Application of Minimum Spanning Tree for Mail Distribution in Surabaya

Ahmad Alfani Handoyo 13520023[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13520023@std.stei.itb.ac.id

***Abstract*—Modelling a mail distribution route for a big city requires a precise balance between operational costs and interconnectivity. This paper proposes a hypothetical mail distribution route for the city of Surabaya, Indonesia by using a minimum spanning tree. The minimum spanning tree is found by using a classic Prim's algorithm. This paper also tries to determine the center of the minimum spanning tree to represent a central office in a network of post or courier offices.**

***Keywords*—Center of a tree, Mail distribution route, Minimum spanning tree, Prim's algorithm.**

I. Introduction

Catering for the needs of 2.87 million souls in a city deeply embedded as one of Indonesia's economic hubs is not a walk in the park. Year by year as Surabaya's economy continues to rise, so does the overall quality of life of its citizens. Suddenly not only the richest of rich can blow away their money for leisure activities and luxury.

Behind the spectacles, it is truly the emergence of online shopping which plays a massive role in accelerating the growth of the local economy, as suddenly everyone can open shop while having the most adequate of funding which would otherwise be impossible in a traditional shop or market.

Around all corners of Surabaya, a lot of merchants are turning to online shopping due to its simplicity and ease. A house that may look to be purely inhabitable can be a little warehouse for an online boutique shop with boxes upon boxes piled up in the front yard, waiting to be delivered by a courier service. Moreover, the local city government is very reputable in supporting small to medium businesses to be able to grow, as it, in turn, helps the local economy to grow as well.

As the demand for online delivery in Surabaya continues to skyrocket in the past few years, postal or courier services should seize this opportunity, filling in the gap to be able to supply the services needed to deliver these goods.

This paper aims to accommodate that goal by trying to model a hypothetical mail distribution route that is efficient in covering every district in Surabaya. The absolute necessity of a connection between one district and another is heavily analyzed using a minimum spanning tree which is constructed using a classic Prim's algorithm. Hopefully, this paper can serve its function as a guide or reference for postal or courier services looking to expand their business in Surabaya, as numerous other competitors have.

II. Basic Theory

A. Graph

A graph is a discrete set of vertices, each representing a distinct point, and edges that bridge the connection between these vertices. Formally, a graph can be mathematically described as a pair,

$$G = (V, E)$$

where G is the graph itself, V a non-empty set of vertices v_1, v_2, ..., v_n, and E a set of edges e_1, e_2, ..., e_n which connects a pair of vertices.

Notice that a graph must have at least a vertex. Yet, the same does not need to apply for edges, meaning a graph may have vertices that are not connected by any edges at all.

Graphs may have multiple edges connecting the same pair of vertices and loops, edges with both ends connected to the same vertex. An edge may also have a direction, indicating its starting and pointing vertices. The presence of multiple edges and loops and the directedness of the edges in a graph determine the type of graph it is.

Based on the existence of multiple edges and/or loops, a graph can be categorized as a simple or an unsimple graph. A simple graph is a graph that has neither multiple edges nor loops as shown in Fig. 2.1a, whereas an unsimple graph is the opposite, allowing the presence of these multiple edges and/or loops.

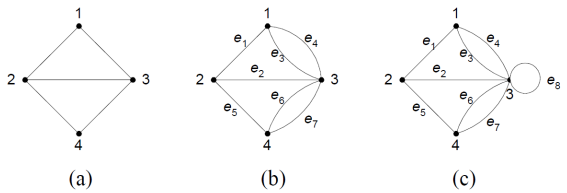
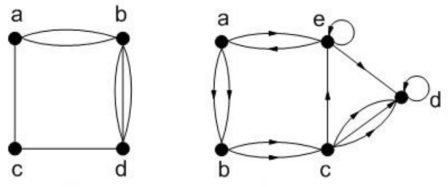


Fig. 2.1. (a) simple graph, (b) multigraph, and (c) pseudograph [1].

An unsimple graph can then be further differentiated between a multigraph and a pseudograph. Multigraphs may have multiple edges connecting the same vertices as shown in Fig. 2.1b where the edges e_3 and e_4 both connect vertices v_1 and v_3. On the other hand, graphs that may have loops, and possibly multiple edges, are called pseudographs. This is shown in Fig. 2.1c where the graph has a looping edge on e_8 having both ends connecting the same vertex v_3, while also having multiple edges as does a multigraph on edges e_3 and e_4.

Based on the directedness of its edges, a graph can be classified between an undirected graph and a directed graph.



(a) (b)

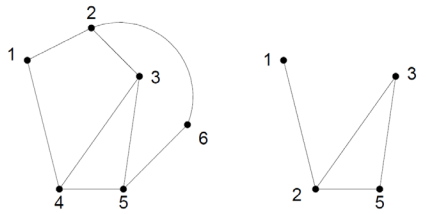
Fig. 2.2. (a) undirected graph and (b) directed graph [2].

An undirected graph does not factor in the directionality of its edges, meaning that it is impossible to know the start and end of an edge. This is shown in Fig. 2.2a with the two edges between vertices v_a and v_b. On the other hand, a directed graph has edges that specify their direction. In Fig. 2.2b the direction of the two edges between vertices v_a and v_e, is distinguishable as one edge starts at v_a and ends at v_e, while the other edge starts at v_e and ends at v_a.

Combining the types of graphs above, it is then possible to make other more complicated variations of graphs. A simple directed graph has directed edges while having no loops or multiple edges. A directed multigraph has directed edges while allowing loops and multiple edges. A mixed graph, while also having loops and multiple edges, has both directed and undirected edges. It then depends on the need in applying it in real-life scenarios in which type of graph is to be used.

There are a few common graph terminologies that will be used in this paper:

1. Adjacency
   A pair of vertices u and v is said to be adjacent if both vertices are endpoints of an edge e. Take for example in Fig. 2.1b, v_1 and v_3 are adjacent as edges e_3 and e_4 connect both vertices. In contrast, v_1 and v_4 are not adjacent.
2. Incidence
   An edge $e = (u, v)$ is said to be incident with vertices u and v if u and v are the endpoints of edge e.
3. Degree
   The degree of a vertex u is the total number of edges that are incident with u. A loop with both endpoints at u adds two to the degree of vertex u.
4. Path
   A path is a sequence of edges that starts at a specific vertex, which continues to traverse along its edges and pass through their incident vertices.
5. Circuit or cycle
   A circuit or a cycle is a path that starts and ends at the same vertex, having a length greater than zero. If every edge visited is different, then the path or circuit is said to be simple.
6. Connectedness
   Two vertices u and v are said to be connected if there exists a path that goes from u and v or vice versa. Moreover, a graph is said to be a connected graph if for every pair of vertices v_1 and v_2 in the graph, there exists a path that goes from v_1 and v_2 or vice versa.
7. Subgraph
   There exist a subgraph $G_1 = (V_1, E_1)$ of a graph $G = (V, E)$ if $V_1 \subseteq V$ and $E_1 \subseteq E$.



(a) (b)

Fig. 2.3. (a) graph G and its (b) subgraph G_1 [1].

As can be seen in Fig. 2.3, subgraph G_1 has vertices and edges which is a subset of all vertices and edges of G.

8. Spanning subgraph
   A subgraph $G_1 = (V_1, E_1)$ is said to be the spanning subgraph of $G = (V, E)$ if $V_1 = V$, that is the vertices of G_1 contain all the vertices of G.
9. Weighted graph
   A weighted graph is a graph in which each edge is given a specific value or weight. This value can represent anything in the real world such as price or distance.

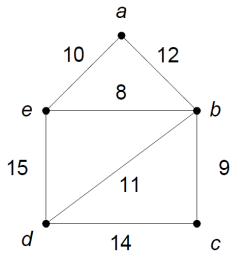
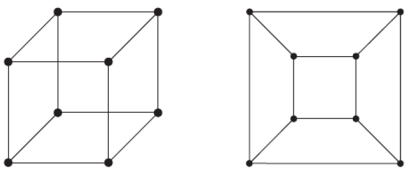


Fig. 2.4. A weighted graph [1].

B. Planar and Dual Graph

A graph is considered a planar graph if it can be drawn on a flat plane without any of its edges crossing each other. Keep in mind, a graph does not necessarily have to not have crossings when drawn to be considered planar, as it may be possible to draw the graph in another way such that there are no crossing edges. A drawing of a planar graph in which there are no crossing edges is called a planar representation of the graph, or a plane graph. The edges of a plane graph can then divide the graph into regions or faces. When a graph cannot be represented as its planar representation, it is said to be non-planar.



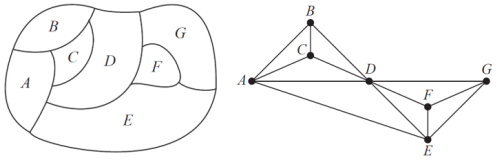
(a) (b)

Fig. 2.5. (a) a planar graph and its (b) planar representation or plane graph [2].

As can be seen above, the graph in Fig. 2.5 can be represented as a plane graph with no crossing edges, thus confirming that it is a planar graph. The plane graph itself has 6 regions/faces, including the outer region.

A plane graph G can then be converted into its dual graph G^* by turning every face of G as a vertex of G^*. Then, draw edges connecting all vertices of G^*, passing through all the original

edges of *G*.



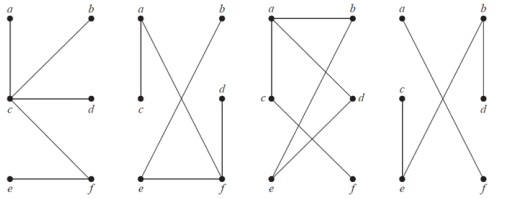
(a) (b)

Fig. 2.6. (a) a map and its (b) dual graph representation [2].

A dual graph becomes a handy tool to represent a map in the form of a graph, as can be seen in Fig. 2.6. Each corresponding region is converted into a vertex and edges are drawn for each pair of faces/regions that share a common border.

C. Tree

A tree is a connected, undirected, and simple graph that has no circuits. A consequence of not having any circuits means that a tree does not have multiple edges or loops either, thus confirming the condition of a tree to be a simple graph.



(a) (b) (c) (d)

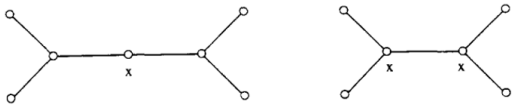
Fig. 2.7. (a) and (b) are trees while (c) and (d) are not trees [2].

Interestingly, let there be a simple undirected graph *G = (V, E)* with a total number of vertices *n*. If one of the statements below applies, all the other statements apply and are equivalent. These statements also make up the properties of a tree:

1. *G* is a tree,
2. Every pair of vertices in *G* is connected by a unique simple path,
3. *G* is connected and has *n-1* edges,
4. *G* does not have any circuits and has *n-1* edges,
5. *G* does not have any circuits and the addition of an edge in the graph will form a circuit, and
6. *G* is connected and all its edges are bridges.

D. Center of a Tree

As proven by Jordan (1869), a tree will always have either a single vertex or a pair of adjacent vertices as its center. This can be seen in the examples in Fig. 2.8 with *X* labeled as the center of the tree.



(a) (b)

Fig. 2.8. Tree with (a) a single vertex and (b) two adjacent vertices as its center [4].

There are two possible methods to find the center of a tree. The first method is to find all the longest paths in a tree, and the center would be the middle or the two middle vertices in every longest path. Finding the longest path itself requires a traversal through all the combinations of paths possible in the tree.

The second method is to iteratively remove the leaves (vertices with a degree of 1) of the tree. Gradually remove the outermost leaves of the tree successively obtaining smaller trees in each step, and in the end, the centermost vertex/vertices are left.

In a network consisting of a weighted tree (as will be discussed in the paper), it is more desirable to find the center of the unweighted version of the tree, ignoring the complexifying effects of the weights on the edges.

G. Minimum Spanning Tree

A spanning tree of a connected graph is the spanning subgraph of a graph that is a tree. It can be obtained by removing edges until a circuit can no longer be formed in the graph. A property that emerges is that a connected graph will have at least a single spanning tree.

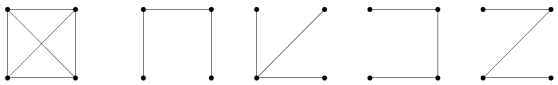


Fig. 2.9. A graph (leftmost) with its possible spanning trees [1].

When a graph has weighted edges, it is then possible to construct its minimum spanning tree. A minimum spanning tree is a spanning tree that has the minimum total weight of edges possible for all variations of spanning trees in the graph. An important side note is that a graph may have more than one minimum spanning tree if several of its edges have the same weight, thus forming different combinations of edges while also having the same total weight. The application of a minimum spanning tree is truly endless and vast. It is commonly used in network scenarios trying to strive for the lowest cost, which is easily represented in a minimum spanning tree. As such is the case for the main discussion of this paper.

H. Prim's Algorithm

There are several algorithms for constructing the minimum spanning tree of a graph. Such algorithms have their own merits and demerits in terms of time complexity, depending on how the algorithm is run and the data structures used. The two classic and most frequently used algorithms are Prim's algorithm and Kruskal's algorithm. While both have fundamentally similar time complexities for sparse graphs, Prim's algorithm comes somewhat ahead in denser graphs that have more edges.

Prim's algorithm builds a minimum spanning tree by progressively adding a new edge to the growing tree. Setting up the tree, depending on the variation of the algorithm, start either with any arbitrarily chosen vertex or the minimum weight edge of the graph. Then, iteratively take the next minimum weight edge which connects a vertex on the tree to another vertex not yet on the tree. By choosing an edge that connects a vertex on the tree to one that is not yet on the tree, it is ensured that no circuits will be formed. Repeat the iteration until all the vertices in the graph are connected, forming a complete minimum spanning tree.

III. Constructing the Mail Distribution Route

A. Modelling the Problem

A mail distribution route like the one proposed in this paper should in general take two important factors into account. Firstly, the route must be able to reach all its intended destinations which in this case would be the branches of post or courier offices tending the mail services in each district. Countless variants and types of graphs or networks should suffice in completing this task. However, perhaps the more important factor to consider is that the route must also be efficient in connecting the various branches of post offices. Simply connecting all the branches is not enough as there are operational costs involved such as fuel and labor. These operational costs should in principle be proportional to the distance needed to cover each connection.

A postal or courier service is a business after all and like all conventional businesses should strive for the greatest efficiency to minimize costs in its operation while maintaining quality to keep its customers happy. By removing a few connections between branches, the overall production costs can be reduced. However, arbitrarily removing edges may in turn hurt the delivery time between a branch and another. This perplexing problem to maintain the delicate balance between costs and delivery time can be solved by finding the minimum spanning tree of the graph. The resulting minimum spanning tree will be the final mail distribution route.

Several studies have been conducted in the past to try to tackle distribution route issues like the one at hand. Shahin and Jaferi (2015) modelled a minimum spanning tree by Prim's algorithm amongst other algorithms to find the theoretical shortest route for an Iranian factory producing car batteries to distribute its products from Tehran to every province of Iran. Here the collecting points are presumed to be the center of each province, which in its modelled tree would be its vertices. Neagu et al. (2015) explored the optimization of transportation networks to accommodate the delivery needs of the postal services in regions of Romania by a Kruskal's algorithm-generated minimal spanning tree. Each vertex in the minimum spanning tree represents an already existing post office in each region.

In modelling the graph and ultimately its minimum spanning tree to represent the mail distribution route, several key features of the graph or tree must be decided beforehand.

Firstly, the graph itself is modelled by making the dual graph representation of districts in Surabaya. The resulting graph would be undirected, simple, and connected which is required to find its minimum spanning tree. A vertex represents a post or courier office branch unique to each district. An edge represents a connection between a pair of office branches, allowing the distribution of mail and packages from one branch to another. However, edges are only allowed to be drawn when these districts share a common border.

Secondly, it is assumed that all the branches are connected in some way, thus validifying the requirement for Prim's algorithm for the graph to be connected. The graph itself would also have to have weighted edges, storing the data of distance between each branch in the weighted edges, enabling the construction of the graph's minimum spanning tree itself.

Thirdly, the location of each post or courier office branch representing the points or vertices in the graph must be decided as the route constructed here and therefore the location of each office branch is theoretical. Analyzing the examples mentioned before, the network in Neagu et al. (2015) uses preexisting post offices as its vertices while Jaferi (2015) assumes each branch to be located at the center of each province. It is then decided that each branch in its corresponding district is assumed to be located in close vicinity to the local district office. This decision is based upon the fact that district offices will in most cases be in the local central business area of each district, allowing an easier reach for local deliveries to be made to the general district population.

Lastly, the weight of an edge is assumed to be the radial distance between each pair of adjacent post office branches incident with the edge. Rather than following the paths of physical roads, this assumption is made for the sake of showing the connection between each branch. Due to the characteristic of Surabaya's district borders in which the size of each district is somewhat evenly distributed and no district is separated by sea borders, all connections should cross through actual land rather than crossing any water mass which would further complicate any estimation regarding efficiency due to the extra costs needed to travel by water.

Once the weighted graph is modelled, the minimum spanning tree is then constructed by applying Prim's algorithm until all branches are connected. Finally, the center vertex/vertices of the tree are to be found. This center vertex/vertices acts as the central office for Surabaya, becoming a gateway for all incoming and outgoing mail and packages connecting the mail network of Surabaya to all cities of Indonesia.

B. Collecting Data and Modelling the Weighted Graph

According to the Body for Government Administration and Regional Autonomy of Surabaya, the city has 31 districts in which each district has a local district office. To accurately pinpoint the location of each district office, their respective coordinates are used and can be easily measured using Google Maps. The location of each district office then represents its respective district vertex v_n with n being its numbering. The resulting vertices are shown in Table 3.1 below.

Table 3.1. Coordinates for each district vertex.

Vertex	District	Latitude	Longitude
v_1	Pakal	-7.2403488	112.62541
v_2	Benowo	-7.2488194	112.63525
v_3	Asemrowo	-7.2521531	112.71529
v_4	Tandes	-7.2591761	112.67791
v_5	Sambikerep	-7.2611079	112.65213
v_6	Lakarsantri	-7.3034317	112.63251
v_7	Karang Pilang	-7.3333161	112.69941
v_8	Wiyung	-7.3144402	112.69525
v_9	Dukuh Pakis	-7.2833257	112.68817
v_{10}	Sukomanunggal	-7.2607884	112.71239
v_{11}	Krembangan	-7.2328676	112.72258
v_{12}	Pabean Cantian	-7.2169835	112.72942
v_{13}	Bubutan	-7.2516552	112.73411
v_{14}	Sawahan	-7.2840879	112.71608
v_{15}	Jambangan	-7.3221961	112.71387
v_{16}	Gayungan	-7.3381504	112.71677
v_{17}	Wonoloco	-7.3199923	112.74116

v_{18}	Wonokromo	-7.2916684	112.73219
v_{19}	Tegalsari	-7.2880078	112.74052
v_{20}	Genteng	-7.2577884	112.75181
v_{21}	Simokerto	-7.2436234	112.75794
v_{22}	Semampir	-7.2253347	112.7445
v_{23}	Kenjeran	-7.2266571	112.77543
v_{24}	Tambaksari	-7.2575164	112.75523
v_{25}	Gubeng	-7.2716891	112.75596
v_{26}	Tenggilis Mejoyo	-7.3139426	112.75699
v_{27}	Gunung Anyar	-7.3409584	112.7833
v_{28}	Rungkut	-7.3229193	112.77099
v_{29}	Sukolilo	-7.2997009	112.77032
v_{30}	Mulyorejo	-7.2614324	112.78484
v_{31}	Bulak	-7.2315981	112.78544

The next step is to examine the edges connecting the vertices. For each pair of vertices of districts that share a common border, draw an edge that is incident with these vertices. Finding these edges can be done by examining a map which in this case is based on the map in Fig. 3.1.

Fig. 3.1. A map showing the district borders of Surabaya [12].

The weight of an edge is the distance between its adjacent district vertices. This can be measured in Google Maps once again which has a feature to calculate the radial distance between two points measured in kilometers. In this case the two points are the coordinates of the pair of district vertices previously measured. Repeat this process until all districts sharing common borders are represented by an edge. Finally, the resulting edges e_x with x being their numbering are shown in Table 3.2 below. As can be seen, there are a total of 70 edges connecting the districts.

Table 3.2. Edges connecting the district vertices.

Edge	Vertices	Distance (km)	Edge	Vertices	Distance (km)
e_1	(v_1, v_2)	1.44	e_{36}	(v_{14}, v_{19})	2.73
e_2	(v_2, v_3)	8.83	e_{37}	(v_{14}, v_{20})	4.91
e_3	(v_2, v_4)	4.85	e_{38}	(v_{15}, v_{16})	1.80
e_4	(v_2, v_5)	2.31	e_{39}	(v_{15}, v_{18})	3.95
e_5	(v_3, v_4)	4.19	e_{40}	(v_{16}, v_{17})	3.36
e_6	(v_3, v_{10})	1.01	e_{41}	(v_{16}, v_{18})	5.44
e_7	(v_3, v_{11})	2.29	e_{42}	(v_{17}, v_{18})	3.30
e_8	(v_3, v_{13})	2.09	e_{43}	(v_{17}, v_{25})	5.62
e_9	(v_3, v_{14})	3.55	e_{44}	(v_{17}, v_{26})	1.87
e_{10}	(v_4, v_5)	2.85	e_{45}	(v_{18}, v_{19})	1.00
e_{11}	(v_4, v_9)	2.91	e_{46}	(v_{18}, v_{25})	3.44
e_{12}	(v_4, v_{10})	3.81	e_{47}	(v_{19}, v_{20})	3.58
e_{13}	(v_5, v_6)	5.18	e_{48}	(v_{19}, v_{25})	2.49
e_{14}	(v_5, v_8)	7.60	e_{49}	(v_{20}, v_{21})	1.71
e_{15}	(v_5, v_9)	4.68	e_{50}	(v_{20}, v_{24})	0.38
e_{16}	(v_6, v_7)	8.09	e_{51}	(v_{20}, v_{25})	1.61
e_{17}	(v_6, v_8)	7.03	e_{52}	(v_{21}, v_{22})	2.52
e_{18}	(v_7, v_8)	2.15	e_{53}	(v_{21}, v_{23})	2.70
e_{19}	(v_7, v_{15})	2.02	e_{54}	(v_{21}, v_{24})	1.57
e_{20}	(v_8, v_9)	3.55	e_{55}	(v_{22}, v_{23})	3.41
e_{21}	(v_8, v_{15})	2.23	e_{56}	(v_{23}, v_{24})	4.09
e_{22}	(v_9, v_{10})	3.66	e_{57}	(v_{23}, v_{31})	1.23
e_{23}	(v_9, v_{14})	3.08	e_{58}	(v_{24}, v_{25})	1.58
e_{24}	(v_9, v_{15})	5.17	e_{59}	(v_{24}, v_{30})	3.30
e_{25}	(v_9, v_{18})	4.94	e_{60}	(v_{24}, v_{31})	4.41
e_{26}	(v_{10}, v_{14})	2.63	e_{61}	(v_{25}, v_{26})	4.70
e_{27}	(v_{11}, v_{12})	1.92	e_{62}	(v_{25}, v_{29})	3.49
e_{28}	(v_{11}, v_{13})	2.45	e_{63}	(v_{25}, v_{30})	3.38
e_{29}	(v_{12}, v_{13})	3.89	e_{64}	(v_{26}, v_{27})	4.17
e_{30}	(v_{12}, v_{20})	5.17	e_{65}	(v_{26}, v_{28})	1.84
e_{31}	(v_{12}, v_{21})	4.32	e_{66}	(v_{26}, v_{29})	2.16
e_{32}	(v_{12}, v_{22})	1.91	e_{67}	(v_{27}, v_{28})	2.42
e_{33}	(v_{13}, v_{14})	4.12	e_{68}	(v_{28}, v_{29})	2.58
e_{34}	(v_{13}, v_{20})	2.07	e_{69}	(v_{29}, v_{30})	4.54
e_{35}	(v_{14}, v_{18})	1.97	e_{70}	(v_{30}, v_{31})	3.32

From the vertices and edges previously obtained, a simple, undirected, connected, and weighted graph can then be constructed as can be seen in Fig. 3.2. It is important to note that the graphical representation presented is not scaled proportionally in all its edges as the vertices are placed merely as a representing point of the districts, rather than in its precise location of a post office branch coordinate. However, this information of distance is still held by the weight of the edges.

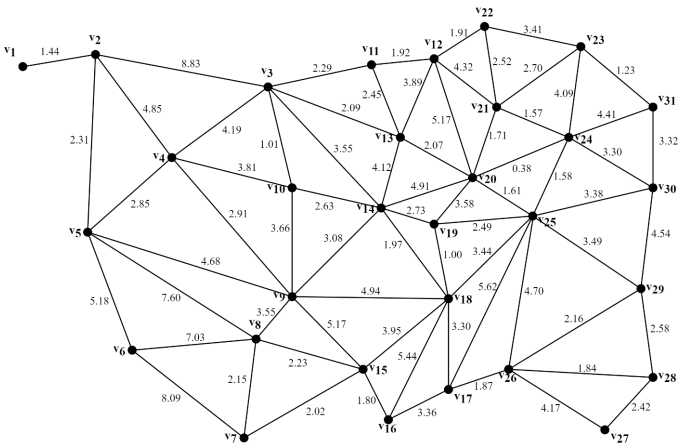


Fig. 3.2. Graph of post office branches in Surabaya and their possible distribution routes.

C. Finding the Minimum Spanning Tree

Prim's algorithm is run to obtain the minimum spanning tree of the graph. As a few edges have identical weights, multiple variations of the minimum spanning tree are possible. From the two variations of the algorithm discussed in the previous section, the start with the minimum weight edge of the graph is chosen.

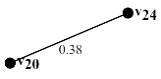


Fig 3.3. Start with edge e_{50} with weight 0.38 km.

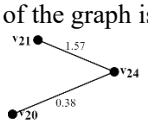


Fig. 3.4. Add edge e_{54} with weight 1.57 km.

Fig. 3.5. Add edge e_{58} with weight 1.58 km.

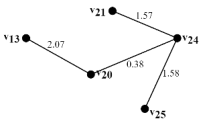


Fig. 3.6. Add edge e_{34} with weight 2.07 km.

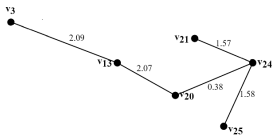


Fig. 3.7. Add edge e_8 with weight 2.09 km.

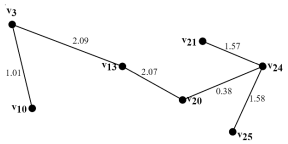


Fig. 3.8. Add edge e_6 with weight 1.01 km.

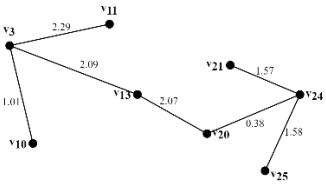


Fig. 3.9. Add edge e_7 with weight 2.29 km.

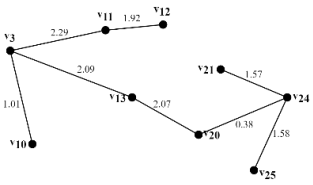


Fig. 3.10. Add edge e_{27} with weight 1.92 km.

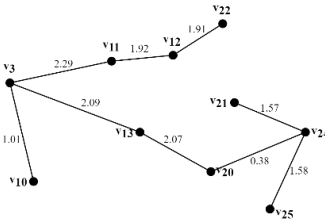


Fig. 3.11. Add edge e_{32} with weight 1.91 km.

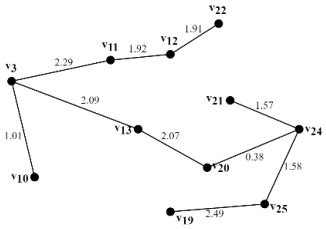


Fig. 3.12. Add edge e_{48} with weight 2.49 km.

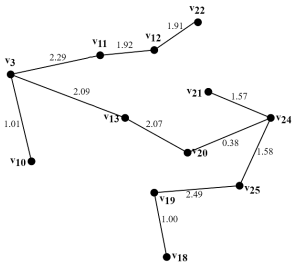


Fig. 3.13. Add edge e_{45} with weight 1.00 km.

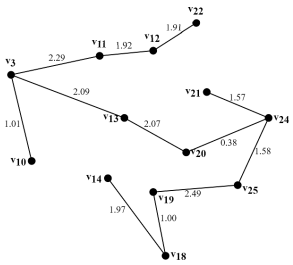


Fig. 3.14. Add edge e_{35} with weight 1.97 km.

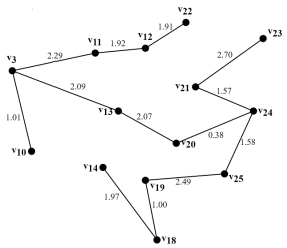


Fig. 3.15. Add edge e_{53} with weight 2.70 km.

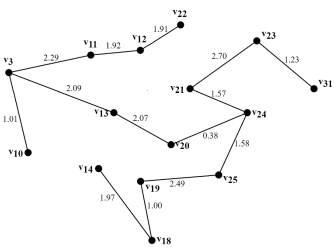


Fig. 3.16. Add edge e_{57} with weight 1.23 km.

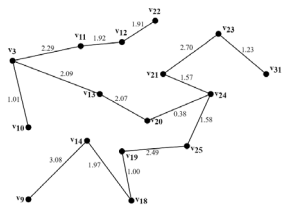


Fig. 3.17. Add edge e_{23} with weight 3.08 km.

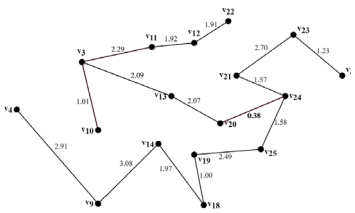


Fig. 3.18. Add edge e_{11} with weight 2.91 km.

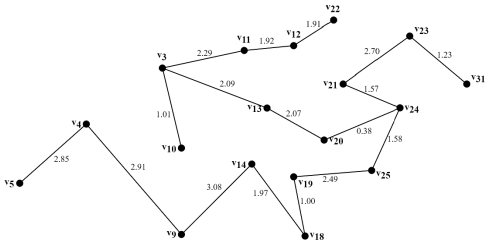


Fig. 3.19. Add edge e_{10} with weight 2.85 km.

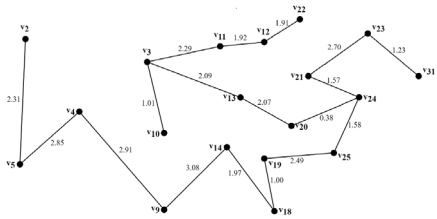


Fig. 3.20. Add edge e_4 with weight 2.31 km.

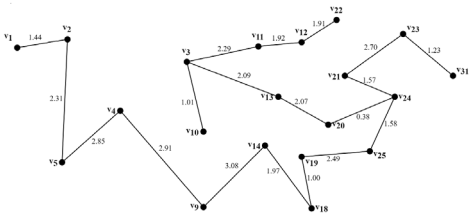


Fig. 3.21. Add edge e_1 with weight 1.44 km.

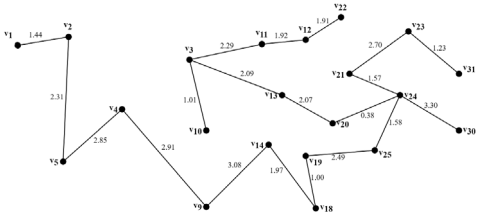


Fig. 3.22. Add edge e_{59} with weight 3.30 km.

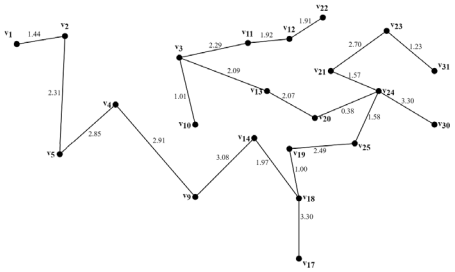


Fig. 3.23. Add edge e_{42} with weight 3.30 km.

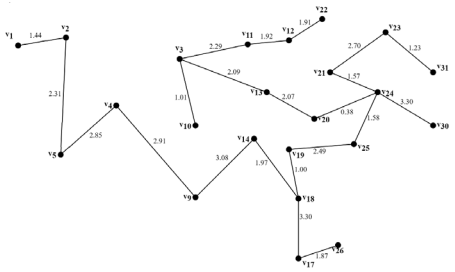


Fig. 3.24. Add edge e_{44} with weight 1.87 km.

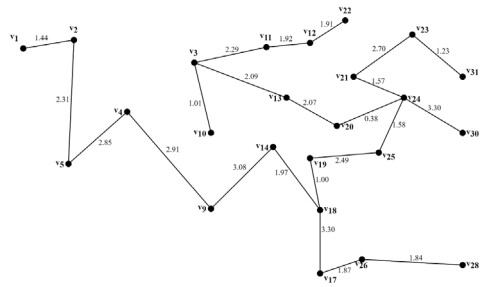


Fig. 3.25. Add edge e_{65} with weight 1.84 km.

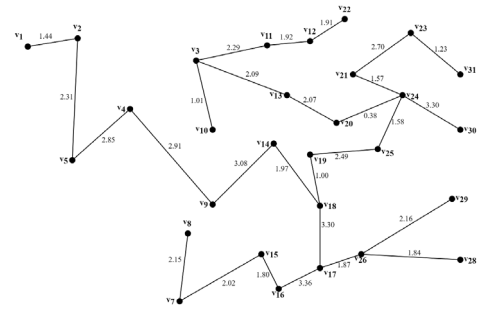


Fig. 3.26. Add edge e_{66} with weight 2.16 km.

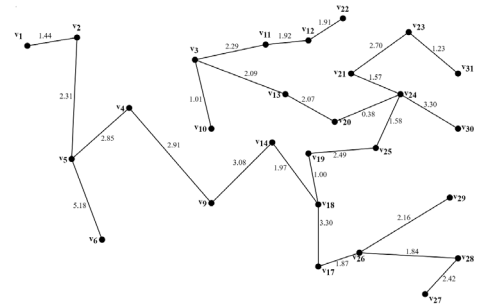


Fig. 3.27. Add edge e_{67} with weight 2.42 km.

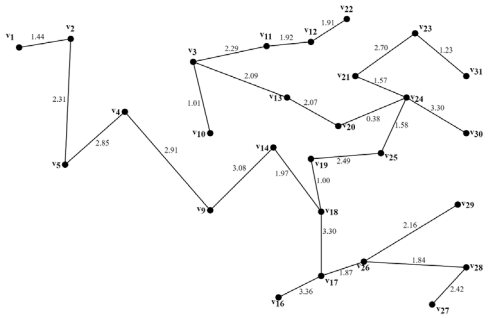


Fig. 3.28. Add edge e_{40} with weight 3.36 km.

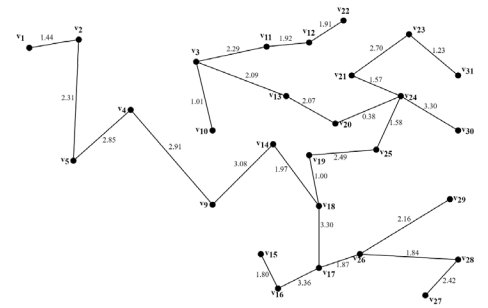


Fig. 3.29. Add edge e_{38} with weight 1.80 km.

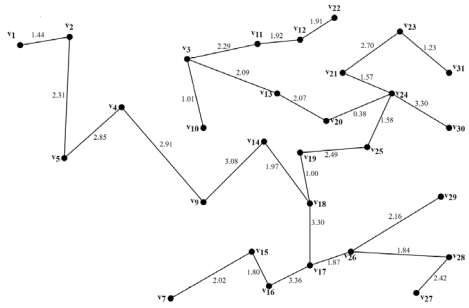


Fig. 3.30. Add edge e_{19} with weight 2.02 km.

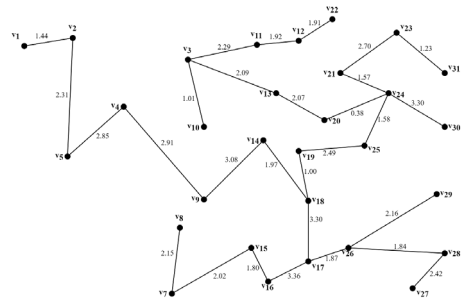


Fig. 3.31. Add edge e_{18} with weight 2.15 km.

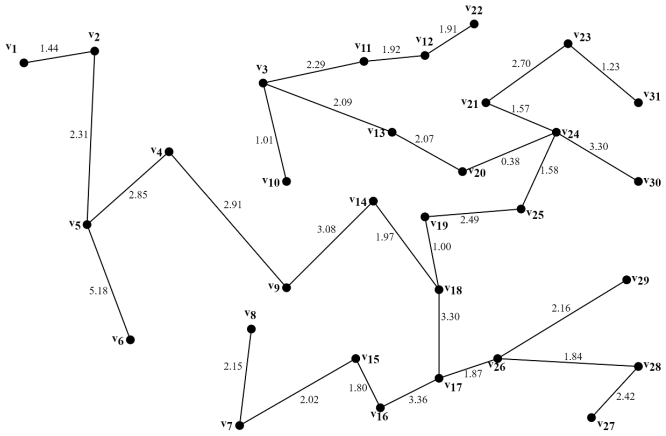


Fig. 3.31. Add edge e_{13} with weight 5.18,
the final minimum spanning tree.

Once the edge e_{13} is added, all the vertices are connected thus making the minimum spanning tree. For a graph of 31 vertices, the algorithm goes through 30 steps including the setup of the minimum weight edge of the graph.

D. Finding the Center of the Tree

As the minimum spanning tree is obtained, its center can also be identified by examining its unweighted representation. Looking at the resulting tree at hand, even though it seems as though the tree has a simple structure, finding the tree center proves to be somewhat of a challenge.

Using the second method as discussed in the previous section, start by removing outermost vertices having a degree 1 or leaves which in the first iteration would be vertices v_1, v_6. v_8, v_{10}, v_{22}, v_{27}, v_{29}, v_{30}, and v_{31}. As a result of removing these vertices, some of their parent vertices then become leaves as well. Remove these vertices as well and repeat. In the end, there should only be one or two vertices that cannot be removed as they will always have a degree not equal to 1, thus being the center of the tree.

However, trying to apply the second method leaves an ambiguous situation in which depending on which leaves are to be removed first, the center of the tree changes. This can be due to the tree's unbalanced structure in which the bulk of the tree lies northside of the branch at v_{18}. On the other hand, with intuitive thinking, it can be assumed that the center of the tree lies somewhere in vertices close to v_{19} as the tree seems to divide in that general vicinity.

Finding the center of the tree with the first method previously discussed gives a more definite answer. The longest path possible in the tree is the path from v_1 to v_{22}, passing through 15 edges. As the vertices visited is 16 which is even, therefore the center of the tree consists of the two adjacent vertices v_{19} and v_{25}.

IV. Analysis

As the tree has two center vertices, it is justifiable to assign these two office branches to be the central offices. Looking at the fact that v_{19} is the district Tegalsari and v_{25} is the district Gubeng, The Tegalsari branch can oversee operations such as incoming and outgoing mail and packages while Gubeng oversees more administrational issues. This is because while Tegalsari is in the middle of the city just like Gubeng, the roads there are more friendly for delivery needs. In contrast, Gubeng is right down the middle of the city with roads that are usually congested, therefore potentially putting a burden on delivery times and labor cost.

The Tegalsari branch can oversee the distribution of mail and packages to districts south of the tree, that is a district in which it is connected to the Tegalsari branch through the Wonokromo branch at v_{18}. The Gubeng branch the opposite as it oversees districts north of the tree that are connected to the Gubeng branch through the Tambaksari branch at v_{24}. Even though both the Tegalsari and Gubeng branch is supposedly at the center of the tree, the Tegalsari branch oversees 17 other branches while Gubeng oversees only 12 branches. This fact confirms that the tree is unbalanced, with the root represented by the Tegalsari branch having more descendants than the Gubeng branch.

With the two central offices at the Tegalsari and Gubeng branches in mind, the mail distribution route can then be made by replacing the vertex variable with the original district that it represents. The final mail distribution route is as shown in Fig. 4.1.

Reviewing back to the original dual graph used to construct the minimum spanning tree, it is apparent that distributing a package from branch A to branch B in the dual graph can be done through several different paths. On the other hand, the minimum spanning tree forces the distribution from a branch to another branch through a single path and no other.

Although the dual graph itself is already usable as a route for a mail distribution system, it is inefficient in terms of operational costs. Assuming if the overall operational cost is directly proportional to the total distance covered by the graph, the more connections there are, the more extra labor and fuel costs are needed to serve these connections. Yet, some of these connections may be unnecessary and was not needed in the first place. The total distance covered by the dual graph can be calculated by summing up all the weighted edges, which is 234.56 kilometers. Contrast that to the minimum spanning tree which has total distance 61.91 kilometers. The minimum spanning tree only needs a quarter of the distance of the dual graph to be able to connect every district in the city, thus greatly reducing operational costs.

Say, a parcel needs to be delivered from Dukuh Pakis to Sukomanunggal. In the dual graph as each districts sharing the same border is connected, the parcel can reach Sukomanunggal directly. In the minimum spanning tree however, the parcel needs to first go through Sawahan, Tegalsari, Gubeng, Genteng, Bubutan, Asemrowo and then it will finally reach Sukomanunggal. This becomes the downside of a minimum spanning tree as there may be points that are close to each other but unconnected due to the way the tree is modelled.

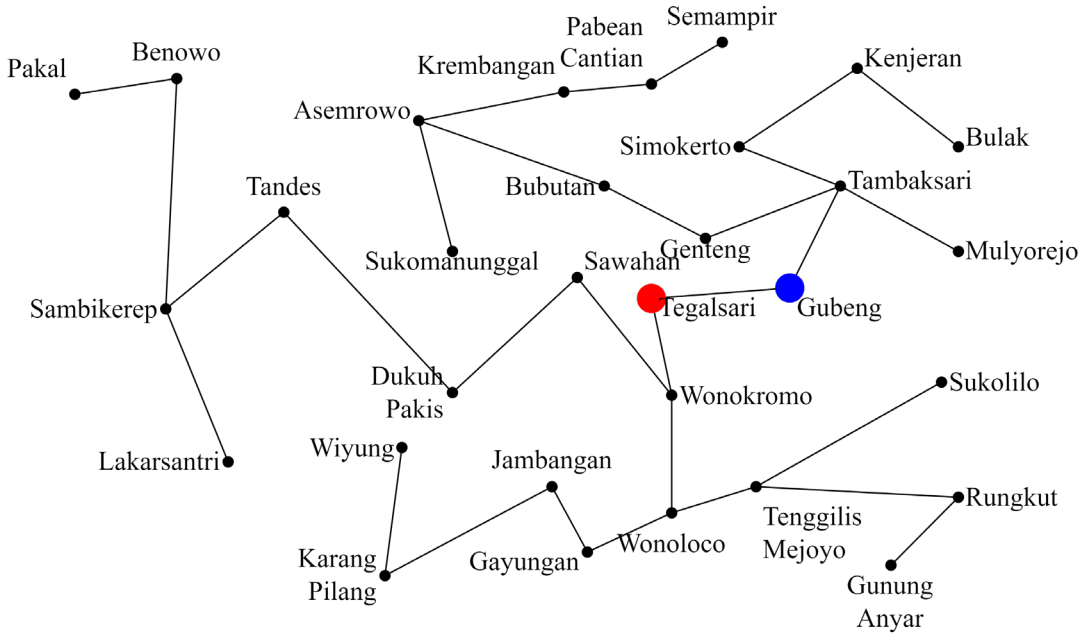


Fig. 4.1. Mail distribution route of Surabaya
(Red: operational central office, blue: administrational central office).

V. Conclusion

A mail distribution route for every district in Surabaya is successfully modelled by using a minimum spanning tree. A dual graph of the districts of Surabaya is first drawn, containing 31 vertices for each district and 70 edges for every district that share a common border. Prim's algorithm then finds the minimum spanning tree from an undirected connected, and weighted graph such as the dual graph by adding minimum weight edges one by one until all the tree's vertices are connected. From the minimum spanning tree its center can be obtained by either finding the middle vertex/vertices of the longest path or by iteratively removing the leaves off the tree. Interestingly, the mail distribution tree has two adjacent central vertices. These two central vertices are then designated as central offices which maintain the other office branches. The mail distribution route in the form of a minimum spanning tree reduces total distance needed to connect all the districts of Surabaya by three quarters when compared to its original graph.

While modelling the mail distribution route, numerous assumptions were made for the sake of simplifying the problem. The author recommends that for future studies, especially ones in analyzing an application to be used in real life scenarios, that these complexifying factors be studied and be examined at how they influence the problem.

Lastly, the author would like to address some of the difficulties faced during the writing of this paper. Constructing the dual graph of a map proves to be a challenging task, especially in this case where there are 31 districts to be checked which district shares the same border with which another district. Running through Prim's algorithm by hand is time consuming as in most cases dozens of edges needs to be checked to find which has the minimum weight. The author recommends Kruskal's algorithm instead as the edges to be added to the tree is simple chosen in ascending order, rather than painstakingly checking edges incident to vertices in the tree.

VI. Acknowledgment

The author would like to bestow the highest gratitude to Dr. Rinaldi Munir, M.T. as the coordinator and lecturer of IF2120 Discrete Mathematics along with other Discrete Mathematics lecturers for having taught all their knowledge as well as giving the author the opportunity to write this paper. The author would also like warmly thank friends and family that has played an integral role in helping the author in any way to finish this paper.

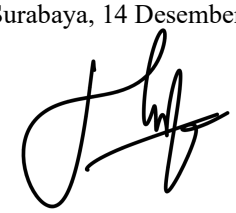
References

[1] R. Munir, *Matematika Diskrit*, 6th ed. Bandung: Informatika, 2016.
[2] K. H. Rosen, *Discrete Mathematics and Its Applications*, 7th ed. New York: McGraw-Hill, 2012.
[3] "Sur les assemblages de lignes.", *Journal für die reine und angewandte Mathematik (Crelles Journal)*, vol. 1869, no. 70, pp. 185-190, 1869.
[4] P. Hage and F. Harary, "Eccentricity and centrality in networks", *Social Networks*, vol. 17, no. 1, pp. 57-63, 1995.
[5] K. Jose, "Graph Theory | Center of a Tree", *Towards Data Science*, 2020. [Online]. Available: https://towardsdatascience.com/graph-theory-center-of-a-tree-a64b63f9415d. [Accessed: 09- Dec- 2021].
[6] S. Pettie and V. Ramachandran, "An optimal minimum spanning tree algorithm", *Journal of the ACM*, vol. 49, no. 1, pp. 16-34, 2002.
[7] R. Sedgewick and K. Wayne, *Algorithms*, 4th ed. Upper Saddle River, NJ: Addison-Wesley, 2011.
[8] A. Shahin and F. Jaferi, "The shortest route for transportation in supply chain by minimum spanning tree", *International Journal of Logistics Systems and Management*, vol. 22, no. 1, p. 43, 2015.
[9] E. Neagu, I. Vieru, A. Boroiu and M. Neagu, "The Optimization of a Transport Network Using Kruskal's Algorithm", *Scientific Bulletin, Automotive Series*, vol., no. 19, 2015.
[10] Badan Pusat Statistik Kota Surabaya, *Statistik Daerah Kota Surabaya 2021*. Surabaya, 2021.
[11] Bagian Administrasi Pemerintahan dan Otonomi Daerah Kota Surabaya, "Kecamatan dan Kelurahan se-Kota Surabaya", *Badan Pusat Statistik*, 2018. [Online]. Available: https://pemerintahan.surabaya.go.id/home/kecamatan_kelurahan. [Accessed: 11- Dec- 2021].
[12] Badan Informasi Geospasial, *Indonesia Geospatial Platform*, 2017. [Online]. Available: https://tanahair.indonesia.go.id. [Accessed: 12- Dec- 2021].

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Surabaya, 14 Desember 2021



Ahmad Alfani Handoyo 13520023