Application of Graph Theory in Ride Sharing Transportation

Abram Perdanaputra Situmorang - 13516083[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]abram.perdanaputra@gmail.com

***Abstract*— Indonesia is a developing country with 261.1 million [1] population and Jakarta as it's most populated city with approximately 10 million [2] city population. The number one problem in a high density population area is traffic jam. Efforts are made to solve this problem. One of it was a ride sharing platform to minimise the number of vehicles on the road, but ride sharing with one destination isn't good enough. Some of the ride sharing companies has released a multiple destination ride sharing option. The driver can pick up another customer while heading to another customer's destination. Otherwise, driver can drop a customer at their destination while heading to another pickup or drop point. An efficient algorithm can minimise the driver's cost. Some algorithms in graph theory such as Dijkstra's Algorithm can be used to find the shortest path from point to point.**

***Keywords*—Ride sharing, Shortest path, Dijkstra's Algorithm, Graph Theory.**

I. Introduction

In May 2014, Lyft, an online transportation company ran a simulation on their existing Lyft Classic rides (rides that aren't shared) and realized that in San Francisco, over 80% of rides shared a significant portion of their route with another ride. This was a staggering number — only a portion of the cars in the city were Lyft rides, and yet potentially the majority of those could be paired together with other passengers. This was a huge opportunity to change transportation and provide more affordable rides to our users [5].

Ridesharing is a feature that automatically pairs riders together with overlapping routes, allowing us to provide a cheaper ride for all passengers. By matching different travellers with similar itineraries in both time and their geographic locations, ride-sharing can improve driver utilization and reduce traffic congestion. This concept of pooling (or called ridesharing), has been a popular concept for decades due to its significant societal and environmental benefits. The more similarity two routes have, more likely it would be paired. But what makes a good ridesharing route? I think it's a cheap, fast, efficient, and friendly ride. And how do we know if we've matched a passenger in a route that's fast and efficient? How can we accurately predict whether two, three, or four passengers together would enjoy the route selected?

II. Theory

Graphs are discrete structures consisting of vertices and edges that connect these vertices [3]. There aren many kind of graphs, depending on the direction of the edges, loops and multiple edges, and whether it can be drawn without any intersection. Problems in almost every discipline can be solved using graph models. For example, city road systems can be modelled with graph to find the nearest distance between one point to another, a computer chip architecture can be represented as graph so it will not have any intersection.

Using graph models, we can determine the possibility to move from one street to another without doing it twice. We can determine how many colours needed to color a region without having a same colour next to each other.

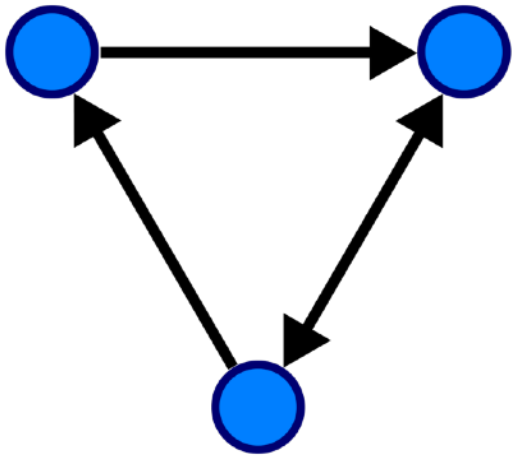
Definition :

$$G = (V, E)$$

Figure 2.1. Graph Definition

Graph is a tuple of V and E where V is a set of vertices (nodes) and E is a set of edges that connect vertices. It is important to know that E is allowed to be an empty set while V is not.

Another type of graph is a directed graph. Directed graph consists of set of vertices *V* and set of directed edges (*arcs*) *E*. Each directed edge is associated with an ordered pair of vertices. The directed edge associated with the ordered pair (u, *v*) is said to *start* at u and *end* at *v*.



Now suppose a road system weighted graph that connects *Malibu, Santa Barbara, Los Angeles, Riverside, Palm Springs, San Diego, El Cajon,* and *Barstow* where the vertices are the cities, the edges are the road that connects them, and the weight of the graph represent distances between two connected cities. We can draw a simple graph to represent this road.

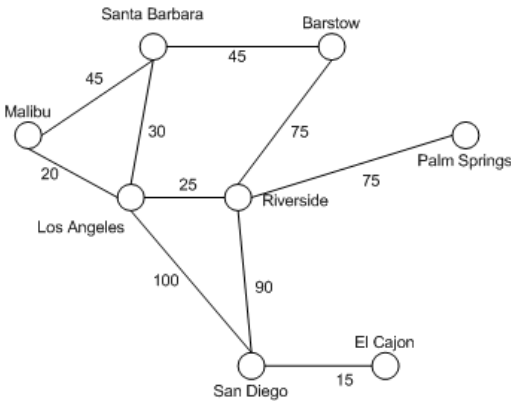


Figure 2.2. Road System Graph
Source : https://msdn.microsoft.com/

I'll introduce some of the terminology we will be using throughout this topic. Some of the basic terminology is adjacency and incidency.

Two vertices *a* and *b* in an undirected graph G are called *adjacent* (neighbours) if *a* and *b* are endpoints of an edge $e \in G$. Such an edge *e* is said to be *incident* with the vertices *u* and *v*. We will introduce an important property called degree. The *degree of a vertex* in an undirected graph is the number of edges incident with it, except a loop at a vertex contributes twice to that degree of that vertex. Degree of the vertex *v* is denoted by deg(*v*). For example vertex *Los Angeles* is adjacent with vertex *Santa Barbara, Malibu, Riverside,* and *San Diego.*

There are also some special simple graph such as complete graph, cycles, wheels, and bipartite graph. Here are some brief examples of these graph.

Complete graph is a simple graph with *n* vertices, each vertex has the degree of *n* - 1. Complete graph is denoted by K_n.

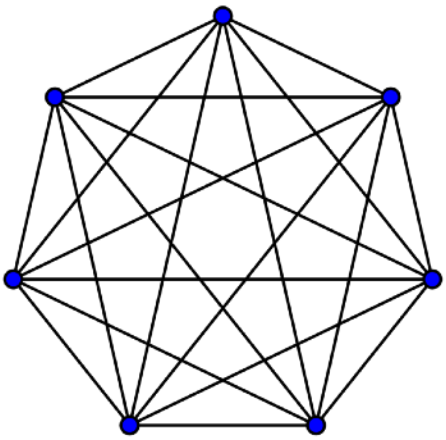


Figure 2.3. Complete graph with 7 nodes (K_7)
Source : https://en.wikipedia.org/wiki/Complete_graph

Cycle graph is a simple graph with n vertices, each vertex has the degree of 2. Cycle graph is denoted by C_n.

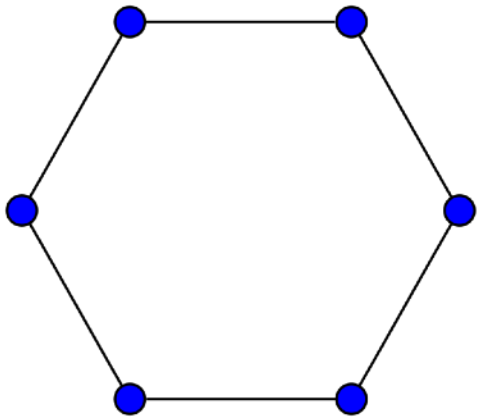


Figure 2.4. Cycle graph with 6 nodes W_6
Sourve : https://en.wikipedia.org/wiki/Cycle_graph

Wheels are cycle graph with a vertex in the middle connected to every other vertex. Wheels with vertex *n* is denoted by W_n.

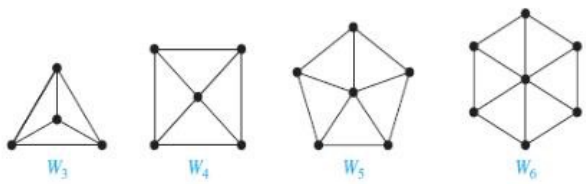


Figure 2.5. Wheel graph with 4, 5, 6, 7 nodes
Source : http://qa.geeksforgeeks.org/7461/qa.geeksforgeeks.org/7461/anyone-please-explain-about-graph-theory-especially-graphs

Bipatrite graph is a graph with set of vertex V can be partitioned into two disjoint sets V_1 and V_2 such that everty edge in the graph connects a vertex in V_1 and a vertex V_2 (so that no edge in G connects either two vertices in V_1 or two vertices in V_2.

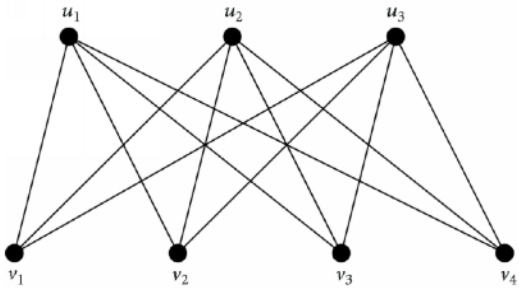


Figure 2.6. Bipartite graph $K_{3,4}$
Source : https://www.researchgate.net/figure/260946774_fig15_Complete-bipartite-graph-K34

The next important terminology is Path and Connectivity. Path is a sequence of edges that begins at a vertex of a graph

and travels from vertex to vertex along edges of the graph. An undirected graph G is said connected graph if and only if there exist a path between every pair of distinct vertices of the graph. Connectivity of a graph is important to this topic since we're analysing a road system. A road system must be a connected graph.

In graph theory, there's a term called Euler path and Euler circuit. Euler path in graph G is a simple path containing every edge of G, and Euler circuit in graph G is a simple circuit containing every edge of G. This theory is useful to determine whether we can traverse every edge in the graph exactly once and end at the starting vertex.

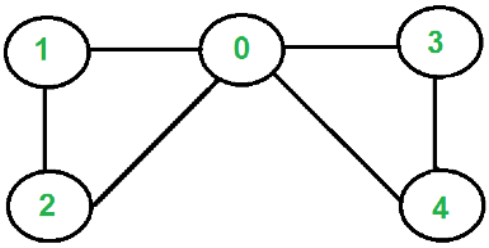


Figure 2.7. Graph with Euler circuit 0, 1, 2, 4, 3, 0
Source : http://www.geeksforgeeks.org/eulerian-path-and-circuit/

The other type of path and circuit are the Hamilton path and circuit. A Hamilton path is a simple path in graph G that passes trough every vertex exactly once, and Hamilton circuit is a simple circuit that passes trough every vertex in graph G exactly once except the first and the last vertex (Because it passed twice). This is important since this can solve problems such as whether we can pass a place exactly once in a particular planned ride sharing trip.

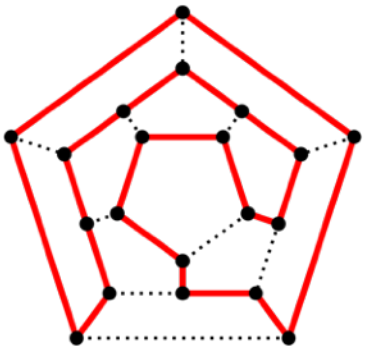


Figure 2.8. Hamiltonian circuit in a graph
Source : https://en.wikipedia.org/wiki/Hamiltonian_path

The last one is the Shortest-Path Algorithm. This algorithm was developed by Edsger Wybe Dijkstra to find a shortest path between a pair of vertices.

procedure *Dijkstra*(G: weighted connected simple graph, with all weights positive)
{G has vertices $a = v_0, v_1, \ldots, v_n = z$ and lengths $w(v_i, v_j)$ where $w(v_i, v_j) = \infty$ if $\{v_i, v_j\}$ is not an edge in G}
for $i := 1$ **to** n
  $L(v_i) := \infty$
$L(a) := 0$
$S := \emptyset$
{the labels are now initialized so that the label of a is 0 and all other labels are ∞, and S is the empty set}
while $z \notin S$
  $u :=$ a vertex not in S with $L(u)$ minimal
  $S := S \cup \{u\}$
  for all vertices v not in S
    if $L(u) + w(u, v) < L(v)$ **then** $L(v) := L(u) + w(u, v)$
    {this adds a vertex to S with minimal label and updates the labels of vertices not in S}
return $L(z)$ {$L(z) =$ length of a shortest path from a to z}

Figure 2.9. Dijkstra's algorithm
Source : Discrete Mathematics 7th Ed, K. H. Rosen

III. The Story Behind Ride Sharing

In Indonesia, some of the ride sharing companies such as GoJek and Grab had served millions of individual traveling requests by an integrated dispatching system. The drivers' locations and other states are maintained in the system such that we can simultaneously find drivers and make assignments for thousands of traveling requests. With a GPS-enabled mobile device, the users (known as passengers) can use a mobile platform app to place transportation requests from specified origin to destination [4]. At the same time, drivers who have registered with their own or rented vehicles can login to company's system through a driver app to indicate their readiness to take nearby passengers. The service is similar to a traditional taxi service in that a completed ride consists of three steps:

1. A passenger makes a booking;
2. A driver is assigned to the booking;
3. The assigned driver picks up the passenger and ferries him/her to the destination and the ride is completed.

It is common for people to arrange for manual ride-sharing with our friends traveling in the same direction to save on travel cost as well as to socialize and connect during the trip. By making use of real-time integrated ride information in the system, the company aimed to automatically match strangers traveling in similar directions and assign the same vehicle to both their journeys, allowing them to effectively car-pool. Before promoting the concept of ride sharing however, we had to verify its potential from the existing bookings. This was an example how Grab deals with the ride sharing concept.

Before we continue, there are some benefits of ride sharing concept. First, most of the online transportation companies offer discounts for ride sharing feature. Second, with using ride sharing feature, we're reducing our carbon footprint. Before, two similar trips used two different vehicle, with ride sharing feature, only one car needed to finish two or more similar trips.

The benefit wasn't only for the users. The driver got many benefit too. First, the driver can get more income since the fare got almost doubled every ridesharing trip. Second, the driver won't have to arrange the pickup and drop-off orderings, since it's managed by the companies already. Third, this will increase the income per cost ratio since the income is higher and the trip costs about the same than before.

Figure 3.1. Some implementation of ride sharing - Lyft Line
Source : https://eng.lyft.com/matchmaking-in-lyft-line-9c2635fe62c4

IV. Applications of Graph in Ride Sharing

Ride sharing is analogous to the famous Traveling Salesperson problem. A traveling salesperson wants to visit each of n places exactly once and return at the starting point. The ride sharing feature doesn't require the driver to return at the starting point, but the driver must end at the most end drop-off point. The traveling salesperson problem asks for the circuit of minimum total weight in the complete graph, because each vertex is visited exactly once in the circuit.

The most straightforward way to solve the traveling salesperson problem is to examine all possible Hamilton circuits and select one of minimum total length. Then, the ride sharing can also be solved by examining all possible Hamilton paths that ends at the drop-off point and select one of minimum total length.

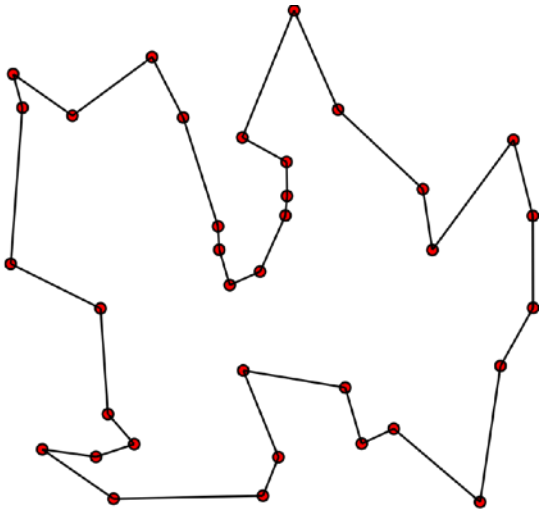


Figure 4.1. Traveling Salesperson Problem graph
Source : https://en.wikipedia.org/wiki/Travelling_salesman_problem

To solve the traveling salesperson problem, we must pick a starting point. After the starting point was selected, there are (n - 1)! Different Hamilton circuits to examine, because there are n - 1 choices for the second vertex, n - 2 coices for the third vertex, and so on. Because a Hamilton circuit can be traveled in reversed order, we need only examine (n - 1)!/2 circuits to find our answer. See that (n - 1)!/2 grows extremely rapidly.

According to our first solution of the traveling salesperson problem, we can easily conclude that in the ride sharing problem we must examine more than (n - 1)!/2 paths. Why? There could be more than one drop-off point. That made us to reconsider all drop-off point as the last destination.

The most important thing in the ride sharing sequence is the matchmaking. How does the app know who to match? What's maximum distance? How long is the detour time? Is the destination near? Every question above is something we need to build ground up. Starting point and the drop-off point of each passenger is the thing to consider. The app must determine the distance of first passenger's pickup point from the second passenger's, the distance of the drop-off point of the first passenger and the second passenger. The shortest route can be found with Dijkstra's algorithm for the map's graph.

Now, Grab only accepting two-destination ride sharing. How's the future?

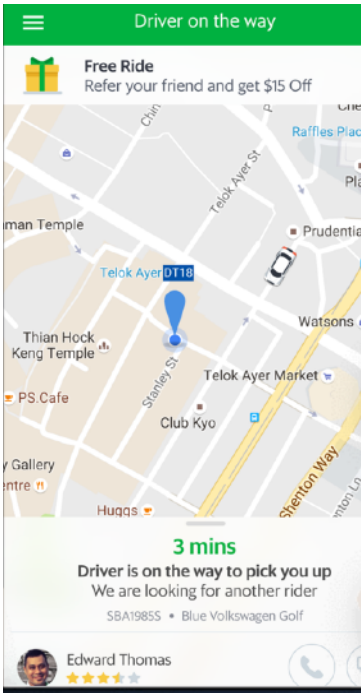


Figure 4.2. Grab application is searching for a match.
Source : https://www.grab.com/sg/share/

V. Future of Ride Sharing

Currently, most of the ride sharing transportation company only supports two-destination ride sharing. New algorithm should be developed for more efficient and more accurate matchmaking. Next, the app could always refresh and look for another matched order on the road. For example, while driving to the destination, the driver would be able to take and pickup passenger who has a similar route, so less cost and trip efficiency increased.

However, existing approaches are still limited in their complexity. For example, some ride-sharing systems require that user B be on the way for user A, and need to have all the requests submitted before they can create a route.

In contrast, the new system allows requests to be rematched to different vehicles. It can also analyze a range of different types of vehicles to determine, say, where or when a 10-person van would be of the greatest benefit.

The system works by first creating a graph of all of the requests and all of the vehicles. It then creates a second graph of all possible trip combinations, and uses a method called

"integer linear programming" to compute the best assignment of vehicles to trips.

After cars are assigned, the algorithm can then rebalance the remaining idle vehicles by sending them to higher-demand areas [6].

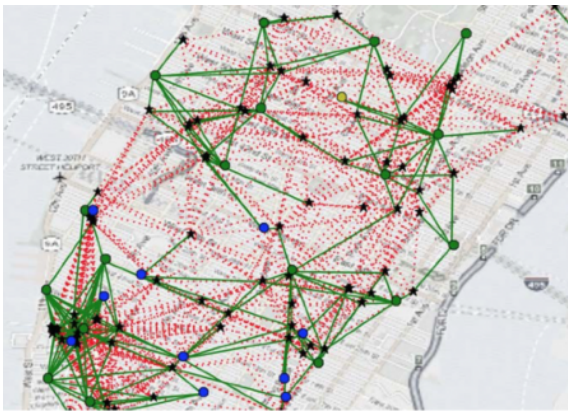
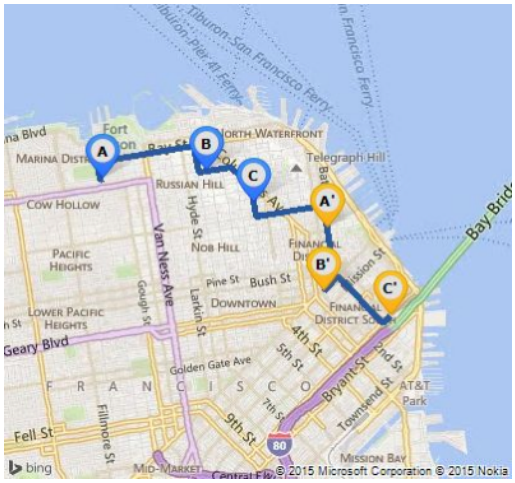


Figure 5.1. Graph modelling of orders submitted
Source : http://www.csail.mit.edu/ridesharing_reduces_traffic_300_percent

Triple matching could be one of many solution to increase the number of passenger per trip. This method are reconsidering four more permutations with two passenger on board. It will determine whether it is efficient to pickup another passenger and determine the which passenger are going to be dropped first, second, and last. This will also



determine whether the route gives good experience to the passenger or not.

Figure 5.2. Example of Triple Matching
Source : https://eng.lyft.com/matchmaking-in-lyft-line-691a1a32a008

One route that quickly became evident as a bad experience was ABCDBCDA. That's six stops between pickup and drop-off for passenger A. Even on a route with no detours, that would be a long ride given the extra time it takes for pickup and drop-offs. Additionally, though we asked users how many riders they had in their party, if any rider reported an incorrect number, drivers would have to awkwardly decide how to handle the fifth passenger. Lastly, although riding in a full car with four others is ultimately the most efficient route we can make, it doesn't always create the best experience. To adjust for these issues, more constraints were added to the system, and we focused on how to handle the level of scale that was required to support all permutations [5].

VI. Conclusion

Ride sharing feature is a breakthrough for the world of transportation. Many research states that ride sharing could reduce up to 70% of city's traffic [6]. Currently, ride sharing already contribute to reduce traffic jams in many cities such as San Francisco, Los Angeles, Singapore, and even Jakarta.

This ride sharing feature had so many uncovered potential. It still uses an inefficient greedy-style algorithm that can be made more efficient, still can't increase the safety of car-based transportation, all the heavy computing is running on the companies server, many more improvement must be made to maximize this potential.

The future may be a fully automated car that calculates it's own route, pickup and drop passengers more efficiently, and increase the level of safety in car-based transportation.

Further academic research must be made to accelerate the growth and progress in this area, so human lives may be more efficient and productive trough less time in the road and maybe no need to drive anymore because all had been automated.

I hope this will trigger more research and grow interest in learning how to make human lives more efficient and productive not only in transportation, but on all aspects of human life.

VII. Acknowledgment

The Author thanked all the Discrete Mathematics (IF 2120) lecturers :

1. Dr. Ir. Rinaldi Munir, M.T.
2. Dr. Judhi Santoso, M.Sc.
3. Dra. Harlili, M.Sc.

References

1. https://www.bps.go.id/linkTabelStatis/view/id/1274 accessed on December 1, 2017, at 11.00 AM
2. https://www.bps.go.id/linkTabelStatis/view/id/1274 accessed on December 1, 2017, at 11.00 AM
3. K. H. Rosen, *Discrete Mathematics and Its Application Global Edition*. New York: McGraw-Hill, 2013, ch. 10.
4. Tang Muchen, *The Data and Science Behind GrabShare Part I: Verifying potential and developing the algorithm*, http://engineering.grab.com/the-data-and-science-behind-grabshare-part-i, 2017.
5. Timothy Brown, *Matchmaking in Lyft Line—Part 2,* https://eng.lyft.com/matchmaking-in-lyft-line-691a1a32a008, 2016.
6. Adam Conner-Simons, Study: Carpooling Apps Could Reduce Taxi Traffic by 75%, http://www.csail.mit.edu/ridesharing_reduces_traffic_300_percent, MIT CSAIL, 2016.

Statement

This project was written by me and in my own words, except for quotations from published and unpublished sources which are clearly indicated and acknowledged as such. I am conscious that the incorporation of material from other works or a paraphrase of such material without acknowledgement will be treated as plagiarism, subject to the custom and usage of the

subject.

Bandung, 3 Desember 2017

Abram Perdanaputra - 13516083