Application of Density-Based Spatial Clustering of Application With Noise to Optimize Matching Problems in Ridesharing for Maximize Total Distance Proximity Index

I G Syafira1, a), Y Satria1, b), H Burhan1, c)
1Department of Mathematics, Faculty of Mathematics and Natural Sciences (FMIPA), University of Indonesia, Depok 16424, Indonesia

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

Abstract. Ridesharing is one of models that attempt to reduce congestion problems due to increased use of private vehicles with low occupancy. The problem related to ridesharing is to get an optimal pair of drivers and riders, while the numbers of participants involved are very large and optimization must be done in a short amount of time. In this paper DBSCAN clustering will be used as the first step to optimize the matching problem in ridesharing with DP Index as the objective function. Driver and rider with similar distance will be a good match by using DP Index as the objective function if both origin and destination of driver and rider are in a close proximity. DBSCAN clustering is one of the methods of clustering based on the density of 2-dimensional objects. In the initial stage, the DBSCAN clustering method is used to cluster the origin and destination locations of the drivers and riders. After obtaining the clusters, the driver-rider pair will be matched based on the maximum DP Index by the Hungarian algorithm. This paper uses three times periods as the result of this experiment shows that DBSCAN clustering able to increase the total number of pairs of driver-rider matching.

1. Introduction
Jakarta is the nation’s capital of Indonesia. This city also plays role as the business center and government center. As the business and government’s center, the high mobility of people and commodities is inevitable. With this high mobility of people and commodities in this city, the transportation needs are urgently needed.

According to BPS (Badan Pusat Statistik), the high mobility of people and commodities followed by the lack availability of safe and comfortable public transportations causes the use of private transport grow very rapidly every year and it is not proportional to the growth of roads [1]. Based on the transportation condition in Jakarta, average annual growth of motorized vehicles was about 9.5% in the last 5 years meanwhile the average annual growth of road length was about 0.01% [2]. The rapid growth of private transportation and low of road construction growth are the causes of traffic congestion in Jakarta [3].

Ridesharing is one of transportation models that attempt to reduce congestion problems due to increased use of private transports with low occupancy. The problem encountered in this system is to get the optimal driver-rider pair, while the huge amount of participants are involved in the system but the optimization time is very short. We propose a method to reduce the driver-rider pairs that are possible
to do ridesharing by clustering. This clustering is based from the location of origins and destination both drivers and riders. The clustering method that will be used for this paper is density-based spatial clustering of application with noise (DBSCAN).

DBSCAN clustering is one of clustering methods based on density of dataset. DBSCAN distinguish points from dataset into clusters or groups based on the dense adjacent region within the total data space then separated it into different clusters by contiguous areas of lessened density [5].

The type of ridesharing that will be used in this paper is single driver-single rider. The problem for single driver-single rider ridesharing is to optimize the matching of driver-rider pair [6]. This matching problem in ridesharing tries to find the appropriate match solution between both driver and rider that are available in ridesharing system. The optimization problem to get the driver-rider match can be represented by maximum-weight in bipartite graph [6]. The maximum total distance proximity index (DP Index) is considered in this paper to maximize the optimal matching between driver and rider. The basic idea of DP Index is if both the driver’s and rider’s origin and destination are in close proximity then the driver and rider will be a good match [7].

In this paper, Hungarian algorithm will be used to solve the matching of maximum-weighted in bipartite graph. Hungarian algorithm mostly used to solve assignment problem. In this paper Hungarian method will find maximal matchings in which this method will find matchings with the bipartite graph’s total edge weights as many as possible.

2. Literature Review

2.1. Ridesharing

Ridesharing is a model of transportation in which at least two ridesharing participants share the same vehicle to arrive at a similar destination and depart from similar origin. Ridesharing involves at least two set of participants: set of drivers that will be denoted by $D$ who drives a vehicle that still got an unoccupied capacity in it and set of riders that will be denoted by $R$ who requires a share-trip with the driver going in similar travel trip [8]. For each participant $a \in D \cup R$ has its origin $\omega_a$, destination $\delta_a$, the latest time the participant can arrive at its destination $l(a)$, time announcement or the time that participant apply to ridesharing system $t(a)$, earliest time participant can depart from its origin $e(a)$ [7]. The time window of each participant is $[e(a), q(a)]$ in which $q(a) = l(a) - T(\omega_a, \delta_a)$ where $T(\omega_a, \delta_a)$ is the travel time of each participant departs from its origin and arrive at its destination [7].

Ridesharing requires a good coordination to get a successful ridesharing because each participant of ridesharing has its own specified origin and destination, so it needs the specification of pick up point and drop off point of the rider [4]. Other coordination may rule other specified condition such as gender, travel cost, and reliability of driver and rider [4].

Drivers may offer a ride to one single-rider or maybe take multiple-riders. Riders may also similarly apply a ride with one single-driver or apply with multiple-riders by shifting from one driver to another driver to reach its own destination. A single-driver single-rider ridesharing will obtain matching between driver and rider [6]. The others will obtain the routing of single-driver multiple-riders, routing of single-rider multiple-drivers, and routing of multiple-drivers and multiple-riders [6].

2.2. Density-Based Spatial Clustering of Application with Noise (DBSCAN)

Clustering is a method to divide a given dataset into clusters that are similar lie together in one cluster [9]. Density-Based Spatial Clustering of Application with Noise is one of the clustering methods to cluster spatial data based on the density of 2-dimensional objects [10]. DBSCAN does not requisite numbers of clusters for predetermination [11]. Two parameters are needed for DBSCAN: 1) $\epsilon$ or epsilon, will be denoted by eps and 2) minimum points that will be denoted by minpts. Epsilon or eps is the radius of contiguous neighbourhood from a respected data point. Minimum points or minpts is the minimum total numbers of contiguous data points to be considered as a cluster [5]. To determine the best value for Eps, we can use the algorithm to determine suitable Eps. Figure 1 shows the algorithm to determine suitable Eps [12].
Some of terminology that are commonly used in DBSCAN algorithm are [10]:

**Epsilon neighbourhood** (ε-neighborhood) is the neighbourhood from a data point with a radius of eps. If the ε-neighborhood from the respected data point consists of total minimum numbers of adjacent points, minpts, then respected data point is considered as **core point**. If the ε-neighborhood from the respected data point does not consist of a minimum number of adjacent points, minpts, but on its ε-neighborhood contains at least one core point, then the respected point is considered as **border point**.

A point p in dataset D is considered **directly density-reachable** from other data point q if p is in ε-neighborhood of q, q considered as core point. A point p is considered as **density reachable** from other point q if there is a chain of points q₁, q₂, ..., qₙ, q₁ = p and qₙ = q such that qᵢ₊₁ is directly density-reachable from qᵢ, for 1 ≤ i ≤ n, qᵢ ∈ D. A point p is considered as **density connected** to other point q if there exist o ∈ D such that p and q are density-reachable from o.

Let a dataset D. A set C with certain eps and minpts is a non-empty subset from D is considered a **cluster** by following maximality and connectivity conditions:

∀p, q, if p ∈ C and q density reachable from p then q ∈ C (maximality condition).
∀p, q ∈ C : p density connected with q (connectivity condition).

**Noise** is a set of points in dataset D that does not include in any cluster.

The algorithm of DBSCAN will be shown in figure 2.
participants so that the matching problem will be done in a short time and ensure the participants are in the close proximity.

This paper presents a pre-processing method to reduce driver-rider pair that will be optimized based on their location and time feasibility. The driver-rider pair that will be denoted as \((d, r)\) must be considered as possible match before optimization. This possible match selection aims to reduce computation of \((d, r)\) that will be optimized. This possible match procedure ensures that both driver’s and rider’s origin and destination are in a close proximity. DBSCAN clustering will be used to determine the driver-rider pair is considered as possible match or not. A driver-rider pair is considered as possible match if both the driver and rider fulfill these following conditions: 1.) the origin of both driver and rider are belong in the same cluster. 2.) The destination of both driver and rider are belong in the same cluster. The algorithm to determine either driver-rider pair is a possible match or not will be shown in figure 3.

**Input:** labeled and clustered dataset \(D^*\)

**Initialize:**

\[ i = 1 \]

\[ \text{pm} = [] \]

\[ \text{driver} = [] \]

\[ \text{rider} = [] \]

**While** \(i \leq \text{len(driver in dataset } D^*)\)

**for** \(j \) in range \((1, \text{len(rider in dataset } D^*)+1)\):

if cluster label origin driver in \(D^*\) = cluster label origin rider in \(D^*\):

if cluster label destination driver in \(D^*\) = cluster label destination rider in \(D^*\):

\[ \text{driver.append}(i) \]

\[ \text{rider.append}(j) \]

else:

\[ \text{continue} \]

\[ j = j + 1 \]

\[ i = i + 1 \]

**for** \(i \) in range(\(\text{len(driver)}\)):

\[ \text{pm.append}((\text{driver}[i], \text{rider}[i])) \]

\[ i = i + 1 \]

**Output:** \(\text{pm}\)

**Figure 3.** Possible match algorithm

After we obtain the sets of possible match, we will determine if the \((d, r)\) that are possible match is feasible by time or not. To be considered as feasible match, the time window between driver and rider must be overlap [7]. The feasibility of \((d, r)\) could be found out by computing the latest time driver must pick-up rider \(k_d = \min[l(r) - T(\omega_r, \delta_r) - T(\omega_d, \omega_r), l(d) - T(\delta_d, \delta_r) - T(\omega_d, \omega_r)]\) then compare it to the earliest time driver could depart from its origin. The \((d, r)\) is called feasible if the the pair meets two following conditions: 1.) \(\Delta T_d = k_d - \max[t, e(d)] \geq 0\). 2.) \(\Delta T_r = k_d + T(\omega_d, \omega_r) - \max[t, e(r)] \geq 0\), by \(t\) is the optimization time is done. We will denote the set of driver-rider pairs that are feasible by \(\bar{P}\).

3.2. Model formulation

Based on previous research by Najmi et al (2017), the matching problem in ridesharing could be formulated by following equations

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

(1)

Subject to:
\[
\sum_{r \in R : (d,r) \in P} x_{dr} \leq 1 \quad \forall d \in D \\
\sum_{d \in R : (d,r) \in P} x_{dr} \leq 1 \quad \forall r \in R \\
x_{dr} \in \{0,1\} \quad \forall (d,r) \in P
\]  

Eq. (1) describes the objective function is to maximize the sum of \(w_{dr}\). Equation (2) and (3) assure that the set of drivers and set of riders that are considered as feasible match are counted in only one optimal solution. Equation (4) is a binary variable that is equal to 1 if the \((d,r)\) is matched, 0 if the \((d,r)\) is not matched.

The formulation in Equations (1)-(4) is equal to matching problem in bipartite graph. In this model, the relation between driver and rider can be expressed by bipartite graph \(G\) and the nodes of the graph represents the set of drivers \(D\) and set of riders \(R\) that are disjoint.

The weight \(w_{dr}\) that will be used in this paper is to maximize total distance proximity index (DP Index). The definition of DP Index will be shown in equation (5).

\[
DP(d,r) = \min \left( \frac{S(\omega_d, \delta_d)}{S(\omega_r, \delta_r)}, \frac{S(\omega_r, \delta_r)}{S(\omega_d, \delta_d)} \right)
\]

The idea of this DP Index objective function is if the driver and rider which are in a close proximity and have the similar distance from its origin to destination then the driver-rider pair will be a good match [7]. By using DP Index as the weight function, then the objective function from Equation (1) will turn into

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

The DP Index is used as the weight of edges that are connecting the driver’s nodes to rider’s nodes in bipartite graph. To solve the mathematical model of matching problem in ridesharing by maximizing the DP Index as the weight function, this paper uses Hungarian algorithm [13].

According to [7], by using DP Index as objective function, the result of the matching problem may disserve both driver and rider because of the big negative value of distance savings. To cover this problem, a condition is proposed by adding side constraint from mathematical formulation in (1)-(4) with form:

\[
\Delta S \geq \varepsilon
\]

While the formulation of \(\Delta S\) is \(\Delta S = S_p(d,r) - S_m(d,r)\). \(S_p(d,r)\) denotes the individual trips for drivers and riders with the formulation \(S_p(d,r) = S(\omega_d, \delta_d) + S(\omega_r, \delta_r)\). \(S_m(d,r)\) denotes the matched trips for drivers and riders with the formulation \(S_m(d,r) = S(\omega_d, \omega_r) + S(\omega_r, \delta_r) + S(\delta_r, \delta_d)\). The \(\varepsilon\) that will be used in this paper is \(\varepsilon = 0\) to ensure that both drivers and riders are saving their both distance when they are using ridesharing system.

3.3. Illustration

The data used in this paper consists of latitude and longitude of both origin and destination from each participant, the latest time the participant can arrive at its destination \(l(a)\), time announcement \(\tau(a)\), earliest time participant can depart from its origin \(e(a)\). For this illustration, the location origin and destination from 4 drivers and 4 riders will be used. Table 1 shows the example of the dataset that will be used.

| \(\tau(a)\) | \(e(a)\) | \(l(a)\) | Origin | Destination |
|---|---|---|---|---|
| Driver | 07:00:01 | 07:01:00 | 07:33:06 | 106,778204 | -6,20613 |
| (a = d) | 07:00:02 | 07:01:00 | 07:36:51 | 106,945188 | -6,2957 |

| Origin | Destination |
|---|---|
| Latitude | Longitude | Longitude | Latitude |
| 106,7853 | -6,40111 |
| 106,7363 | -6,16012 |
DBSCAN will be used in the next step to obtain the possible matches. We use DBSCAN to cluster location of driver and rider’s origin and destination. Based on the DBSCAN algorithm we obtain the labelled dataset by following picture in figure 4.

By the result from DBSCAN, we found that origin of driver 4 doesn’t belong to any cluster above, then we considered origin of driver 4 as noise which means driver 4 will not match with any riders. We proceed the possible match algorithm then we obtain the possible matches \((d, r)\) are \{\(1,1\), \(1,4\), \(2,2\), \(3,3\)\}.

This example assumed that 4 drivers and 4 riders are in the same time period. After we obtain the possible matches, the time feasibility and \(\Delta S \geq \varepsilon\) condition must be checked. The result of checking time feasibility and \(\Delta S \geq \varepsilon\) condition will be shown in table 2.

| \((d, r)\) | \(k_d\) | \(\Delta T_r\) | \(\Delta T_d\) | \(\Delta S\) |
|-----------|--------|------------|------------|--------|
| \(1,1\)   | 7:01:30 | 373.38     | 30.4013    | 15.208535 |
| \(1,4\)   | 6:49:43 | -190.70740131967 | -676.875542569 | not feasible |
| \(2,2\)   | 7:03:17 | 387.87     | 137.470    | 20.255032  |
| \(3,3\)   | 7:06:43 | 473.63     | 343.331    | 23.622392  |
Driver and rider 4 got the negative value for $\Delta T_r$ and $\Delta T_d$, so (1,4) will not be included to next step that is to find the DP Index to get the weight function. DP Index used as the weight function to determine which driver-rider pair that will be match for ridesharing by using Hungarian algorithm. Based from Hungarian algorithm with DP Index as the weight function, we got (1,1), (2,2), and (3,3) are match. We will compare the result with clustering method and without clustering method. Without clustering method, using the same example data, we match all the possible driver rider pairs. So that, we got 16 possible matches of driver-rider pair. By these 16 driver-rider pair we will check the time feasibility and $\Delta S \geq \varepsilon$ condition. The result of checking time feasibility and $\Delta S \geq \varepsilon$ condition without clustering method will be shown in table 3.

| $(d, r)$ | $k_d$ | $\Delta T_r$ | $\Delta T_d$ | $\Delta S$ |
|----------|-------|--------------|--------------|-----------|
| (1,1)    | 7:01:30 | 373.38 | 30.4013 | 15.208535 |
| (1,2)    | 6:26:05 | -1085.6014762073 | -2094.72470492 | not feasible |
| (1,3)    | 6:45:10 | 70.53986206784 | -949.468565677 | not feasible |
| (1,4)    | 6:49:43 | -190.7074131967 | -676.875542569 | not feasible |
| (2,1)    | 6:26:16 | -1076.6977891644 | -2083.05380039 | not feasible |
| (2,2)    | 7:03:17 | 387.87037425546 | 137.4709868351 | 20.255032 |
| (2,3)    | 6:25:00 | -1041.3894240199 | -2159.98095523 | not feasible |
| (2,4)    | 6:39:32 | -239.5585839108 | -1287.490644547 | not feasible |
| (3,1)    | 6:46:54 | 373.380295110 | -845.905806157 | not feasible |
| (3,2)    | 6:37:23 | -329.99277390 | -1416.348186464 | not feasible |
| (3,3)    | 7:06:43 | 473.6398316976 | 343.331426408 | 23.622392 |
| (3,4)    | 6:56:22 | 359.973756274 | -277.727882290 | not feasible |
| (4,1)    | 6:33:23 | 165.875468852 | -1656.360289464 | not feasible |
| (4,2)    | 6:39:07 | 387.87037425546 | -1312.521944358 | not feasible |
| (4,3)    | 6:21:15 | 254.505406538 | -2384.235113447 | not feasible |
| (4,4)    | 6:28:25 | 359.973756274 | -1954.6458096 | not feasible |

From table 3 we got only (1,1), (2,2), and (3,3) driver-rider pair that are feasible match. We continue the step to determine which driver-rider pair that will be match for ridesharing by using Hungarian algorithm with DP Index as the weight function. Based from Hungarian algorithm with DP Index as the weight function, we got (1,1), (2,2), and (3,3) are match.

4. Experimental Result

In this paper, python programming language is used with google colab as the coding environment on a hardware with AMD Ryzen 5 3500U processor specification with Radeon Vega Mobile Gfx (8 CPUs), 2.1GHz, memory 8192 MB and Windows 10 Home Single Language 64-bit operating system.

The experimental data used in this consists of 120 coordinates imported from google maps. These 120 coordinates consist of 30 coordinates of drivers’ origins, 30 coordinates of riders’ origins, 30 coordinates of drivers’ destinations, and 30 coordinates of riders’ destinations. The total time period used in this paper is three time periods with 50 seconds as the time steps. The first time period consists of drivers and riders with time announcement in interval 07.00.01-07.00.50 and so on for other’s time announcement with 50 seconds as the time steps. We get the latest time of participant can arrive to their destination by calculating $l(a) = e(a) + T(\omega_r, \delta_r) + (40\% \times T(\omega_d, \delta_d))$. In this system, we ensure both participants, by doing ridesharing, will only add the travel time by 30 percent of the participants’ total travel time without doing ridesharing. The driver is also assumed to be able to drive within 1 km in 1 minute then we calculate travel time of both participants by multiplying the travel distance with 60 seconds.
The following table shows the total participants involved in each three time periods, the total \((d, r)\) that is possible match in each period, the total \((d, r)\) that is feasible match in each period, and the final match \((d, r)\) that obtained from each time periods with clustering method.

| Period | Numbers of Participants | Possible match | Feasible match | Final match |
|--------|-------------------------|----------------|----------------|-------------|
|        | Driver | Rider |                  |              |              |
| 1      | 11    | 11    | 64              | 11           | 7           |
| 2      | 17    | 18    | 98              | 27           | 12          |
| 3      | 11    | 11    | 71              | 5            | 3           |

We will compare the result with clustering method and without clustering method. The following table shows the total participants involved in each three time periods, the total \((d, r)\) that is possible match in each period, the total \((d, r)\) that is feasible match in each period, and the final match \((d, r)\) that obtained from each time periods without clustering method.

| Period | Numbers of Participants | Possible match | Feasible match | Final match |
|--------|-------------------------|----------------|----------------|-------------|
|        | Driver | Rider |                  |              |              |
| 1      | 11    | 11    | 121             | 15           | 8           |
| 2      | 16    | 17    | 272             | 26           | 10          |
| 3      | 12    | 12    | 144             | 5            | 3           |

Based on table 5, we obtain the total of matched pairs is 21 meanwhile, based on table 4, we get 22 matched pairs if we use the clustering method. The clustering method also able reduce the calculation of the driver-rider pair that will be optimized based on each possible match from each time period, this means the clustering method will ensure the driver-rider pairs that will be optimized are in a close proximity so that driver does not need to go too far to pick rider up. Therefore, the results of 3 periods show that DBSCAN clustering method can solve matching problem in ridesharing better than solving matching problem in ridesharing without clustering method to get more matching pair of driver and rider.

**5. Conclusion**

This study discusses about the application of density-based spatial clustering of application with noise to optimize matching problem in ridesharing with total distance proximity index as the objective function. Based on the experimental result, we can conclude that with DBSCAN clustering we obtain more matched pairs of driver and rider than without clustering. With DBSCAN clustering, we can also reduce to total calculation of the driver-rider pair that will be optimized by reducing the possible match.

**Acknowledgement**

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

**References**

[1] Badan Pusat Statistik DKI Jakarta 2018 Statistik Transportasi DKI Jakarta 2018. DKI Jakarta : Badan Pusat Statistik.

[2] Jakarta Local Government 2011 Jakarta Urban Transport Problems and Their Environmental Impacts [PowerPoint slides] accessed on 18 August 2020 see http://www.ui.ac.id/download/apru-awi/jakarta-local-goverment.pdf

[3] Pinagara F A and Khamtanet S 2017 Will Jakarta Still Have Traffic Congestion after MRT? International Conference on Business and Management Research (ICBMR 2017). Atlantis
Press.

[4] Furuhata M, Dessouky M, Ordóñez F, Brunet M E, Wang X, and Koenig S 2013 Ridesharing: The state-of-the-art and future directions *Transportation Research Part B: Methodological* **57** p 28-46.

[5] Khan M M R, Siddique M A B, Arif R B, and Oishe M R 2018 ADBSCAN: Adaptive density-based spatial clustering of applications with noise for identifying clusters with varying densities 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT) p 107-111

[6] Agatz N, Erera A, Savelsbergh M, and Wang X 2012 Optimization for dynamic ride-sharing: A review *European Journal of Operational Research* **223** p 295-303

[7] Najmi A, Rey D, and Rashidi T H 2017 Novel dynamic formulations for real-time ride-sharing systems *Transportation research part E: logistics and transportation review* **108** p 122-140

[8] Lee A and Savelsbergh M 2015 Dynamic ridesharing: Is there a role for dedicated drivers? *Transportation Research Part B: Methodological* **81** p 483-497

[9] Gulati H and Singh P K 2015 Clustering techniques in data mining: A comparison 2015 2nd international conference on computing for sustainable global development (INDIACom) p 410-415

[10] Ester M, Kriegel H P, Sander J, and Xu X 1996 A density-based algorithm for discovering clusters in large spatial databases with noise *Kdd* vol **34** 96 p 226-231

[11] Arya R and Sikka G 2014 An Optimized Approach for Density Based Spatial Clustering Application with Noise *ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India* **1** p 695-702

[12] Elbatta M N T 2012 An improvement for DBSCAN algorithm for best results in varied densities. Dissertation (Islamic University of Gaza: Gaza)

[13] Evans J 1992 *Optimization algorithms for networks and graphs* CRC Press.