Application of agglomerative hierarchical clustering to optimize matching problems in ridesharing for maximize total distance savings

A Nurseptiani¹, a), Y Satria¹, b), H Burhan¹, c)

¹Department of Mathematics, Faculty of Mathematics and Natural Science (FMIPA), University of Indonesia, Depok 16424, Indonesia

Corresponding author’s e-mail: a)anggun.nurseptiani@sci.ui.ac.id, b)ysatria@sci.ui.ac.id, c)helen.burhan@sci.ui.ac.id

Abstract. Ridesharing is transportation mode which brings together its participant whose similar itineraries and time schedule. The problem is how to compute matching requests with the large number of participants in short optimization time. For \( m \) driver \( n \) rider at the same time period, there is \( mn \) driver-rider combination (possible match / pm) which will be considered whether time feasibility constraint is satisfied or not, if it satisfied then the pair is feasible match (fm). This paper proposed Agglomerative Hierarchical Clustering (AHC) to be applied for determining pm. By applying AHC, the pair which is not feasible based on their location is eliminated. AHC is clustering which merge two closest cluster, recursively, until all objects merge into one cluster. After we get fm set, we will check is there any pair whose distance saving less than zero? If it is, then it will be eliminated. To decide pairs that will do ridesharing trip (optimum match) we use Hungarian Algorithm. As the result, we obtain same number of optimum matched with and without applying AHC. At the process, by applying AHC, we can reduce the number of pm, so that total calculation of the driver-rider match that will be optimized is less.

1. Introduction

Between 2000 and 2018, the number of vehicles in Indonesia has increased. Based on data from the Badan Pusat Statistik[1], the average growth rate of the number of vehicles is 13% per year. According to Rum (Suara.com, 2019), the growth in demand for transportation which is in line with the increase in the number of vehicles that are not proportional to road capacity is a cause of traffic jam, especially in Jakarta. In addition, the increase in the number of vehicles and the non-optimal use of vehicles, in other words, the low occupancy rate of vehicles, is a significant cause of traffic jam. Traffic jam is the cause of increased fuel consumption. This increase in fuel consumption occurs every year in Indonesia. This illustrates the inefficiencies of the transportation sector in Indonesia[2]. In addition, a significant factor of traffic jam is the cause of the high level of influence of the transportation sector on urban air pollution in Indonesia[3].

The alternative to reduce traffic jam is to increase road infrastructure development. However, according to Sjafruddin (2011)[2], road infrastructure development tends to encourage high levels of inefficient usage of private vehicles. An alternative way to overcome this problem is to increase the occupancy rate of vehicles by applying the ridesharing driving model. Ridesharing is defined as a ridesharing model that aims to bring travelers with the same or nearly the same location and travel time,
simultaneously, thereby saving mileage. The application of ridesharing is a sustainable transportation system because it is environmentally friendly, in the sense that it can reduce the level of fossil fuel use and reduce motor vehicle exhaust emissions. Therefore, the implementation of ridesharing in driving is one of the efforts in the transportation sector to realize the Sustainable Development Goals regarding climate change. On the other hand, the advantage that ridesharing participants get is the efficiency of travel costs because drivers or drivers can split travel costs with passengers or riders[4]. However, there are obstacles on implementing ridesharing, namely matching problem, the problem of finding the optimal driver and rider pair to obtain the maximum total distance savings[5]. The search for the optimal driver and rider pair is constrained by the high level of demand for ridesharing by drivers and riders which must be served in a very short optimization time.

An alternative solution in dealing with the high level of ridesharing demand, where the number of participants is very large and the short of optimization time, is to cluster the origin and destination points of each participant. According to Apeh and Gabrys (2006)[6], clustering is an alternative technique that promises to reduce the burden of recording all comparisons in the matching process. For this reason, this paper will discuss clustering using the agglomerative hierarchical clustering (AHC) method with the aim of classifying participants from the ridesharing system. AHC is a bottom-up clustering technique, where this algorithm combines n objects into a single cluster by paying attention to the proximity between objects. Furthermore, the single cluster is partitioned into several clusters as needed. Different from K-means clustering, in AHC the number of clusters to be formed does not affect the cluster formation process.

This paper will focus on a single driver-single rider ridesharing type, where one driver shares a ride for only one passenger request. This case refers to the problem of matching optimization between ridesharing participants, drivers and riders[7]. The optimization in this case aims to find the optimal combination of driver-rider pairs to take a ridesharing trip that results in maximum total distance savings (DS). Distance saving is the distance saving that occurs when a driver-rider pair travels a ridesharing trip, where the total mileage by a driver-rider pair during ridesharing is less than the total mileage by the driver and rider when taking individual trips[5].

On this paper, we will compare the result of matching problem optimization process with and without using AHC in three optimization time periods. The use of AHC aims to obtain a set of possible matches or pairs that will be checked for time feasible requirements and distance savings. From the set of feasible matches or the set of pairs that meet the time feasible requirements, the optimal match set will be determined. The set of optimal matches is the set containing the combination of pairs that results in the maximum total distance savings. The optimal match is selected using the Hungarian Algorithm. This driver-rider paired in optimal match set will later undertake the ridesharing trip.

2. Literature review

2.1. Ridesharing

Ridesharing is defined as a transportation mode, where drivers who have the empty seat shares a ride, also travel costs with other people who have similar destinations[8]. A ridesharing system, which aims to bring together followers who have the same or similar itinerary and time period, can provide significant social and environmental benefits by reducing the number of vehicles used for personal travel and increasing the utility of available seat capacity[4].

For each request from \( a \in D \cup R \) participant, which is element of the union of driver set \( D \) and the set of riders \( R \), there is an origin location \( \omega_a \), destination point \( \delta_a \), time announcement \( \tau(a) \), earliest departure time \( e(a) \) and the latest arrival time at destination \( l(a) \). For each participant, the time window is \([e(a), q(a)]\) where \( q(a) = l(a) - T(\omega_a, \delta_a) \) is the latest departure time of the participant and \( T(\omega_a, \delta_a) \) is the travel time from origin to destination[4]. In order to obtain a successful ridesharing, a method is needed to determine the driver-rider pair on ridesharing by considering some constraints such as time feasibility, distance savings, etc. Based on its management, ridesharing is divided into two, organized ridesharing and unorganized ridesharing. Organized ridesharing is operated by the organizer which provides ride-matching opportunities for participants who have never met each other. The
managers are service operators or matching agencies. Service operators provide ridesharing services with vehicles and drivers for a trip, whereas matching agencies only connect private drivers and rider who are traveling together[8].

There are several types of ridesharing. First, ridesharing that involves a single driver and a single rider, the problem lies in the matching of driver and rider pairs. Second, ridesharing which involves single drivers and multiple riders, focuses on routing drivers to pick up and drop off riders. Third, ridesharing which involves multiple drivers and a single rider, focuses on rider routing to move from one driver to the next. Fourth, ridesharing which involves multiple drivers and multiple riders, the problem is focused on routing drivers and riders[7]. On this paper, the type of ridesharing that will be discussed is the matching of driver and rider pairs involving a single driver and a single rider.

2.2. Agglomerative hierarchical clustering (AHC)

Agglomerative hierarchical clustering (AHC) is a clustering method that merges two objects or nearby clusters into one cluster, recursively, so that one big cluster that contains all the objects in the data is formed[3]. There are two parameters that are used to measure the proximity between two objects or clusters, namely the distance measure and the linkage method. Distance measure is used to measure the proximity between two objects or clusters which contains only two objects if it is merged. The linkage method is used to measure the proximity between two clusters which when it merged the number of objects is more than two. Output of AHC is a dendrogram which is a hierarchy diagram that summarizes the process of clustering. The following algorithm is AHC algorithm[9].

| Input: A set X of objects \{x_1, ..., x_n\}, distance function dist\(c_1, c_2)\) |
| Procedure: |
| for \(i = 1\) to \(n\) |
| \(c_1 = \{x_i\}\) |
| end for |
| \(C = \{c_1, ..., c_n\}\) |
| \(l = n + 1\) |
| while \(C\.size > 1\) do |
| \((c_{min1}, c_{min2}) = \text{minimum}\ \text{dist}(c_i, c_j)\ \text{for all} \ c_i, c_j \ \text{in} \ C\) |
| \(\text{Remove} \ c_{min1} \ \text{and} \ c_{min2} \ \text{from} \ C\) |
| \(\text{Add} \ \{c_{min1}, c_{min2}\} \ \text{to} \ C\) |
| \(l = l + 1\) |
| end while |
| Output: Dendrogram |

Figure 1. Algorithm of AHC

Carvalho, et al (2009) [10] recommend to use Ward’s linkage on geospatial data because it results clusters with similar sizes while the others result very different sizes. Since we work on geospatial data, we will use Ward’s linkage. By using Ward’s linkage, the cluster formed is a cluster that has a minimum sum of square value. We will use the most common distance measure for Ward’s linkage, Euclidean distance. Here is the Euclidean distance formula.

\[
d(x, y) = \sqrt{\sum_{i=1}^{k} (x_i - y_i)^2}
\]  

(1)

For equation (1), \(x_i\) represents variable \(i\)-th of object \(x\), \(y_i\) represents variable \(i\)-th of object \(y\), and \(k\) is the number of variables. The following equation is Ward’s linkage formula[11].
\[ d(I \cup J, K) = \sqrt{(n_i + n_K)d(I, K) + (n_j + n_K)d(J, K) - n_i d(I, J)} \]

For equation (2), \( n_i, n_j, n_K \) are the number of objects in cluster \( I, J, K \), respectively. \( d(I, K), d(J, K), d(I, J) \) are the distance between cluster \( I \) and cluster \( K \), cluster \( J \) and cluster \( K \), and cluster \( I \) and cluster \( J \).

3. Model formulation

3.1. Pre-processing

At the pre-processing stage, we need to select the possible match of driver-rider by using AHC. Let \((d, r)\) as pair of driver \(d\) and rider \(r\). \((d, r)\) is said to be possible match if it satisfies a criterion, which is their origin are belong in the same cluster and their destination are belong in the same cluster. The goal of using this method is to reduce computation of \((d, r)\) combinations that will be optimized. Also, the criteria of possible match assures the proximity of both driver’s and rider’s origin and destination. The algorithm of determining the set of possible match will be shown in figure 2.

```
Input: labeled and clustered dataset \(X\)
Initialize:
i = 1
possible_match = []
driver = []
 rider = []
While i \leq \text{len(driver in dataset } X) + 1):
  for j in range (1, len(rider in dataset } X) + 1):
    if cluster label origin driver in \(X\) = cluster label origin rider in \(X\):
      if cluster label destination driver in \(X\) = cluster label destination rider in \(X\):
        driver.append(i)
        rider.append(j)
      else:
        continue
      i = i + 1
  for i in range(len(driver)):
    possible_match.append((driver[i], rider[j]))
Output: A set of possible match
```

![Figure 2. Possible match algorithm](image)

Let \(PM\) be a set of possible match that we obtained. Next, for every \((d, r) \in PM\), it will be determined if it is a feasible match or not. The basic idea of determining a feasible match is the time window of driver and rider which must be overlap [5]. It could be known by comparing the latest time driver must depart to pick-up rider \(k_d \leq \min\{l(r) - T(\omega_r, \delta_r) - T(\omega_d, \omega_r), l(d) - T(\delta_r, \delta_d) - T(\omega_r, \delta_r) - T(\omega_d, \omega_r)\}\) to the earliest time driver could depart from its origin. If \((d, r) \in PM\) satisfies the following conditions, \(\Delta T_d = k_d - \max\{t, e(d)\} \geq 0\) and \(\Delta T_r = k_d + T(\omega_d, \omega_r) - \max\{t, e(r)\} \geq 0\), then \((d, r)\) is a feasible match, which its set will be denote by \(\tilde{P}\). \(k_d, l(r), l(d), T(\omega_r, \delta_r), T(\omega_d, \omega_r), T(\delta_r, \delta_d), t, e(d), e(r)\) respectively denotes the latest time driver must depart to pick-up rider, the latest departure time of rider \(r\), the latest departure time of driver \(d\), the travel time from rider \(r\)’s origin to its destination, the travel time from driver \(d\)’s origin to rider \(r\)’s origin, the travel time from rider \(r\)’s destination to driver \(d\)’s destination, the time when the matching problem is solved, the earliest departure time of driver \(d\), and the earliest departure time of rider \(r\).
3.2. Model formulation

The following equations are matching problem formulation [5].

\[
\text{max} \sum_{(d,r) \in P} w_{dr} x_{dr}
\]  

(3)

Subject to

\[
\sum_{r \in R \setminus \{d,r\} \in P} x_{dr} \leq 1 \quad \forall d \in D
\] 

(4)

\[
\sum_{d \in D \setminus \{d,r\} \in P} x_{dr} \leq 1 \quad \forall r \in R
\]

(5)

\[
x_{dr} \in \{0,1\} \quad \forall (d,r) \in \bar{P}
\]

(6)

\(x_{dr}\) is a variable which equals to 1 if the \((d, r)\) is an optimum match then the decision variable equals to 1, and 0 otherwise. \(w_{dr}\) denotes weight that is the contribution of pairing driver \(d\) and rider \(r\). Equation (3) represent the objective function of matching problem which maximize the sum of weight \(w_{dr}\). Equation (5) ensures the drivers paired only to one optimal rider. Equation (4) ensures the riders paired only to one optimal driver. Equation (6) is a binary decision.

The formulation above is equivalent bipartite graph matching problem. So, we can use weighted-bipartite graph to represent the relation between driver and rider, where the disjoint vertex are represent the set of drivers \(D\) and set of riders \(R\) with \(w_{dr}\) as the weight. Since the goal is to maximize the total distance savings, then \(w_{dr}\) is substituted by distance savings (DS). The following equation is the definition of DS.

\[
DS(d, r) = S_v(d, r) - S_u(d, r)
\] 

(7)

\[
S_v(d, r) = S(\omega_d, \delta_d) + S(\omega_r, \delta_r)
\]

(8)

\[
S_u(d, r) = S(\omega_d, \omega_r) + S(\delta_r, \delta_d)
\]

(9)

\(S_v(d, r)\) and \(S_u(d, r)\) represent the total distance of individual trip and the total distance ridesharing trip of driver \(d\) and rider \(r\). \(S(\omega_d, \delta_d)\), \(S(\omega_r, \delta_r)\), \(S(\omega_d, \omega_r)\), and \(S(\delta_r, \delta_d)\) respectively are the distance between origin driver \(d\) and destination driver \(d\), origin rider \(r\) and destination rider \(r\), origin driver \(d\) and origin rider \(r\), destination rider \(r\) and destination driver \(d\). For each \((d, r) \in \bar{P}\) whose \(DS < 0\) then it will be eliminate from \(\bar{P}\). On this paper, we use Hungarian algorithm to get optimum match or the combination of \((d, r) \in \bar{P}\) which result the maximum total distance savings.

4. Experimental result

We work on a dataset that was imported from google maps. The dataset contains 30 coordinate points of drivers’ origin and destination, also, 30 coordinate points of riders’ origin and destination. We implemented this program and dataset on python programming language with google collab by using device whose specification as follows, processor Intel(M) Core(TM) i5-8265U CPU @1.60GHz 1.80 GHz and Windows 10 Home Single Language 64-bit operating system. The number of time periods we used is 3 times periods. Drivers and riders whose time announcement at 07:00:01 until 07:00:50 are on the first time period. Meanwhile, the drivers and riders whose time announcement in 50 seconds after the first time period are on the second time period, and so on. By doing ridesharing, we make sure the
driver and rider will only take the additional travel time as many 40% of their total travel time when doing individual trip. So, the latest arrival time of a participant is \( l(a) = e(a) + T(\omega_a, \delta_a) + (40\% \times T(\omega_a, \delta_a)) \). We assume that 1 km could be traveled within 1 minute. Thus, by multiplying the travel distance with 111.322 and 60 seconds, we get the travel time. 111.322 is a constant number which use to convert Euclidean distance to kilometer[12].

Table 1. shows the number of participants in 3 time periods with and without AHC and its detail. As the result, we get the same total number of optimum match and the same driver-rider paired by using and without using AHC method. Note that the number of possible match for each period with using AHC is less than the number of possible match without using AHC. It means AHC is better to use because its ability to reduce the number of driver-rider combination which will be optimized. Since AHC only combine the participants which their origin or destination is close to each other.

5. Conclusion
In this paper, we discuss about the application of agglomerative hierarchical clustering to optimize matching problem in ridesharing with total distance savings maximum as the objective function. As the result, we obtain the same number of optimum matched of driver and rider with and without applying AHC. At the process, by applying AHC, we can reduce the number of possible match, so that the total calculation of the driver-rider match that will be optimized is less.

Acknowledgement
This research supported financially by Publikasi Terindeks Internasional (PUTI) research grant of Universitas Indonesia 2020.

References
[1] Badan Pusat Statistik Perkembangan Jumlah Kendaraan Bermotor Menurut Jenis, 1949-2018 (Statistics Indonesia) accessed on 20 July 2020 see https://www.bps.go.id/linkTableDinamis/view/id/1133
[2] Sjafruddin A 2011 Pembangunan Infrastruktur Transportasi untuk Menunjang Pembangunan Berkelanjutan Berbasis Ilmu Pengetahuan (Indonesian Institute of Sciences) accessed on 20 July 2020 see http://www.opi.lipi.go.id/data/1228964432/data/13086710321319703573.makalah.pdf
[3] Nurdjanah N 2015 Emisi CO2 Akibat Kendaraan Bermotor di Kota Denpasar J. Penelit. Transp. Darat 17 p. 1–14
[4] Agatz N Erera A Savelsbergh M and Wang X, 2010 Sustainable Passenger Transportation: Dynamic Ride-Sharing ERIM Report Series Reference Number ERS-2010-010-LIS
[5] Najmi A Rey D and Rashidi T H 2017 Novel dynamic formulations for real-time ride-sharing systems Transp. Res. Part E Logist. Transp. Rev. 108 November p. 122–140
[6] Apeh E T and Gabrys B 2006 Clustering for Data Matching (Berlin: Springer) vol 4251 p. 1216–1225
[7] Agatz N Erera A Savelsbergh M and Wang X 2012 Optimization for dynamic ride-sharing: A review Eur. J. Oper. Res. 223, 2 p. 295–303
[8] Furuhata M Dessouky M Ordóñez F Brunet M E Wang X and Koenig S 2013 Ridesharing: The state-of-the-art and future directions Transp. Res. Part B Methodol. 57 p. 28–46
[9] Saed S 2010 Hierarchical Clustering (Associate Professor of Professional Practice in Rutgers School of Arts and Science, New Jersey) accessed on 22 July 2020 see https://www.saedsayad.com/clustering_hierarchical.htm
[10] Carvalho AXY, Albuquerque P Junior, GRA and Guimaraes RD 2006 Spatial Hierarchical Clustering Revista Brasileira de Biometria 27(3): 411-442
[11] Müllner D 2011 Modern hierarchical agglomerative clustering algorithms (New York: Cornell University) p. 1–29
[12] Pamungkas CA 2019 Aplikasi Penghitung Jarak Koordinat Berdasarkan Latitude Dan Longitude Dengan Metode Euclidean Distance Dan Metode Haversine J. INFORMA Politeknik Indonusa Surakarta, vol. 5, 2 p. 8-13