on the feature, then the "Direction" feature comes up with the distance of the two cities.
3. If there are two or three distances shown on Google Maps, we choose the closest distance between the two cities.

Table 2 conveys samples of routes among cities in Java based on "Direction" feature on Google Maps.

Table2. Sample of distance between start and destination city in Java

Start – Destination City	Distance (km)
Surabaya – Malang	92.8
Gresik – Tuban	87.5
Surabaya – Malang	92.8
Malang – Blitar	75.3
Surabaya – Mojokerto	49.5
Malang – Kediri	102

IV. THE EXPERIMENTAL RESULTS

The cities that are chosen to be tested are shown in Table 1 (there are 46 cities). For this experiment, the start cities are Malang city, Banyuwangi and Surabaya.

Figure 11 show the result of Google Maps, from Surabaya city to Mojokerto city. The distance using Google Maps is 49.5 km and the distance using the proposed method is also 49.5 km. The path using the proposed method is same with Google Maps : Surabaya->Mojokerto

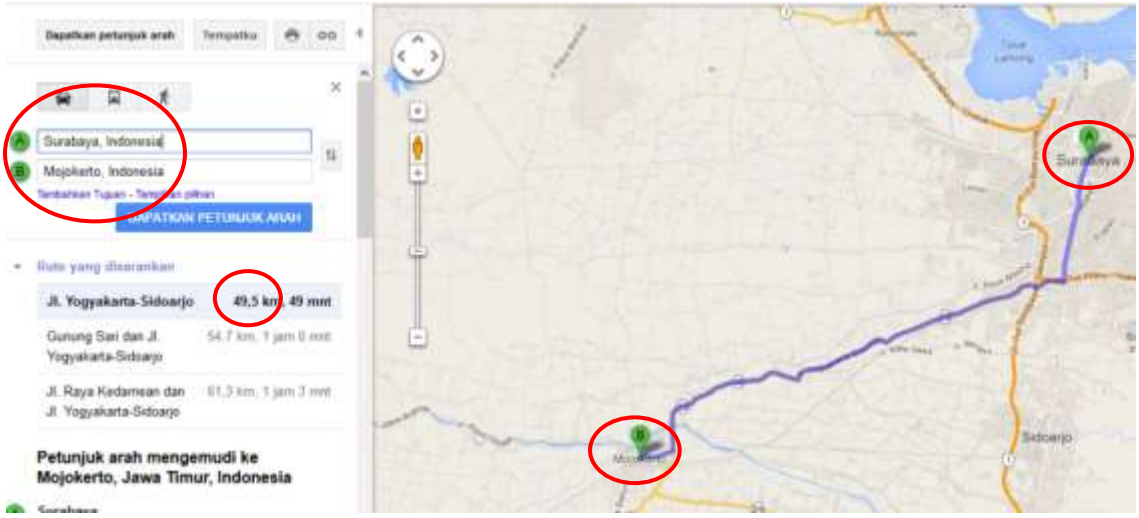

Figure 11. the result of Google Maps, from Surabaya to Mojokerto

Figure 12 show the result of Google Maps, from Surabaya city to Blitar city. The distance using Google Maps is 172 km and the distance using the proposed method is 168.1 km. This is smaller 3,9 km than Google Maps. The path using the proposed method is same with Google Maps : Surabaya->Malang->Blitar.

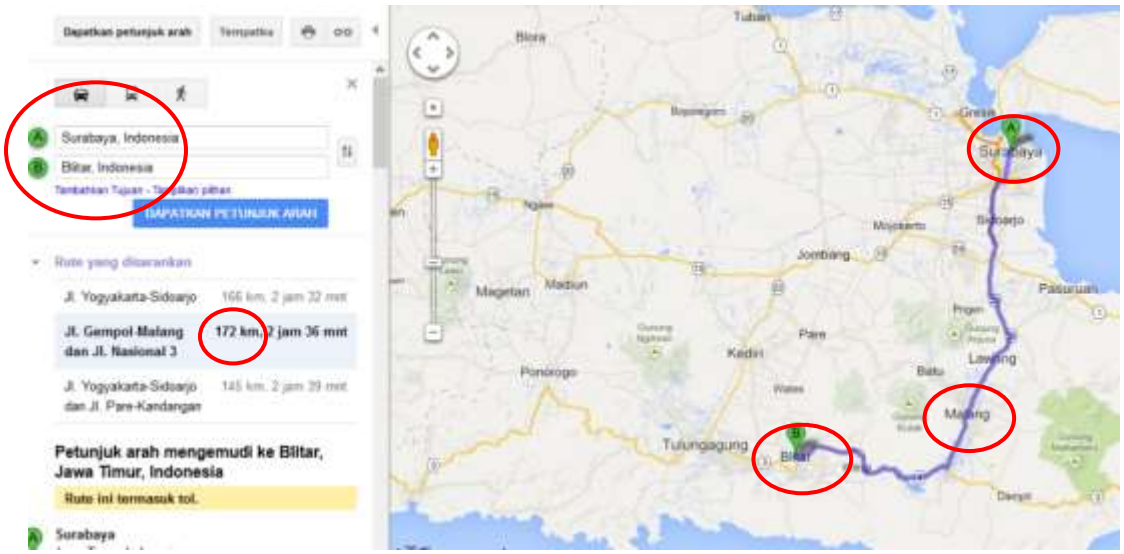

Figure 12. the result of Google Maps, from Surabaya to Blitar

Figure 13 show the result of Google Maps, from Malang city to Rembang city. The distance using Google Maps is 286 km and the distance using the proposed method is 291.7 km. This is greater 5,7 km than Google Maps. The path using the proposed method is same with Google Maps : Malang->Surabaya->Gresik->Tuban->Rembang.

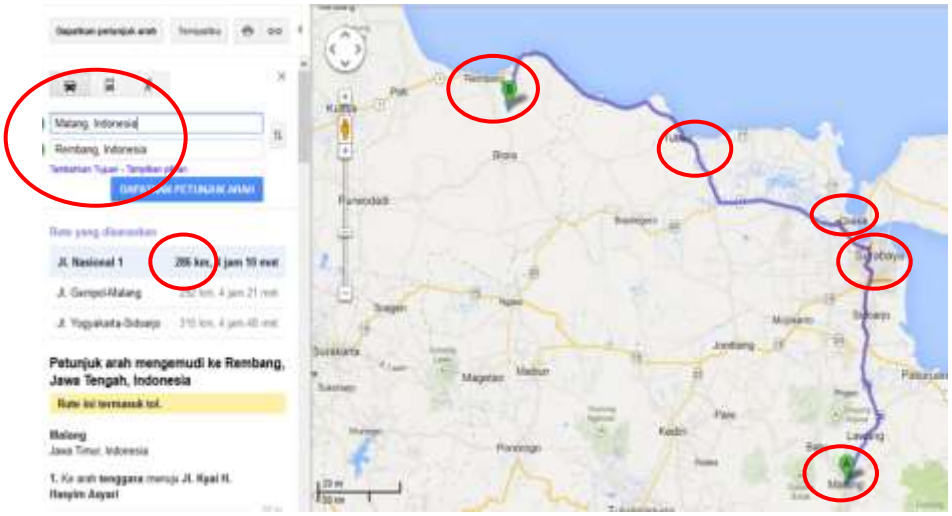

Figure13. the result of Google Maps, from Malang to Rembang

Figure 14 show the result of Google Maps, from Surabaya city to Kediri city. The distance using Google Maps is 124 km and the distance using the proposed method is 127.4 km. The path using the proposed method is same with Google Maps : Surabaya->Mojokerto->Jombang->Kediri.

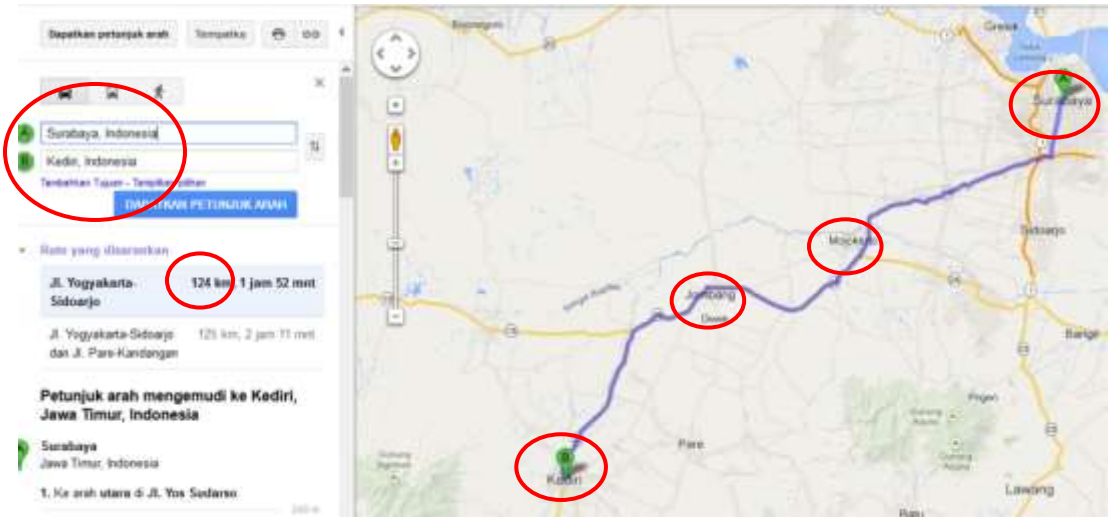

Figure 14. the result of Google Maps, from Surabaya to Kediri

Figure 15 show the result of Google Maps, from Banyuwangi city to Tuban city. The distance using Google Maps is 390 km and the distance using the proposed method is 433.5 km. The path using the proposed method is same with Google Maps : Banyuwangi->Probolinggo->Pasuruan->Surabaya->Gresik->Tuban.

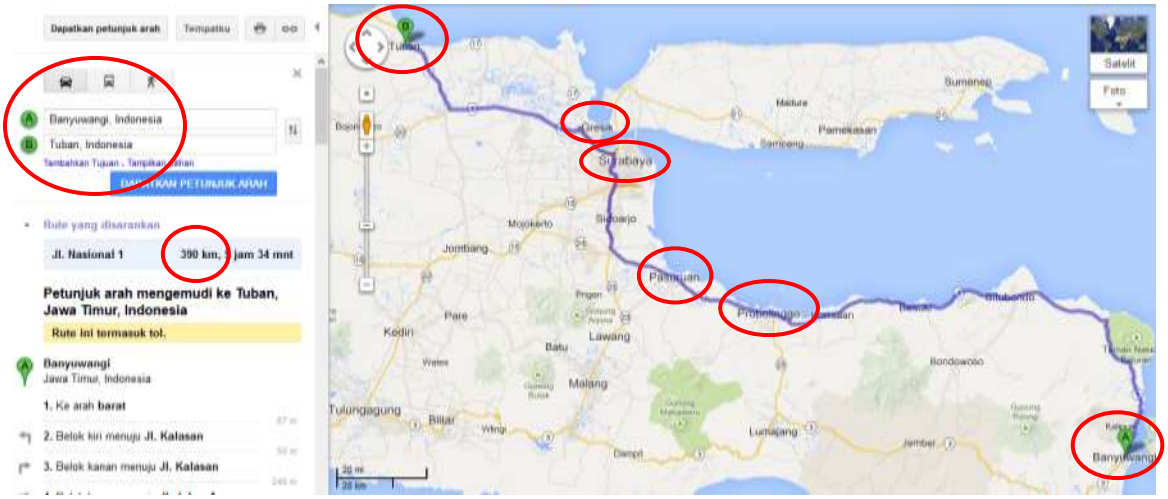

Figure 15. the result of Google Maps, from Banyuwangi to Tuban

The experiments use 5 cities from Malang as the start node (Table 3. Number 1-5), 5 cities from Banyuwangi as the start node (Table 3. Number 51-55) and 45 cities from Surabaya as the start node (Table 3. Number 6-50). There are 55 path distance result shown at Tabel 3. The proposed method can show the total distance and the cities which have been visited.

Table 3 presents the distance result between the node combination based on Dijkstra algorithm and Google Maps.

It can be evaluated from Table 3 that there are 3 kinds of results:

- 47 paths,the result of the proposed method is bigger than Google Maps.
- 4 paths (Surabaya to Blitar, Surabaya to Wonosari, Surabaya to Temanggung and Surabaya to Rembang; mark with box), the result of the proposed method is smaller than Google Maps.
- 4 paths (Surabaya to Malang, Surabaya to Mojokerto, Surabaya to Pasuruan and Surabaya to Gresik; mark with ellips), result of the proposed method is the same as Google Maps.

Table 3 shows that the accuracy of the program is 92.88%. Different results between the proposed method and Google Maps occur because the proposed method uses direct distance between two cities and Google Maps uses alternative road between two cities.

The different results of the proposed method and Google Maps are shown by an example of the distance from Surabaya to Jombang. The path result of theproposed method starts from Surabaya through Mojokerto and finish at Jombang. The proposed method will calculate the distance from Surabaya to Mojokerto and from Mojokerto to Jombang. The distance from Surabaya to Mojokerto is 49.5 km. Figure 16 and 17 shows the path of Surabaya to Mojokerto that ends at Gajah Mada Street, not passing Raya By Pass Road. The distance from Mojokerto to Jombang is 25.9 km. Figure 18 and 19 shows the path of Mojokerto to Jombang that starts from Gajah Mada Street, not Raya By Pass Road. In the light of this, the total distance from Surabaya to Jombang by using the proposed method is 49.5 + 25.9 = 75.4 km. Meanwhile, the distance from Surabaya to Jombang by using Google Mapsis 74.2 km. Figure 20 and 21 shows the path of Surabaya to Jombang through Raya By Pass Road.



Figure 16. Path of Surabaya to Mojokerto

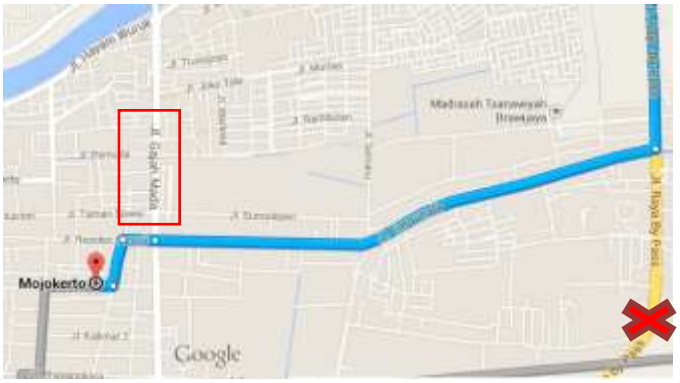

Figure 17. Zoom of box at figure16

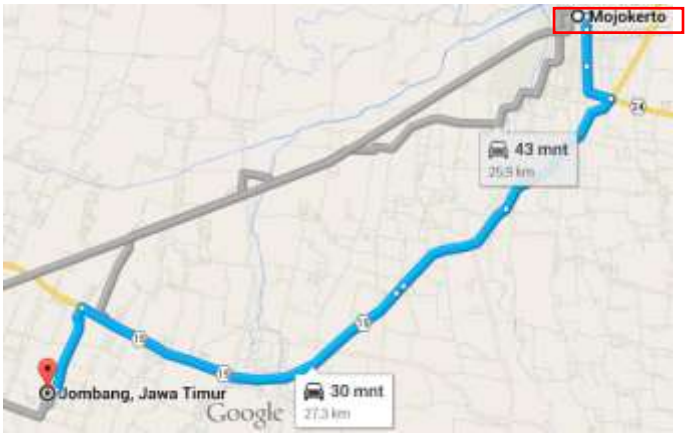

Figure 18. Path of Mojokerto to Jombang

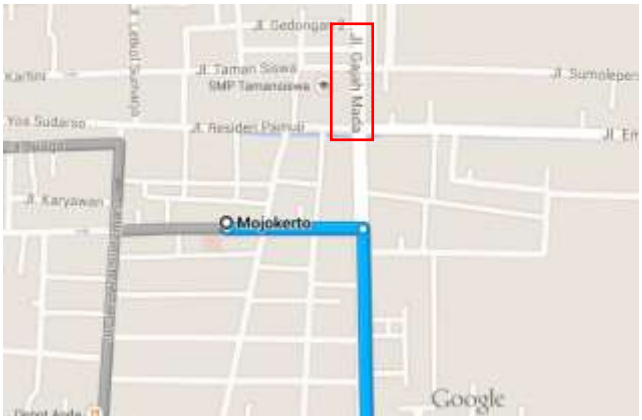

Figure 19. Zoom of boxat figure 18

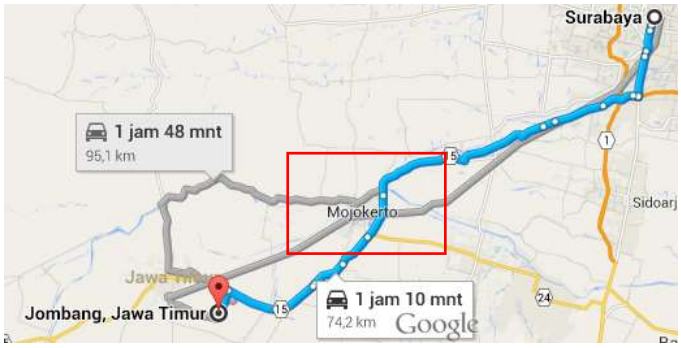

Figure 20. Path of Surabaya to Jombang

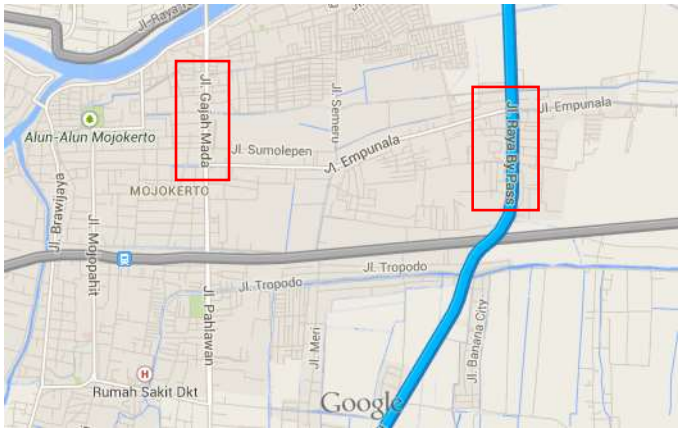

Figure 21. Zoom of box at figure 20

Table 3. The Distance Results between Node Combination Based on Dijkstra Algorithm Method and Google Maps

No	Path	Google Maps (km)	Node Combination based Dijkstra algorithm (km)	Accuracy
1	Malang->Surabaya->Gresik->Tuban->Rembang	286	291.7	98.007%
2	Malang->Kediri->Madiun->Ngawi->Sragen->Surakarta->Yogyakarta->Purworejo->Purwokerto->Tasikmalaya	661	689.3	95.719%
3	Malang->Surabaya->Gresik->Tuban->Rembang->Kudus	344	371.4	92.035%
4	Malang->Kediri->Madiun->Ngawi->Sragen->Surakarta->Yogyakarta	363	365.7	99.256%
5	Malang->Probolinggo->Banyuwangi	282	283.6	99.433%
6	**Surabaya->Malang->Blitar**	**172**	**168.1**	**97.733%**
7	Surabaya->Mojokerto->Jombang->Kediri	124	127.4	97.258%
8	Surabaya->Mojokerto->Jombang->Madiun	172	179.4	95.698%
9	**Surabaya->Malang**	**92.8**	**92.8**	**100.000%**
10	**Surabaya->Mojokerto**	**49.5**	**49.5**	**100.000%**
11	Surabaya->Mojokerto->Jombang	74.2	75.4	98.383%
12	**Surabaya->Pasuruan**	**82.8**	**82.8**	**100.000%**
13	Surabaya->Pasuruan->Probolinggo	102	131.8	70.784%
14	**Surabaya->Gresik**	**21.2**	**21.2**	**100.000%**
15	Surabaya->Pasuruan->Probolinggo->Banyuwangi	293	324.8	89.147%
16	Surabaya->Gresik->Tuban	103	108.7	94.466%
17	Surabaya->Mojokerto->Jombang->Madiun->Magetan	190	201.4	94.000%
18	Surabaya->Mojokerto->Jombang->Madiun->Ngawi	182	209.8	84.725%
19	Surabaya->Mojokerto->Jombang->Madiun->Ponorogo	198	209.5	94.192%
20	Surabaya->Mojokerto->Jombang->Madiun->Ponorogo->Pacitan	276	287.5	95.833%
21	Surabaya->Pasuruan->Probolinggo->Jember	200	231.8	84.100%
22	Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen->Surakarta->Magelang	352	398.2	86.875%
23	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan	410	436.2	93.610%
24	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang	312	334.2	92.885%
25	Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen->Surakarta	264	293.2	88.939%
26	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal	473	499.8	94.334%
27	Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen->Surakarta->Yogyakarta	328	358	90.854%
28	**Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen->Surakarta->Wonosari**	**346**	**323.9**	**93.613%**
29	Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen	234	262.6	87.778%
30	Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen->Surakarta->Yogyakarta->Purworejo	394	413.6	95.025%
31	Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen->Surakarta->Yogyakarta->Purworejo->Purwokerto	490	527.6	92.327%

32	Surabaya->Mojokerto->Jombang->Madiun->Ngawi->Sragen->Surakarta->Yogyakarta->Purworejo->Purwokerto->Cilacap	504	584.5	84.028%
33	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Temanggung->Wonosobo	428	449	95.093%
34	**Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Temanggung**	**428**	**406.1**	**94.883%**
35	**Surabaya->Gresik->Tuban->Rembang**	**200**	**198.9**	**99.450%**
36	Surabaya->Gresik->Tuban->Rembang->Kudus	258	278.6	92.016%
37	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon	545	571.7	95.101%
38	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Tasikmalaya	662	675.7	97.931%
39	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang	673	700.7	95.884%
40	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bandung	682	785.8	84.780%
41	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Purwakarta	718	758.1	94.415%
42	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bandung->Cianjur	741	849.7	85.331%
43	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bandung->Cianjur->Sukabumi	771	879.9	85.875%
44	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi->Bogor	816	862.2	94.338%
45	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi	764	805.7	94.542%
46	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi->Jakarta	783	826.2	94.483%
47	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi->Jakarta->Tangerang	812	859.2	94.187%
48	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi->Jakarta->Tangerang->Serang	865	925.7	92.983%
49	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi->Jakarta->Tangerang->Serang->Cilegon	885	946.9	93.006%
50	Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi->Jakarta->Tangerang->Serang->Cilegon->Merak	891	956.2	92.682%
51	Banyuwangi->Probolinggo->Pasuruan->Surabaya	294	324.8	89.524%
52	Banyuwangi->Probolinggo->Pasuruan->Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal	760	824.6	91.500%
53	Banyuwangi->Probolinggo->Pasuruan->Surabaya->Gresik	309	346	88.026%
54	Banyuwangi->Probolinggo->Pasuruan->Surabaya->Gresik->Tuban	390	433.5	88.846%
55	Banyuwangi->Probolinggo->Pasuruan->Surabaya->Gresik->Tuban->Rembang->Kudus->Semarang->Pekalongan->Tegal->Cirebon->Subang->Bekasi->Jakarta->Tangerang	1100	1184	92.364%
			Average	**92.88%**

V. CONCLUSION

The shortest path between cities in Java can be found by node combination based on dijkstra algorith with accuracy 92.88%. There are different results between the proposed method and Google Maps because the proposed method applies direct distance between two cities and Google Maps uses alternative road between two cities. The proposed method can show which cities that are visited. There are 4 paths result of the proposed method is smaller than Google Maps. There are 4 paths result of the proposed method is same as Google Maps.

REFERENCES

[1]. X. Zhang, Y. Zhang, "Rapid Physarum, Algorithm ForShortest Path Problem", Applied Soft Computing Vol. 23,pp.19–26, Oktober2014.

[2]. M. Farhanchi, R. Hassanzadeh, "A Modified Ant Colony System For Finding The Expected Shortest Path In Networks With Variable Arc Lengths And Probabilistic Nodes", Applied Soft Computing Vol. 21,pp. 491–500, August 2014.

[3]. F. W. Trevizan, M. M. Veloso, "Depth-Based Short-Sighted Stochastic Shortest Path Problems", Artificial Intelligence, 216,pp. 179–205, 2014.

[4]. J. DeCarufel, C. Grimm, "A Note On The Unsolvability Of The Weighted Region Shortest Path Problem", Computational Geometry, Vol. 47, issue 7, pp. 724–727, August 2014.

[5]. A. K. Ziliaskopoulos, F. D. Mandanas, "An Extension Of Labeling Techniques For Finding Shortest Path Trees", European Journal of Operational Research, Vol. 198, issue 1,pp.63–72, October 2009.

[6]. C. Gao, C. Yan, "An Amoeboid Algorithm For Solving Linear Transportation Problem", Physica A: Statistical Mechanics and its Applications, Vol. 398, pp. 179–186, March 2014.

[7]. E. Amaldi, A. Capone, "Energy-Aware IP Traffic Engineering With Shortest Path Routing", Computer Networks, Vol. 57, issue 6, pp. 1503–1517, April 2013.

[8]. Y. Zhang, Z. Zhang, "A Biologically Inspired Solution For Fuzzy Shortest Path Problems", Applied Soft Computing,Vol. 13, Issue 5, pp. 2356–2363, May 2013.

[9]. A. Nazemi, F. Omidi, "An Efficient Dynamic Model For Solving The Shortest Path Problem", Transportation Research Part C: Emerging Technologies, Vol.26,pp.1–19, January 2013.

[10]. W. Peng, X. Hu, "A Fast Algorithm to Find All-Pairs Shortest Paths in Complex Networks", Procedia Computer Science, Vol. 9, pp. 557 – 566, 2012.

[11]. R. Hassanzadeh, I. Mahdavi, "A Genetic Algorithm For Solving Fuzzy Shortest Path Problems With Mixed Fuzzy Arc Lengths", Mathematical and Computer Modelling, Vol. 57, Issue 1-2, pp. 84 – 99, January 2013.

[12]. W. Shu-Xi ,"The Improved Dijkstra's Shortest Path Algorithm and Its Application", Procedia Engineering, Vol. 29, pp. 1186 – 1190, 2012.

[13]. S. Subramanian, R. Tamassia, "An Efficient Parallel Algorithm for Shortest Paths inPlanar Layered Digraphs", Algorithmica, Vol. 14, Issue 4, pp. 322-339, October 1995.

[14]. J. Joaci, A. R. Backes, "Texture Analysis And Classification Using Shortest Paths In Graphs", Pattern Recognition Letters, Vol. 34, Issue 11, pp. 1314–1319, August 2013.

[15]. J. Quan Li, "Match Bus Stops To A Digital Road Network By The Shortest Path Model", Transportation Research Part C: Emerging Technologies, Vol. 22, pp. 119–131, June 2012.

[16]. C. Bode, S. Irnich, "The Shortest-Path Problem With Resource Constraints With (K,2) – Loop Elimination And Its Application To The Capacitated Arc-Routing Problem", European Journal of Operational Research, Vol. 238, Issue 2, pp. 415–426, October 2014.

[17]. F. J. Pulido, L. Mandow, "MultiobjectiveShortest Path Problems With Lexicographic Goal-BasedPreferences", European Journal of Operational Research, Vol. 239, Issue 1, pp. 89–101, November 2014.

[18]. Y. Xi, L. Schwiebert, "Privacy Preserving Shortest Path Routing With An Applicationto Navigation", Pervasive and Mobile Computing, Vol. 13, pp. 142–149, August 2014.

[19]. C. L. Azevedo, J. L. Cardoso, "Vehicle Tracking Using The K-Shortest Paths Algorithm And DualGraphs", Transportation Research Procedia, Vol. 1, Issue 1, pp. 3 – 11, 2014.

[20]. M. Baxter, T. Elgindy, "Incremental Network Design With Shortest Paths", European Journal of Operational Research, Vol. 238, Issue 3, pp. 675–684, November 2014.

[21]. L. Sitanayah, K. N. Brown, "A Fault-Tolerant Relay Placement Algorithm For Ensuring Kvertex-Disjoint Shortest Paths In Wireless Sensor Networks", Ad Hoc Networks, Vol. 23, pp. 145–162, December 2014.

[22]. B. Doerr, D. Johannsen, "More Effective Crossover Operators For The All-Pairs Shortest Path Problem", Theoretical Computer Science, Vol. 471, pp. 12–26, February 2013.

[23]. F. Chu, S. Chen, "Optimal Design Of Pipeline Based On The Shortest Path", Physics Procedia, Vol. 33, pp. 216 – 220, 2012.

[24]. X. Lu, M. Camitz, "Finding The Shortest Paths By Node Combination",Applied Mathematics and Computation, Vol. 217,Issue 13, pp. 6401–6408, March 2011.

[25] G. Yu, H.Song, "Unmanned Aerial Vehicle Path Planning Based On TLBO Algorithm", International Journal On Smart Sensing And Intelligent Systems, Vol. 7, No. 3, pp. 1310 – 1325, September 2014.

[26] Z. Cui, Y. Zhao, "An Energy-Efficient Routing For Vehicular Ad Hoc Networks Using Real-Time Perception Of Node Information" International Journal On Smart Sensing And Intelligent Systems Vol. 8, No. 2, pp. 1142– 1161, January 2015.

[27] C. J. Zhu, K.Y Lam, "Approximate Path Searching For Supporting Shortest Path Queries On Road Networks", Information Science, Vol. 325, pp. 409-428, December 2015.

[28] F.J. Pulido, L. Mandow, "Dimensionality Reduction In Multiobjective Shortest Path Search", Computers & Operations Research, Vol. 64, pp. 60-70, December 2015.