Route Subnetwork Generation using OpenStreetMap Data for Emergency Response Problem Modeling in Indonesia

***Abstract*—Route subnetwork generation is useful in modeling disaster emergency response operation problems such as evacuation, aid distribution, and personnel scheduling. Through this paper, we propose an end-to-end approach of generating a subnetwork based on a list of points of interest (villages, shelters, depots, etc) and publicly available data using a combination of opensource tools. Our final product is an opensource software published in a public code repository. We also present some experiments for three emergency response operations in Indonesia: the Yogyakarta earthquake (2020), Jakarta flood (2013), and Lombok earthquake (2018).**

***Index Terms*—subnetwork, openstreetmap, emergency response**

I. Introduction

Route subnetwork generation is a process of extracting some portion of map data (eg. OpenStreetMap data) to build a network/graph where point of interest (POI) nodes are connected by minimum required routes as edges. This subnetwork is useful in efficiently modeling emergency response problems such as evacuation, aid distribution, and personnel scheduling.

The goal of the subnetwork generation is to process a list of POI (POI ID, POI category, latitude, and longitude) to generate a subnetwork network/graph where:

1) Each node represents POI or route intersection
2) Each edge represents route between node
3) Only routes that are required to connect POI included
4) Each node has a risk index (greater value means greater risk)
5) Each routes has a risk index derived from its nodes' risk

II. Related Works

OpenStreetMap.org (OSM) [1] is an open community-driven map data provider that supplies standards and data needed to build a subnetwork for emergency response problem modeling. The problem is that it contains too much data so filtering and extraction are needed.

OpenStreetMap.id is a website that provides extracted OSM data for most provinces in Indonesia in Protocolbuffer Binary Format (PBF). This website solved the issue regarding regional data extraction.

InaRISK [2] is a public Geographic Information System provided by The National Agency for Disaster Countermeasure (BNPB) of Indonesia which was launched in 2016. This system provides information about the disaster risk index of all regions in Indonesia in a form of a colored map layer as shown in Figure 1.

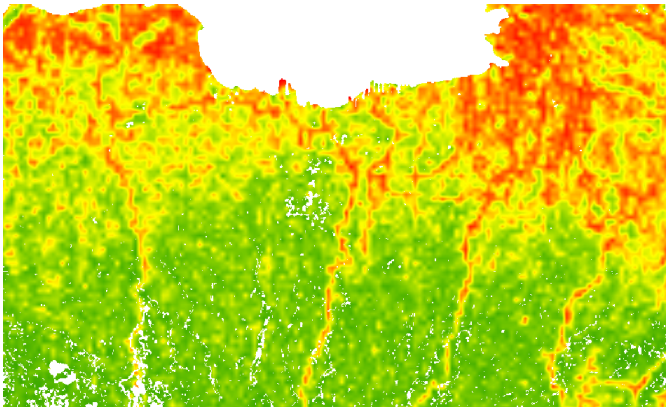

Fig. 1. Jakarta flood disaster risk index from INARISK

OsmToRoadGraph [3] is an opensource tool to generate graph/network from raw OSM data. Since the OpenStreetMap.id only provides PBF files, we will need an additional tool called Osmconvert [4] to convert PBF to OSM before using OsmToRoadGraph to generate graph/network data in PYCGR/PYCGRC and JSON format.

PostGIS [5] is a spatial data extension of one of the most popular opensource database, PostgreSQL [6]. PostGIS allows us to map each POI provided by the user to their nearest node in the PYCGR file. The only thing left to solve is to remove unnecessary nodes and routes from the graph to reduce the size of final input data for the model.

NetworkX [7] is an opensource Python package that provides data structure and APIs for complex graph search and manipulations. This package provides a way to remove unnecessary nodes and routes to generate minimized subnetwork/subgraph suitable for modeling.

III. Methodology

In a nutshell, the methodology that we propose is combining related tools into a new system to generate route subnetwork from POI of emergency response operation. The flowchart of the system is shown in Figure 2.

Our proposed Subnetwork Generator requires five inputs (shown as yellow documents and manual input in the flowchart):

1) **Locations/POI file (*.csv)**. This is a list of locations or points of interest in an emergency operation in Comma Separated Value (CSV) for-

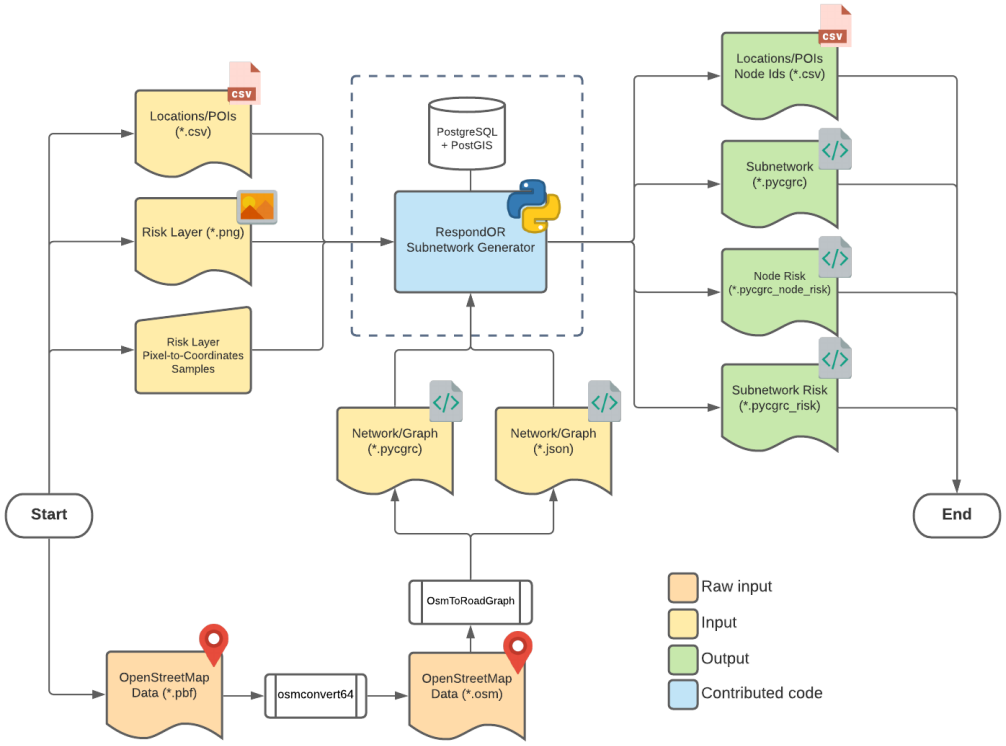

Fig. 2. End-to-end network generation system flowchart

TABLE I
EXPERIMENT RESULTS

No	Operation	Area (km^2)	# POI	% Reduction		Avg Processing Time	
				Nodes	Routes/Edges	AMD 3970X	i7 5500U
1	Yogyakarta earthquake (2020)	32.5	182	97.4%	97.5%	23.24	39.35
2	Jakarta flood (2013)	661.5	346	82.0%	84.2%	26.01	43.88
3	Lombok earthquake (2018)	4,725.0	480	77.0%	78.1%	166.70	268.77

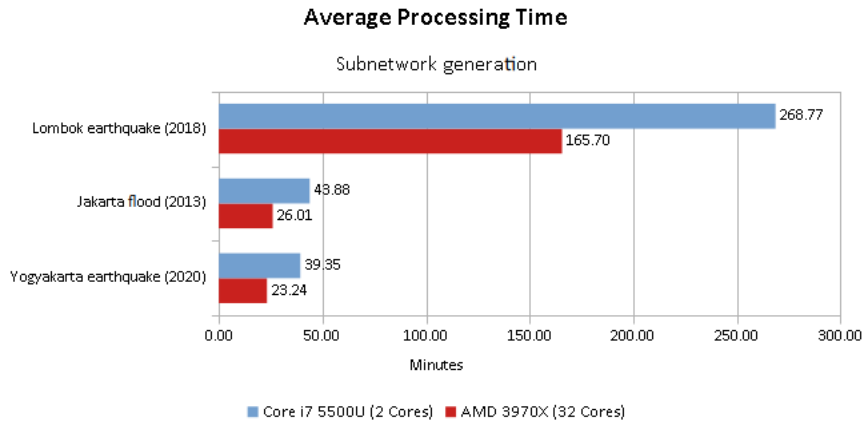

Fig. 3. Comparison of average processing time on two computers

mat. Each row consists of 4 columns: location name, location category (village/shelter/depot/other), latitude, longitude. For example `ANCOL, village, -6.125215, 106.8362474`.

2) **Risk Layer (*.png)**. This is an image representing the disaster risk layer downloaded from InaRISK [2] ArcGIS web service. The web service contains risk layers for many types of disasters (flood, earthquake, volcano eruption .etc) for all populated areas in Indonesia. But we only need to extract only minimal area that covers the whole POI. For example, we can extract the flood risk layer for Jakarta as shown in Figure 1 from https://service1.inarisk.bnpb.go.id:6443/arcgis/rest/services/inaRISK/layer_bahaya_banjir/ImageServer.
3) **Risk Layer Pixel-to-Coordinates Samples**. This is a list of map of pixel-and-coordinates pairs that will be used to estimate the risk index (based on color) of coordinates (latitude, longitude) within the network (locations and routes). These samples must be obtained manually by comparing the Risk Layer and the map beneath it using the InaRISK ArcGIS web service. Our generator will need at least three samples to work. The format of each sample is `[[x,y],[latitude,longitude]]`. For example: `[[199, 151], [-6.124142, 106.656685]]`.
4) **Network/Graph file (*.pycgrc)**. This is a graph representation of OpenStreetMap data generated by OsmToRoadGraph [3]. It is a text file with a specific format where line 1-7 contains comments, line 8 contains the number of nodes, line 9 contains the number of edges/routes, and the rest of the lines are respectively the nodes (ID, latitude, longitude) and the routes (source node ID, target node ID, length, street type, maximum speed, bidirectionality). The nodes included in this network/graph file don't only represent locations but also road junctions/intersections. Further details can be found in the OsmToRoadGraph Github repository https://github.com/AndGem/OsmToRoadGraph.
5) **Network/Graph file (*.json)**. This is also a graph representation of OpenStreetMap data generated by OsmToRoadGraph [3] but in NetworkX [7] JSON format. This input file will be used to find minimal edges/routes efficiently using the NetworkX library.

As indicated in the flowchart, the last two of the input files are the result of preprocessing the raw input, OpenStreetMap Protocolbuffer Binary Format (PBF) file, using existing tools: osmconvert [4] and OsmToRoadGraph [3]. In the case where we can obtain the uncompressed OpenStreetMap data (*.osm) we can directly use OsmToRoadGraph to produce necessary input files.

Our proposed subnetwork generator (the only blue process in the flowchart) does the following steps to generate the route subnetwork:

- Loads OSM nodes name and coordinates (latitude & longitude) from network/graph file (*.pycgrc) to PostGIS-extended PostgreSQL database
- Loads POI list and find nearest OSM nodes using PostGIS distance operator https://postgis.net/docs/geometry_distance_knn.html
- Loads JSON graph and finds minimal nodes/edges required by combining shortest paths of POI permutation pairs. Since the existing shortest-path algorithm takes time to run, the processes are run in parallel utilizing multicores CPU
- Finds risk index of each node by converting pixel colors in InaRISK Risk Layer image (*.png). Manually input pixel-to-coordinates samples are used to estimate the corresponding pixel for each node's coordinates (latitude/longitude)
- Writes output files that represent the route subnetwork, risk index, and mapping to input POI

At the end of the process, our generator produces four outputs:

1) **Locations/POI Node Ids (*.csv)**. The nearest OpenStreetMap node IDs are mapped to each location/POI and appended to the original CSV input file. By having these node IDs, we can link each location/POI to subnetwork (`*.pycgrc`) and risk files (`*.pycgrc_node_risk` and `*.pycgrc_risk`).
2) **Subnetwork (*.pycgrc)**. This is an extracted version of the original network/graph file (*.pycgrc). Our generator will create a new file with a similar name but with a prefix. For example, if the input name is `jakarta.pycgrc` then the subnetwork output file will be `jakarta_subnetwork.pycgrc`.
3) **Node Risk (*.pycgrc_risk)**. This file contains a copy of subnetwork nodes but is appended with the risk index of the node (0.0-1.0). The bigger the value is the more dangerous the route is.
4) **Subnetwork Risk (*.pycgrc_risk)**. This file contains a copy of subnetwork edges/routes appended with an additional value which is the risk index (0.0-1.0). This edge/route risk value is the mean of the source and target nodes' risk indexes.

The source code of our proposed system along with its usage guide are currently published in a publicly accessible code repository https://github.com/yohanesgultom/respondor.

IV. Experiments

In order to validate and test the performance of our proposed system, we generated route subnetworks for three emergency response operations conducted by The National Agency for Disaster Countermeasure (BNPB) of Indonesia:

1) Yogyakarta earthquake (2020)
2) Jakarta flood (2013)
3) Lombok earthquake (2018)

The locations/POI data were collected mainly from BNPB Contingency Plan (*RenKon*) which are available publicly in http://renkon.bnpb.go.id/. While the OSM (PBF) data were provided by OpenStreetMap Indonesia on their website https://

openstreetmap.id/en/data-openstreetmap-indonesia/. The PBF files were converted to network files (*.pycgrc and *.json) using osmconvert [4] and OsmRoadToGraph [3] as explained in the previous section. Finally, the risk layer images and pixel-to-coordinates samples were manually collected from the InaRISK website as mentioned in the previous section.

We used two computers to run the experiment. The first one was a PC-server provided by the Faculty of Computer Science, Universitas Indonesia, with the following specifications:

- CPU: AMD Ryzen Threadripper 3970X (32 Cores)
- RAM: DDR4 64 GB
- Storage: SCSI 4 TB
- OS: Ubuntu 18.04

The other one is a consumer notebook with the following specifications:

- CPU: Intel Core i7 5500U (2 Cores)
- RAM: DDR3L 12 GB
- Storage: SSD 1 TB
- OS: Ubuntu 20.04

We ran each of the processes three times on each computer and obtained results shown in Table I:

- **Operation**: The emergency operation name describing the location, the disaster, and the year
- **Area**: The area of the emergency operation location in km^2
- **# POI**: The number of the points of interest (represented by unique geolocation coordinates) provided as input
- **% Reduction**: The number of reduced/removed **nodes** and **routes/edges** from the original/input network/graph in the result/output subnetwork/subgraph
- **Avg Processing Time**: The average duration of end-to-end processing time (in minutes) from three runs per operation

The result shows that processing time increases along with the number of POI provided as input. But we find a significant increase in processing time for the Lombok earthquake (2018) compared to the Yogyakarta earthquake (2020) and Jakarta flood (2013). This is due to the significantly bigger area of the map which causes the average number of nodes in a route to increase (due to the distance).

Based on the result of our experiments, we concluded that the processing time is mainly affected by two factors:

1) **The number of POI**. The processing time will dramatically increase along with the number of unique locations/POI because our generator needs to find the shortest path of each permutation pair of the POI. The reason we need to use permutation instead of combination is because of the existence of one-way (non-bidirectional) routes.
2) **The area of the map**. Operation map with the bigger area most of the time has many of POI separated by long distance. The longer the distance, the more number of nodes that need to be processed during the shortest path finding which eventually increases the processing time

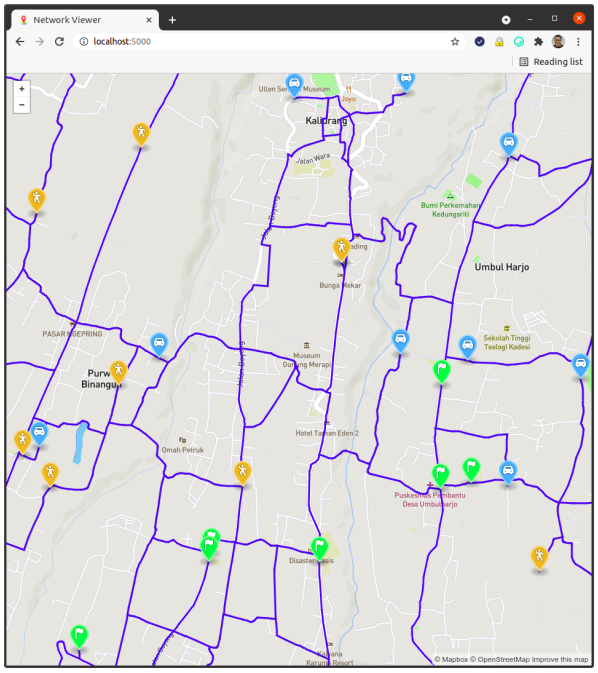

Fig. 4. Visualization of Yogyakarta's subnetwork during earthquake disaster in 2020

We also observe that although increasing the computational power (CPU cores and RAM) significantly reduces the processing time for the largest operation (Lombok earthquake 2018), the improvement is not proportional to the computational power. In our experiment, using a machine with 16 times of CPU cores only reduced the processing time by half (Figure 3) which indicates that the shortest path algorithm used in our system doesn't scale very well.

To validate the generated network easily, we created a simple web server to visualize the subnetwork on top of the interactive map from https://mapbox.com. The visualization of partial subnetworks for Yogyakarta earthquake (2020), Jakarta flood (2013), and Lombok earthquake (2018) are shown respectively in Figure 4, Figure 5, and Figure 6. The blue lines represent the available roads. While the icons represent the locations/POI:

- Yellow icon for a village
- Green icon for a shelter
- Light blue icon for a depot
- Blue icon for a warehouse
- Red icon for a damaged infrastructure

V. Conclusion

Our proposed system was able to generate route subnetwork for all given operations within considerable processing time. The time required to generate the subnetwork is affected by the number of POI and the area of the operation map. The bigger the number of POI and the area, the longer time required to generate the subnetwork.

Since the number of the POI in each operation is usually more than the number of cores of CPU available in the market,

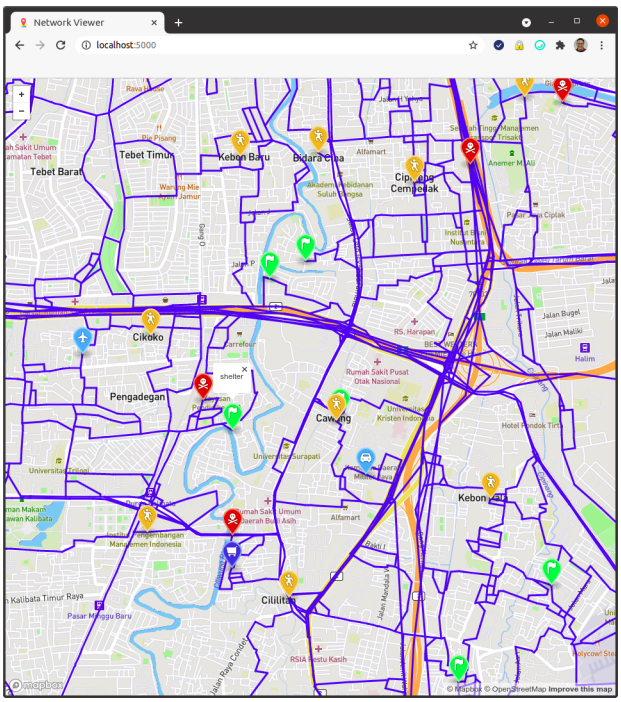

Fig. 5. Visualization of Jakarta's subnetwork during flood disaster in 2013

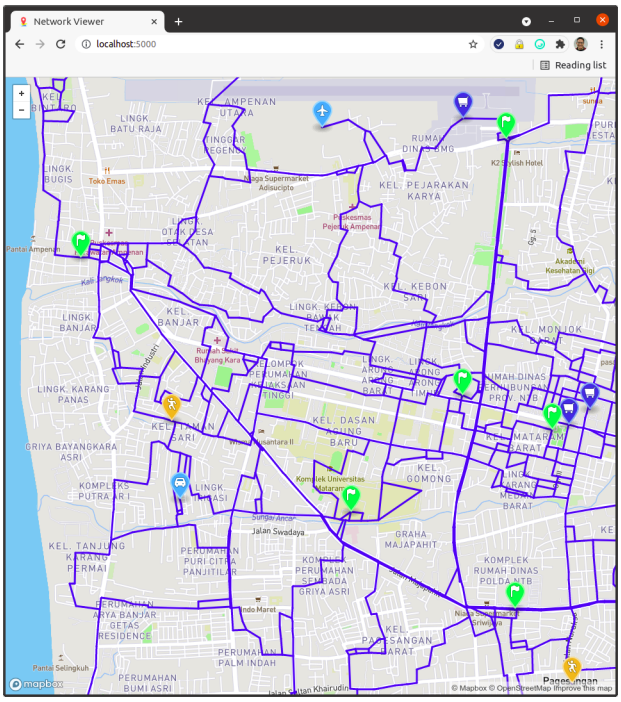

Fig. 6. Visualization of Lombok's subnetwork during earthquake disaster in 2018

we'd suggest integrating the method of running the shortest-path finding algorithm in the Graphics Processing Unit (GPU) which has hundreds to thousands of cores [8].

Acknowledgment

This research was supported by the RESPOND-OR project[1], which is led by Lancaster University, Centre for Transport and Logistics (CENTRAL), in partnership with Universitas Indonesia, Universitas Gadjah Mada, and the University of Khartoum. This project is also supported by an Advisory Committee of stakeholders from government and non-government organizations in Indonesia and Sudan to ensure the alignment of the scientific activities, project results, and deliverables to the stakeholders' needs.

References

[1] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *IEEE Pervasive computing*, vol. 7, no. 4, pp. 12–18, 2008.
[2] BNPB. (2016) Inarisk. [Online]. Available: http://service1.inarisk.bnpb.go.id:6080/arcgis/rest/services/inaRISK
[3] A. Gemsa. (2017) Osmtoroadgraph. [Online]. Available: https://github.com/AndGem/OsmToRoadGraph
[4] OpenStreetMap. (2019) Osmconvert. [Online]. Available: https://wiki.openstreetmap.org/wiki/Osmconvert
[5] PostGIS. (2001) Postgis. [Online]. Available: https://postgis.net/
[6] PostgreSQL. (1996) Postgresql. [Online]. Available: https://www.postgresql.org/
[7] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference*, G. Varoquaux, T. Vaught, and J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11 – 15.
[8] P. Harish and P. J. Narayanan, "Accelerating large graph algorithms on the gpu using cuda," in *International conference on high-performance computing*. Springer, 2007, pp. 197–208.

[1]https://www.lancaster.ac.uk/lums/research/areas-of-expertise/centre-for-transport-and-logistics/projects/respond-or/