Entropy-Based k Shortest-Path Routing for Motorcycles: A Simulated Case Study in Jakarta

Muhamad Asvial[1], M. Faridz Gita Pandoyo[2], Ajib Setyo Arifin[3]
Department of Electrical Engineering
Universitas Indonesia
Depok, Indonesia

***Abstract*—Traffic congestion is a serious problem in rapidly developing urban areas like Jakarta, Indonesia's capital city. To avoid the congestion, motorcycles assisted with navigation apps are popular solution. However, the existing navigation apps do not take into account traffic data. This paper proposes an open-source navigation app for motorcycle by taking into account the traffic data and wide road to avoid congestion. The propose navigation app uses entropy-balanced k shortest paths (EBkSP) algorithm to suggest different routes to different users to prevent further congestion. Tests show that the proposed route planning system in the app gives routes that are significantly shorter than motorcycle routes planned by Google Maps. The EBkSP algorithm also distributes vehicles more evenly among routes than the random kSP algorithm and does so in a practical amount of computing time.**

Keywords*—Traffic congestion; motorcycle; navigation apps; EBkSP

I. INTRODUCTION

The transportation sector has a great influence on development across all other sectors of an economy. In Jakarta, the capital city of Indonesia, as many as 3.5 million people used ground transportation infrastructure to travel to work or school in 2014. Private vehicles are still the favoured means of transportation because Jakarta's public transit architecture is not fully developed. Jakarta has 3.3 million cars and 13 million motorcycles, and these numbers grow by approximately 8%–10% annually [1]. With road capacity increasing by less than 1% each year, the number of vehicles will eventually exceed the available road capacity. In 2014, the ratio of road surface to the number of cars/motorcycles in Jakarta was 3:1, meaning that for every 3 m^2 of road, at least one car or motorcycle was in the city [2].

Motorcycles are very popular in Jakarta for their relatively low price and their manoeuvrability on narrow roads. Motorcyclists can avoid traffic congestion by taking shortcuts that are inaccessible by car. However, many motorcyclists do not have access to a navigation system that shows such shortcuts. With the number of motorcycles in Jakarta almost four times the number of cars, open-source navigation apps intended specifically for motorcyclists are in high demand. Moreover, such an app has the potential to reduce congestion for all vehicles in urban areas by distributing motorcycle traffic away from wider roads.

Navigation apps collect traffic and map data and plan the shortest route between two points using some algorithm. The entropy-balanced k shortest paths (EBkSP) routing algorithm is promising for this purpose and has been shown to be effective in urban environments [3]. The existing navigation apps, such as: Bing Maps and Apple Maps, do not have navigation for motorcycle. There is only Google Maps that provides the navigation for motorcycle. This paper compares performance of proposed apps with the routing with Google Maps. This paper presents a case study of the effectiveness of this algorithm in a motorcyclist-specific navigation system that is simulated using a primary data. The route-planning tests were conducted by a field trial of origin-destination (OD) pairs in 6 (six) sub-districts in Jakarta, Indonesia.

The navigation system presented below uses open-source data and libraries for route planning, and the case study shows that the EBkSP algorithm effectively distributes vehicles among the shortest routes between two points to reduce the navigation app's chances of making congestion worse. The results suggest that a motorcyclist-specific navigation app would be useful in cities like Jakarta, and that EBkSP is a promising strategy for navigation apps that do not concentrate users on suggested routes.

The rest of this paper is organized as follows. Section 2 reviews the literature on route planning to introduce the available algorithms and clarify how the constraints presented by Jakarta's environment indicate demand for a novel open-source navigation system. Section 3 formulates the EBkSP algorithm and outlines the software architecture we used to test it. Section 4 presents the data and map used for route-planning tests. Section 5 presents the test results and compares the performance of the motorcycle-specific EBkSP route planner against similar planners. The route planners are compared in terms of the lengths of routes they suggest, the distribution of vehicles on those routes, and the computation time needed for route planning. Section 6 briefly draws conclusions that highlight the promise of an open-source entropy-based navigation system that considers paths specifically for motorcyclists.

II. Related Works

The literature on navigation apps includes a variety of approaches for collecting traffic data and detecting congestion. Global Positioning System (GPS) location services can be used along with real-time traffic data, such as in the MobiWay app, which also implements Long Short-Term Memory networks. Vehicular ad hoc networks (VANETs) can also be used to quantify road traffic congestion in a distributed manner in an environment with sufficient penetration of connected vehicles. Still others have used data from GPS location services and VANETs simultaneously [4-9]. For an app that can be used in Jakarta, traffic congestion is best detected from vehicle speeds using GPS location data collected from user smartphones. This speed data can then be processed using Kalman Filtering to detect traffic congestion [10].

Route planning is then needed to direct users around traffic congestion. Google Maps and Waze are popular smartphone apps that give users the shortest route to a destination along with alternate routes and estimated travel times. These applications are very popular, and that popularity may actually exacerbate traffic congestion. If a very popular navigation app suggests the same route around a traffic jam to many users at once, that route will then become congested itself. Dynamic Traffic Assignment (DTA) takes account of temporal factors in route planning to ensure that route planning does not incidentally increase traffic congestion [11]. Vehicle to infrastructure (V2I) technology that relies on roadside units can be used to achieve DTA, such as in Congestion Avoidance through Traffic Classification Mechanism and Rerouting Algorithm (CHIMERA) [12], Next Road Rerouting (NRR) [13], and System with Cooperative Routing to Improve Traffic Condition (SCORPION) [14]. However, V2I technology cannot be applied in Jakarta because of its lacking road infrastructure and the lack of vehicles that can communicate with that infrastructure.

Studies have shown that EBkSP performs better than the dynamic shortest paths and random k shortest paths (RkSP) algorithms [15]. EBkSP looks for the shortest k routes and then recommends the most unpopular route to a user, and the popularity of each route is updated with each route-planning request. Simulations of traffic congestion have been shown to be useful in evaluating route-planning algorithms' ability to reduce congestion [16]. Navigation systems that collect data from road side units (RSUs) have also been tested in simulations, but these systems also cannot be implemented in Jakarta due to lacking infrastructure [17] [18] [19]. One study found that navigation systems that do not require RSUs still can overcome traffic congestion [20].

All of the navigation systems mentioned above are in the stage of simulation testing, so they are obviously not available to users. Other researchers have created a navigation app that uses Dijkstra's algorithm for route planning and takes advantage of data from RSUs [21]. The application was not designed for DTA, and cannot be applied with Jakarta's infrastructure. Navigation systems have been designed and tested using the open-source Simulation of Urban Mobility (SUMO) connected to the Google Maps application programming interface (API) for map and traffic data [22]. Another open-source option is to import maps from OpenStreetMap into PostgreSQL databases and use the pgRouting library for route planning [23]. We follow the latter strategy for testing the EBkSP algorithm's usefulness for motorcycle-specific routes. As we will explain in following sections, the algorithm and trial reported in this paper is the continuous development on integrated solution device for motorcycle [24].

III. The Algorithm

A. Algorithm Design

To plan routes specifically for motorcyclists, available map data needs to be reconfigured such that the route planner prioritises routes that are too narrow for cars to pass. We modify the road classes included in OpenStreetMap for this purpose. The map data is configured for motorcycles by modifying the configuration for bicycles and adding some wider roads routes that cars can use, though toll roads are still excluded. The resulting road classes in the map configuration for motorcyclists are as follows, sorted in order of increasing priority for route planning:

- Track (unpaved surface)
- Service (small shortcut to some roads)
- Living street (within residential areas, speeds are kept low)
- Residential (access to housing)
- Tertiary (link smaller towns and villages)
- Secondary (link towns)
- Primary (link larger towns)
- Trunk (divided highway)

The road classes above are written to an xml file that can be used by the osm2pgrouting library. The route planner will seek roads of higher priority first when plannign a route between two points.

For testing of the EBkSP route planner, we modify the k shortest paths (kSP) routing algorithm included in pgRouting to use entropy in the routing calculations. The kSP routing protocol searches for k alternate routes from the current point to a destination, which in this simulation are limited to three. The number of route options is limited to three for left, right, and continuing straight; if turning around is included the algorithm can get stuck in a loop.

The EBkSP routing algorithm the popularity of a route to avoid congestion on the route to be traversed. More popular routes are more likely to be congested congestion as vehicles clog the route. Algorithm 1 describes EBkSP in steps. The algorithm begins with the input of the user's origin and destination points. After receiving this information, the server calculates alternative routes based upon weighted footprint data. The weighted footprint data indicates the popularity of each route, and congestion can be prevented by directing users to unpopular routes that are of equal length to the shortest routes available. In practice, the weighted traffic footprint data

would be collected from users that use the navigation app. For the sake of the present simulation tests, this footprint data was prepared manually.

Algorithm 1. EBkSP algorithm

Input:
1: Get device's origin, destination
2: Collect all k paths
3: Collect weighted traffic data

Output:
4: Appropriate route

Procedure:
5: Analyse weighted traffic data
6: Choose shortest route
7: **for** not updated weighted traffic data
8: **if** route chosen is least popular **then**
9: Choose route
10: Update weighted footprint data
11: **end if**
12: **else if** all routes equally popular **then**
13: Choose route
14: Update weighted footprint data
15: **end if**
16: **else if** eliminate shortest route **then**
17: Choose next-shortest route
18: **end if**
19: **end**

Equation (1) gives the popularity of route j as an exponential function of the entropy of route j. $Pop(p_j)$ takes the value of 0 if no car has taken j. The entropy of each route is calculated using equation (2), which sums the natural logarithms of the inverse of the number of vehicles heading in a given direction from some starting point, N, for each vehicle that has passed through route j, n_j.

$$Pop(p_j) = \begin{cases} e^{E(p_j)}, n_j \neq 0 \\ 0 \quad\quad , n_j = 0 \end{cases} \quad (1)$$

$$E(p_j) = -\sum_{t=1}^{n_j} \frac{1}{N} ln\ \frac{1}{N} \quad (2)$$

$$N = \sum_{t=1}^{n} n_j \quad (3)$$

If the routes are equally popular, the shortest route is chosen. If the routes have different levels of popularity, the server will choose the least popular route even if it is the longest. Once the route is selected, the user is assumed to follow the route, and the server will update the weighted footprint data. If we imagine a sequence of users requesting routes from the algorithm, the first user will be assigned the shortest route because all routes are equally popular before any user has taken them. The next user will be assigned the next-shortest route because the first user made the shortest route more popular than the rest. Through this process, the algorithm distributes vehicles evenly among all possible routes between two points if all vehicles follow the routes it suggests. We can record the average distance of the paths taken between two points in a traffic simulation to assess the effectiveness of this route-planning strategy.

B. Software Architecture

The present research is intended toward the design of a smartphone navigation app that motorcyclists can use in a city like Jakarta. The overall software architecture can use a centralized server for processing. The server has access to map and traffic data and presents a map to the user on his or her smartphone. The smartphone app then requests a route from the server based on user input. The server calculates the popularity of all routes and executes the algorithm, and then the results are displayed on the user's device as a route on the map.

C. Frontend and Backend Design

Frontend software interacts directly with the user. The variables required by the backend section are obtained from the frontend section, and the frontend also displays the output route in an image form that can be understood by the user. In a practical implementation of this navigation app, users would need to use a web application that can be accessed through a computer or a Hypertext Markup Language 5 (HTML5)-based application on a smartphone.

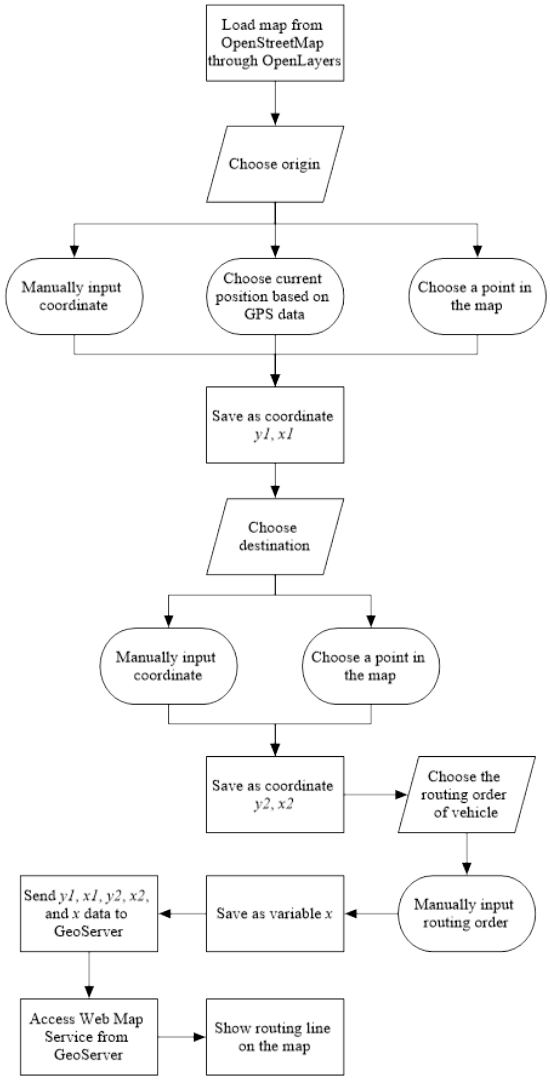

Fig. 1. Frontend Software Diagram.

Fig. 1 shows how the map data from OpenStreetMap is accessed through OpenLayers so that the data used with the web map service (WMS) protocol in GeoServer. The app needs three inputs. First, the user-selectable point of origin must be entered, either by typing in coordinates, drawing data from the smartphone GPS unit, or by selecting a point on the map. The origin latitude and longitude are stored as y1 and x1, respectively. The destination point must also be input, and its coordinates are stored as y2 and x2. If the system is to accomplish DTA, the system also needs data about how many vehicles have taken each route. This data is entered manually for the sake of simulation and is stored as variable x. This data is then passed to the GeoServer for computation. Results obtained from the GeoServer in the form of route lines are accessed from the WMS and then combined with OpenStreetMap via OpenLayers so that the route is presented to the user in an easily understandable form.

The backend software handles all computation and passes all routes back to the frontend software using the WMS protocol. Fig. 2 schematizes how the instance of GeoServer accepts the variables y1, x1, y2, x2, and x sent by the frontend and then accesses the pgRouting database. The GeoServer then runs the EBkSP algorithm and sends the variables y1, x1, y2, x2, and x to it for computation. The results from pgRouting are formatted as a list of road segments that link the origin with the destination, which are then reformatted to work with WMS.

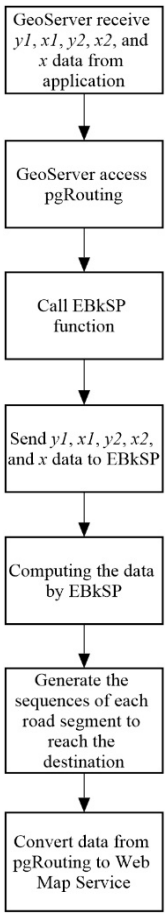

Fig. 2. Backend Software Flowchart.

IV. Test Data

A. Data Collection

We use OpenStreetMap data of Jakarta for testing, which includes vertices (intersections), lengths between vertices, road identifications, and road classes. These are included in a PostgreSQL database using the osm2pgrouting library, implementing the novel modifications to the map configuration that we described above. Fig. 3 schematizes the sequence for data collection. We chose OpenStreetMap data because it is uploaded by local contributors who know their streets well.

For testing, we entered data about the number of vehicles on each road manually. Then the results are not affected by traffic patterns like rush hour, but the tests are also not concerned with travel time, which is the only characteristic of a route that is affected by the higher density of vehicles during rush hour. The simulations also assume that all users follow the routes planned by the app.

B. Route Planning and Map

We chose a select set of origin-destination (OD) pairs for testing the route-planning algorithm, as shown in Table I. Six different OD pairs were chosen to compare routing for motorcycles with routing that includes only car-passable roads. The origin for all pairs was determined as a point on the campus of Universitas Indonesia (UI), while there were six destination points : in the “Cinere” area, in the “Lebak Bulus” area, in the “Pasar Minggu” area, in the “Ragunan” area, in the “Kebagusan” area, and in the “Jagakarsa” area. Those area are main sub-districts in the southern part of Jakarta.

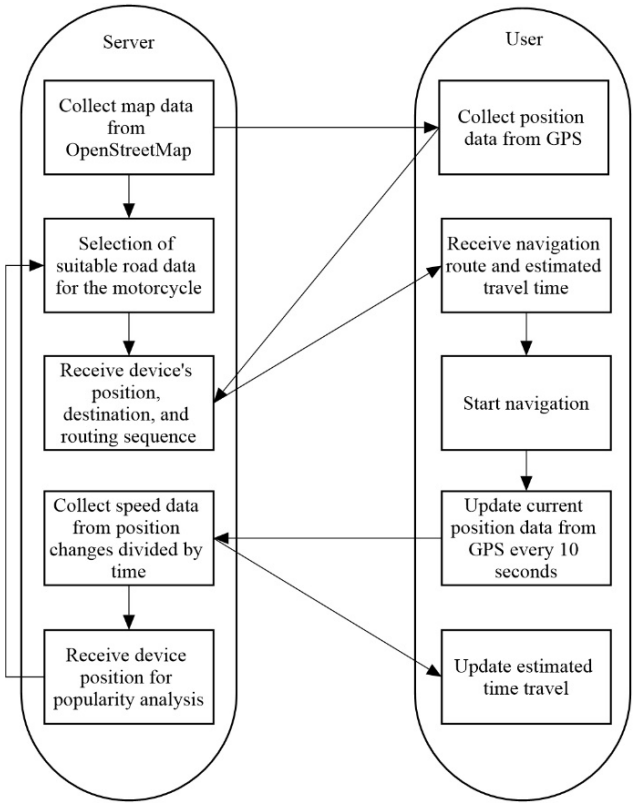

Fig. 3. Data Collection Sequences.

TABLE I. THE 6 ORIGIN-DESTINATION (OD) PAIRS FOR TESTING THE ALGORITHM

No.	OD Pairs	Coordinates	
		Origin	Destination
1	UI to Cinere	-6.36114, 106.82359	-6.33327, 106.78288
2	UI to Lebak Bulus		-6.29848, 106.77436
3	UI to Pasar Minggu		-6.28531, 106.84403
4	UI to Ragunan		-6.31071, 106.81524
5	UI to Kebagusan		-6.31132, 106.82581
6	UI to Jagakarsa		-6.32710, 106.81378

Fig. 4 shows a map of the area covered in the tests. The starting point for all the OD pairs on the campus of Universitas Indonesia (UI) is marked (coordinates -6.36114, 106.82359). Point 1 is in "Cinere" area, point 2 is in "Lebak Bulus" area, point 3 is in "Pasar Minggu" area, point 4 is in "Ragunan" area, point 5 is in "Kebagusan" area, and point 6 is in "Jagakarsa" area.

The shortest path from UI to Cinere and UI to Jagakarsa takes a narrow road (2–4 m wide); the only wider route between these two locations is much longer. For the OD pairs UI–Lebak Bulus, UI–Ragunan, and UI–Kebagusan, the shortest route uses narrow roads and a car-accessible route is available that is only slightly longer. The shortest path between UI and Pasar Minggu uses roads that are wide enough for a car to use.

Fig. 4. The Map for Testing the Algorithm: 1 (One) Starting Point on the Campus of Universitas Indonesia (UI) and 6 (Six) Destination Points.

V. RESULTS AND ANALYSIS

The performance indicators of simulated test focus on the aspect of lengths of calculated routes, distributions of vehicles on each route and computing time. Results are presented in Fig. 5, Fig. 6, and Fig. 7. Tests were also run to compare the developed EBkSP algorithm against Google Maps for motorcycles and the same application architecture with the RkSP algorithm. We also tested performance for maps configured for cars and motorcycles.

For each OD pair, three alternate routes of roughly the same distance were planned for each of the routing systems we tested. Obtaining results for each of these methods and routing configurations required performing the routing process an average of 15 times. For Google Maps, the routing process was done only once for each OD pair because the output path was always the same. The simulations assume that vehicles continue on the suggested route once they begin moving toward the destination. The backend system updates the traffic footprint data every time a vehicle makes a request for routing to a destination from the same departure point.

A. Lengths of Calculated Routes

Fig. 5 shows that the EBkSP algorithm for motorcycles returns routes of almost identical length to those planned with the RkSP algorithm for motorcycles. This result is expected because the EBkSP motorcycle configuration and RkSP motorcycle configuration use the same map configuration, and the available routes are prioritised similarly when the differences in distance between the shortest and longest routes are not very much. The routes planned by the EBkSP motorcycle configuration are shorter than those planned by the Google Maps motorcycle configuration for the UI–Cinere, UI–Ragunan, UI–Kebagusan, and UI–Jagakarsa journeys, which means that EBkSP has an advantage when narrow streets allow a shorter path to the destination point, when wide roads are not available, and when the destination point closer than 10 km from the origin.

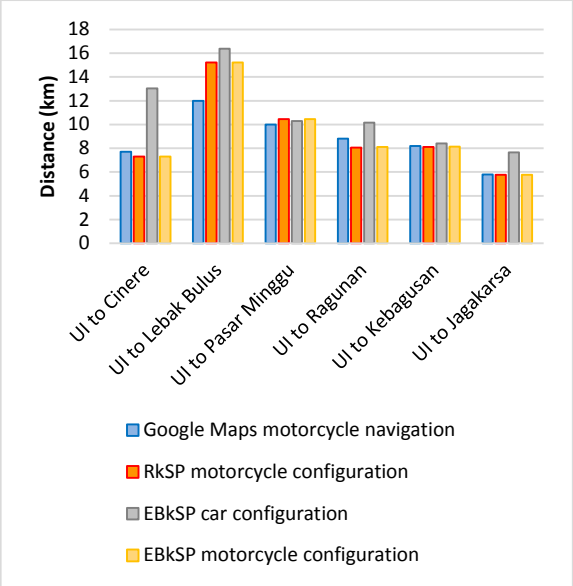

Fig. 5. Average Distances of Routes Planned by each Navigation System.

If the destination point is more than 10 km away, then the EBkSP motorcycle configuration performs no better Google Maps motorcycle configuration, as one can see in the results for UI–Lebak Bulus. Narrow roads near the campus offer a shorter path to Lebak Bulus from UI near the campus, but that path is longer than 10 km. The EBkSP motorcycle configuration plans routes that begin with a nearby wider road before taking narrow roads, while routes planned by Google Maps take the narrow roads right away for a shorter route overall. This difference in performance is explained in part by differences in some of the road data used by Google Maps and OpenStreetMap and in part by Google Maps' tendency to avoid alleyways if the difference in distance is sufficiently small.

All the configurations calculate about the same routes between UI and Pasar Minggu on a highway, and the differences in distance are explained by the algorithms' differences in calculating route distance.

These results suggest that route planning with map data configured specifically for motorcycles yields shorter routes. The EBkSP motorcycle configuration gives shorter routes than the EBkSP car configuration for all journeys except UI–Pasar Minggu, on which a wide road connects the origin and destination directly.

B. Distributions of Vehicles on each Route

Standard deviations can be used to express how evenly the route-planning algorithms distribute vehicles among the possible routes between two points. If the standard deviation is lower, the vehicles are more evenly distributed, so congestion will be less likely to arise. The average standard deviation of the number of vehicles on each route from the mean is plotted for each routing algorithm in Fig. 6. The RkSP motorcycle configuration routes have standard deviations of 1.73 vehicles per route on OD UI–Cinere, 2.65 vehicles per route on OD UI–Lebak Bulus, 2.65 vehicles per route on OD UI–Pasar Minggu, 3.61 vehicles per route on OD UI–Ragunan, 2.65 vehicles per route on OD UI–Kebagusan, and 4.36 vehicles per route on OD UI–Jagakarsa. The RkSP algorithm assigns routes of the same length to users randomly, so vehicles taking those routes can pile up on one route and increase the potential for congestion. The plot in Fig. 6 includes no bars for the EBkSP routes because the standard deviation of vehicles on each route was zero; i.e., the EBkSP algorithm distributes vehicles evenly among routes.

These results show that the EBkSP algorithm is effective for DTA using only the history of routes suggested by the navigation system, under the assumptions we have applied to the above simulations. The algorithm calculates the popularity of each route from the number of vehicles that have taken that route to and preferences unpopular routes, updating this entropy calculation with each route it suggests. By definition, this algorithm will spread users of a navigation system evenly among alternate routes to get around traffic congestion. This even distribution of vehicles among routes suggests that an open-source navigation app using EBkSP can effectively avoid the accumulation of vehicles on one route, so that such an app would be unlikely to exacerbate congestion.

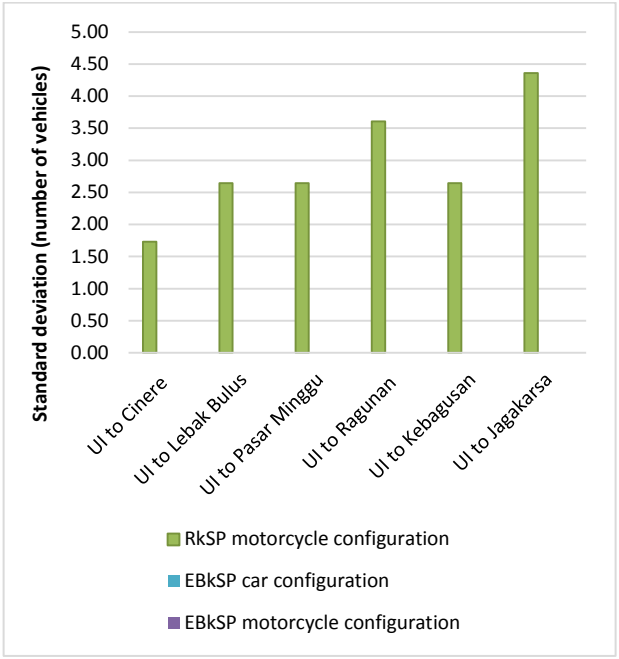

Fig. 6. Standard Deviations for Vehicle Distribution on each Route. Data for EBkSP Algorithms is not Visible because all Values are Zero.

C. ComputingTime

Since the EBkSP algorithm is more complicated than the RkSP algorithm, it will take longer to compute, so we need to check that the computing time is not so long as to make the EBkSP algorithm impractical. The EBkSP algorithm is applied to the PostgreSQL database of map and traffic data in three stages: route sorting based on length, calculating the popularity of each route, and choosing the best route while minimising length and popularity. This computation takes more time than RkSP, which simply assigns shortest routes randomly. Fig. 7 shows that the computation time for EBkSP considering motorcycle-accessible routes is the longest, as expected, but the EBkSP algorithm still returns routes faster than the Google Maps API. The EBkSP car configuration happens to compute faster than the RkSP motorcycle configuration for the origination-destination pairs considered in the present tests.

The relatively long computation time of the Google Maps motorcycle configuration is explained by the detailed graphics of the route lines created and the calculation of the average speed of each different road segment, which is not considered by the other algorithms we tested. Motorcycle routes take longer to compute in all cases because more roads are considered as possible; the algorithm considers only wider roads when calculating routes for cars.

D. Discussion

This paper propose the navigation app for motorcycle using EBkSP algorithm. Performance of the app is measured using three parameters: length route, vehicle distribution, and computing time. Testing is conducted by comparing measurement result with the data from Google Maps. The app is superior for determining shortest path when taking into account the narrow streets. Moreover, taking popularity parameter of the routes can avoid the users pass the same route

and distribute into different routes. The last parameter is computing time. Taking into account the narrow streets means that the app requires more time for making decision. However, the proposed app still outperforms the Google Maps in term of computing time.

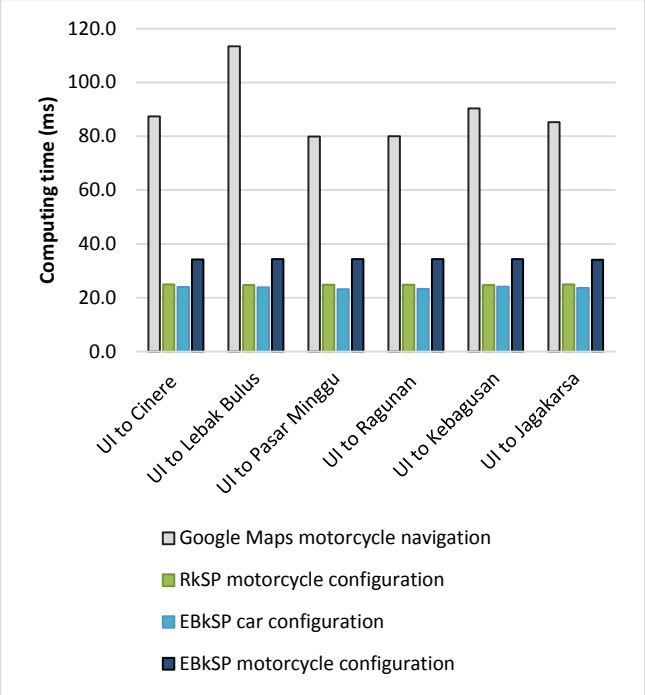

Fig. 7. Average Computing Time for Planning Routes between each Origin-Destination Pair.

VI. Conclusion

The simulation tests suggest that the proposed routing algorithm is advantageous when the destination is within 10 km and can be reached via a shorter route using narrow streets. Motorcyclists can use the system to reach their destinations up to 8% faster than they can with Google Maps for motorcycle riders and up to 44% faster than the EBkSP algorithm for cars, if they are travelling between areas of Jakarta linked by narrow streets. EBkSP accomplishes Dynamic Traffic Assignment (DTA) in that it prevents congestion by directing vehicles to less-popular routes. Computing time is affected more by the configuration of the routing algorithm for car or motorcycles than the particular algorithm used. These results show that an open-source route-planning algorithm for motorcyclist use can offer improved performance over algorithms that consider only routes that are navigable by cars. The EBkSP algorithm shows clear promise for use in a navigation system that achieves DTA without the need for ubiquitous traffic-data infrastructure. In the future research, taking into account as many as possible of traffic attributes such as number of traffic light and cross section, can improve performance of routing.

Acknowledgment

This work was supported by PITTA Universitas Indonesia Grant 2018 with contract number: 2368/UN2.R3.1/ HKP.05.00/2018.

References

[1] Sub-Directorate of Transportation Statistics, Land Transportation Statistics 2015; BPS-Statistics Indonesia: Jakarta, Indonesia, 2016; p. 22.

[2] BPS-Statistics of DKI Jakarta Province. Jakarta in Figures 2016; BPS Statistics of DKI Jakarta Province: Jakarta, Indonesia-, 2016; p. 450.

[3] Pan, J. S. Vehicle re-Routing Strategies for Congestion Avoidance. [Ph.D. dissertation], Department of Computer Science, NJ Inst. of Technol.; Newark, NJ, 2014.

[4] Suciu, G.; Vochin, M.; Vulpe, A.; Fratu, O. Vehicular mobile data collection platform to support the development of intelligent transportation systems. In 24th Telecommun Forum (TELFOR), 2016 pp, 2016; pp. 1-4.

[5] Zhao, Z.; Chen, W.; Wu, X.; Chen, P. C. Y.; Liu, J. LSTM network: A deep learning approach for short-term traffic forecast. IET Intell Syst Trans 2017, 11, 68-75.

[6] Jiang, Z.; Wu, J.; Sabatino, P. GUI: GPS-less traffic congestion avoidance in urban areas with inter-vehicular communication. In IEEE 11th international conference on Mobile Ad Hoc and Sensor Syst. (MASS) 2014, 2014; pp. 19-27.

[7] Milojevic, M.; Rakocevic, V. Distributed road traffic congestion quantification using cooperative VANETs. In 13th Annu Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET) 2014, 2014; pp. 203-210.

[8] Siegel, J. E.; Erb, D. C.; Sarma, S. E. A survey of the connected vehicle landscape--architectures, enabling technologies, applications, and development areas. IEEE Trans Intell Transport Syst 2018, 19, 2391-2406.

[9] Gramaglia, M.; Calderon, M.; Bernardos, C. J. ABEONA monitored traffic: VANET-assisted cooperative traffic congestion forecasting. IEEE Veh Technol Mag 2014, 9, 50-57.

[10] Tossavainen, O.-P.; Blandin, S.; Bayen, A. M.; Iwuchukwu, T.; Tracton, K., An ensemble Kalman filtering approach to highway traffic estimation using GPS enabled mobile devices. In 47th IEEE conference on Decision and Control 2008, D. B. Work (Ed.); 2008; pp. 5062-5068.

[11] Maerivoet, S. Modelling Traffic on Motorways: State-Of-the-Art, Numerical Data Analysis, and Dynamic Traffic Assignment. Ph.D. dissertation, Departement Elektrotechniek ESAT-SCD, Katholieke Universiteit Leuven: Belgium, 2006.

[12] de Souza, A. M.; Yokoyama, R. S.; Maia, G.; Loureiro, A.; Villas, L. Real-time path planning to prevent traffic jam through an intelligent transportation system. In IEEE symposium on Comput. and Commun, ISCC 2016, 2016; pp. 726-731.

[13] Wang, S.; Djahel, S.; Zhang, Z.; McManis, J. Next road routing: A multiagent system for mitigating unexpected urban traffic congestion. Intell Transp Syst IEEE Trans 2016, 17, 2888-2899.

[14] de Souza, A. M.; Yokoyama, R. S.; Botega, L. C.; Meneguette, R. I.; Villas, L. A. Scorpion: A solution using cooperative routing to prevent congestion and improve traffic condition. In IEEE international conference on Comput. and Informativa Technologia; Ubiquitous Computing and Commun.; Dependable, Autonomic and Secure Computing; Pervasive Intell. and Computing 2015, 2015; pp. 497-503.

[15] Pan, J.; Khan, M. A.; Popa, I. S.; Zeitouni, K.; Borcea, C. Proactive vehicle re-routing strategies for congestion avoidance. In IEEE 8th international conference on Distributed Computing in Sensor Syst. 2012, 2012; pp. 265-272.

[16] Codeca, L.; Frank, R.; Faye, S.; Engel, T. Luxembourg SUMO traffic (LuST) scenario: traffic demand evaluation. IEEE Intell Transport Syst Mag 2017, 9, 52-63.

[17] Brennand, C. A. R. L.; da Cunha, F. D.; Maia, G.; Cerqueira, E.; Loureiro, A. A. F.; Villas, L. A. FOX: A traffic management system of computer-based vehicles FOG. In IEEE symposium on Comput. and Commun, ISCC 2016, 2016; pp. 982-987.

[18] Cao, Z.; Jiang, S.; Zhang, J.; Guo, H. A unified framework for vehicle rerouting and traffic light control to reduce traffic congestion. IEEE Trans Intell Transport Syst, IEEE Trans 2017, 18, 1958-1973.

[19] Brennand, C. A. R. L.; de Souza, A. M.; Maia, G.; Boukerche, A.; Ramos, H.; Loureiro, A. A. F.; Villas, L. A. An intelligent transportation system for detection and control of congested roads in urban centers. In IEEE symposium on Comput. and Commun, ISCC 2015, 2015; pp. 663-668.

[20] Pan, J.; Popa, I. S.; Zeitouni, K.; Borcea, C. Proactive vehicular traffic rerouting for lower travel time. IEEE Trans Veh Technol 2013, 62, 3551-3568.

[21] Sun, N.; Han, G.; Duan, P.; Tan, J. A global and dynamitic route planning application for smart transportation. In Theory, Syst and Appl(CCITSA) 1st international conference on Computational Intell 2015, 2015; pp. 203-208.

[22] Griggs, W. M.; R. H. O.; -Hurtado, E.; Crisostomi, F.; Häusler, K.; Massow; Shorten, R. N. A large-scale SUMO-based emulation platform. Intell Transp Syst IEEE Trans 2015, 16, 3050-3059.

[23] Liu, R.; Liu, H.; Kwak, D.; Xiang, Y.; Borcea, C.; Nath, B.; Iftode, L. Themis: A Participatory Navigation System for Balanced.

[24] Asvial, M., Pandoyo, M.F.G. and Arifin, A.S., Internet of Things Solution for Motorcycle Riders to Overcome Traffic Jam in Jakarta Using EBkSP. In 2018 International Conference on Information and Communication Technology Convergence (ICTC), 2018; pp. 636-638. IEEE.