Approach for Finding Ridesharing Paths in Spatiotemporal Space

Oscar Li Jen Hsu

Institute of Information Systems and Applications
National Tsing Hua University
Hsinchu, Taiwan 30013, R.O.C.
Email: uc16@olife.org

Che-Rung Lee

Department of Computer Science
National Tsing Hua University
Hsinchu, Taiwan 30013, R.O.C.
Email: cherung@cs.nthu.edu.tw

***Abstract*—Ridesharing recommendation is an important application in urban computing. The existing grid map method is a popular method but may overlook many possible ridesharing opportunities. In this paper, we proposed an algorithm to find ridesharing paths that consist of two stages. In the first stage, GPS tracjectoreis are segmented and represented as cubes, and in the second stage, those cubes serve as landmarks for identifying possible ridesharing paths. We used the GeoLife GPS trajectories dataset to evaluate this approach and compared our algorithm with the grid map method. The results show that the number of possible ridesharing paths identified by our approach is six times that of the grid map method.**

Keywords–Urban computing; similar trajectories; ridesharing paths; GeoLife GPS Trajectories.

I. INTRODUCTION

"Ridesharing is the sharing of vehicles by passengers to reduce vehicle trips, traffic congestion and automobile emissions.", according to Wikipedia [1]. Many studies in urban computing use GPS trajectories to analyze ridesharing problems [2][3][4][5] in spatial or spatiotemporal space. These methods can find the sub-trajectories where only partial paths have ridesharing possibility while the origins and destinations of two paths are not always the same. Some methods can further find proximate trajectories where two parallel paths are not exactly on the same road but can have ridesharing relation. Most methods require the information of road network to compute ridesharing paths [5][6][7][8][9] , while another popular method proposed in [3][4] works without road network information (termed as the grid map method here) by partitioning a map into numerous blocks and identifying overlapping blocks for potential ridesharing paths. Figure 1 shows a simple example of the grid map method. The data shown in Figure 1 are the GPS temporal traces (trajectories) extracted from GeoLife dataset (from 9 am to 10 am in the downtown area of Beijing). The whole map is divided into tiny rectangular grids according to the x and y coordinates, and the number of traces passing through each grid is calculated. Any grid that has less than 10 passes is not shown on this map. While this method is effective and can successfully identify ridesharing paths without road network information, one caveate of the grid map method is that only the paths in the same block are considered as a candidate of possible ridesharing paths but the paths in the adjacent blocks were not even if the gap between two paths in adjacent blocks is just a few centimeters. Therefore, the grid map method may overlook possible candidates.

In this paper, we proposed yet another approach that does not require road network information either, with the aims to improve the searching ability by solving the abovementioned caveate in the grip map method. Our approach can find proximate sub-trajectories, includes partial ridesharing paths and the parallel paths. Inspired by computational geometry [10], our approach segments and expands trajectories into numerous cubes. Our approach finds ridesharing paths between one target trajectory and all other trajectories on both spatial and temporal space once. The evaluation was carried out by feeding the GeoLife GPS trajectories dataset into our approach as well as the grid map method. The results show that our method outperformed the grid map method in terms of the number of identified ridesharing path candidates.

Figure 1. An example of the grid map method. We used the GeoLife GPS trajectories dataset to evaluate this approach.

The rest of the paper is organized as follows. Section II gives an example of ridesharing paths. Our approach is described in Section III. The experimental evaluation to show the accuracy and compared results are in Section IV. Section V gives an example to apply our approach. Related works are explained in section VI. Finally, we conclude our study and the future works in Section VII.

II. RIDESHARING PATHS

Figure 2 shows an example of ridesharing paths and each symbol represents a GPS point. The averaged distance (spatial

 ISBN: 978-1-61208-4763

gap) between the two paths within the range of the red circle in Figure 2 is about 20 meters. Figure 3 shows the same trajectories as in Figure 2 but further including the temporal data as the third dimension. Figure 3 shows that the temporal gap between the two paths is about 1800 seconds and they have ridesharing relation.

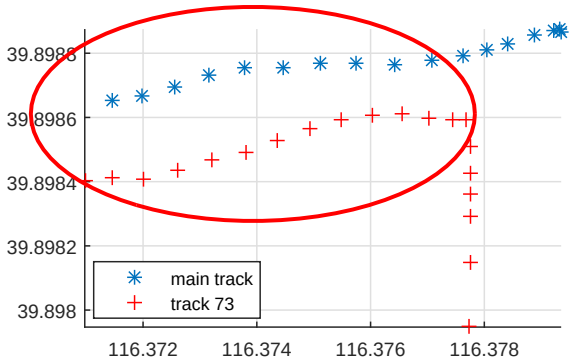

Figure 2. An example of ridesharing paths. X-axis denotes the longitude and Y-axis the latitude. The two paths (main track and track 73) are identified as having a ridesharing relationship within the range of the red circle.

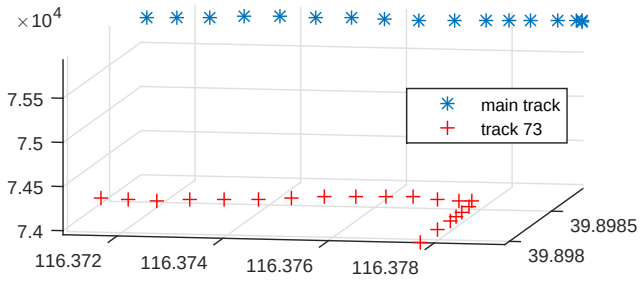

Figure 3. The same tracjectories as in Figure 2 represented in a three-dimensional plot. X-axis denotes the longitude, Y-axis the latitude, and Z-axis the temporal dimension (sec). The total time duration is 86400 seconds(24 hours).

III. Proposed Approach

This section presents our algorithm for finding the ridesharing paths, includes an algorithm for preprocessing raw data, an algorithm for cube-intersection, and a heuristic for finding the multiple ridesharing paths.

A. Preprocessing

The GPS trajectories are segmented and expanded into a series of cubes objects in this step before inputted to our cubes-intersection method. This step makes it feasible to get ridesharing paths and reduce the computational cost while a cube includes many GPS points. In Algorithm 1, the input T is a GPS trajectory, where each point $p_i = \{x_i, y_i, t_i\}$ is characterized by the longitude, latitude and timestamp. The sequence of T is in temporal order. The two parameters, ϵ and τ, are the size of a cube in the x-y plane and time domain respectively. In the beginning, a new cube object c is initiated for inserting GPS points later. Next, each GPS point of T is scanned to check whether the size of a cube c_j is bigger than the parameters, ϵ and τ when a new GPS point p_i have been inserted to c_j. The checking is conducted by (1), (2) and (3). $dist()$ converts the distance of two coordinates to meters. The new GPS point p_i will be removed from c_j if p_i lead c_j to out of the threshold ϵ and τ, and the p_i will be inserted to new c_{j+1}. The center position of the cube c_j is computed by (4) and stored as part of cube attribute. The direction of the route in a cube is computed by (5). In the final step, cubes series C will be outputted when every GPS point of T is scanned. Figure 4 shows the concept of the *cubes* visually.

$$\begin{aligned} lng\_range\,(c) &= dist\,((lat1, b1)\,, (lat1, b2)) \\ lat1 &= Latitude\,(first\_point\,(c)) \\ b1 = min\_Longitude\,(c)\,, b2 &= max\_Longitude\,(c) \end{aligned} \tag{1}$$

$$\begin{aligned} lat\_range\,(c) &= dist\,((lng1, d1)\,, (lng1, d2)) \\ lng1 &= Longitude\,(first\_point\,(c)) \\ d1 = min\_Latitude\,(c)\,, d2 &= max\_Latitude\,(c) \end{aligned} \tag{2}$$

$$\begin{aligned} time\_range\,(c) = \\ max\_Timestamp\,(c) - min\_Timestamp\,(c) \end{aligned} \tag{3}$$

$$\begin{aligned} center(c) &= (x_c, y_c, z_c) \\ x_c &= \frac{max\_Longitude\,(c) + min\_Longitude\,(c)}{2} \\ y_c &= \frac{max\_Latitude\,(c) + min\_Latitude\,(c)}{2} \\ z_c &= \frac{max\_Timestamp\,(c) + min\_Timestamp\,(c)}{2} \end{aligned} \tag{4}$$

$$\begin{aligned} direction\,(c) = arctan(\\ Latitude\,(last\_point\,(c)) - Latitude\,(first\_point\,(c))\,, \\ Longitude\,(last\_point\,(c)) - Longitude\,(first\_point\,(c))) \end{aligned} \tag{5}$$

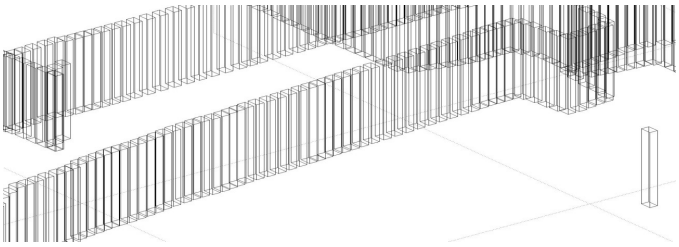

Figure 4. A conceptual plot of the cube series after Preprocessing algorithm.

B. Cubes intersection

This part is to check the intersection between any two cubes in a three dimensional space to find ridesharing paths. In the algorithm 2, the cube series C_1 represents a target trajectory and C_2 represents a combination of all other trajectories. We used a simple all pair checking between the cubes of C_1 and the cubes of C_2. Equation (6) defines a function to check the intersection. After (6), two cubes will be further checked that the direction of two cubes are similar by comparing with

Algorithm 1 Preprocessing

Input: Trajectory $T = \{p_1, p_2, \ldots, p_m\}$. Size threshold ϵ. Time threshold τ.
Output: Cubes series C

```
initial a cube c_1
let j = 1
for i = 1, 2, ..., m do
    add p_i into c_j
    if lng_range(c_j) > ε or lat_range(c_j) > ε or time_range(c_j) > τ then
        Remove p_i from c_j
        compute center position and direction of c_j.
        insert c_j into C
        j = j + 1
        initial a new cube c_j
        insert p_i into c_j
    end if
end for
compute center position and direction of c_j.
insert c_j into C
return C
```

the input parameter δ, an angle threshold. In other words, two trajectories have no ridesharing possibility if they have opposite directions. Two cubes have ridesharing relation if they had a intersection. The information of ridesharing paths will be recorded in L. The cube ID j that intersects with the cube of C_1 will be added to L where the row number is the cube ID i and the column number is the trajectory ID of c_j. The output L is a sparse matrix, whose number of rows is the number of cubes in C_1 and the number of columns is the number of trajectories in C_2.

$$isIntersect(c_i, c_j) = \begin{cases} true, & \text{if } dist((y_{c_i}, x_{c_i}), (y_{c_i}, x_{c_j})) < \epsilon \\ & \text{and } dist((y_{c_i}, x_{c_i}), (y_{c_j}, x_{c_i})) < \epsilon \\ & \text{and } |z_{c_i} - z_{c_j}| < \tau \\ false, & otherwise \end{cases} \tag{6}$$

Algorithm 2 Cubes-Intersection method

Input: cube series $C_1, C_2 = \{c_1, c_2, \ldots, c_m\}$. angle threshold δ
Output: sparse matrix L: each element can store more than one number.

```
for all c_i ∈ C_1 do
    for all c_j ∈ C_2 do
        θ_{i,j} ← get angle difference between direction(c_i) and direction(c_j).
        if isIntersect(c_i, c_j) and θ_{i,j} < δ then
            add j into L(i, track_ID(c_j) ).
        end if
    end for
end for
return L
```

C. Multiple ridesharing paths

The matrix L is already a database of multiple ridesharing paths. L contains every ridesharing path and can be seen as a one-dimensional table where each column represents one trajectory from a three-dimensional space. Each row represents a ridesharing relation between the target and other trajectories. Below is a heuristic to find the multiple ridesharing paths from L.

1) In L, the columns with the number of non-zero elements less than β shall be deleted, where β is a user-defined parameter.
2) In L, rows with only zero shall be deleted.
3) Compute Φ vector from L, whose definition is

$$\begin{aligned} L &= [r_1, r_2, \cdots, r_n]^T \\ \Phi &= [\phi(r_1), \phi(r_2), \cdots, \phi(r_n)] \\ \phi(r_x) &= \#NonZeroElements(r_x) \end{aligned} \tag{7}$$

   Simply said, Φ is a vector that each element is the number of non-zero elements in row r of L.
4) Choose a range in Φ, in which the elements are large enough.
5) Recover the GPS points from those cubes correspond to the range of rows in L, to show the real paths with ridesharing relation.

In L, We want as many non-zero columns as possible because the columns represent the trajectories have ridesharing relation; we want as long non-zero continuing rows as possible because the rows represent a long ridesharing path. A user can retrieve a short range of rows that have many columns with non-zero elements, or just one column that have the longest ridesharing path. However, it is difficult to have both at the same time. Step 1∼4 can be repeated until the range of rows and columns is small enough to find ridesharing paths. In section V, we will show an example to explain this idea.

IV. Experimental evaluation

This section shows our experimental design to investigate the questions of the performance. The structure of this section begins with the experimental purposes, followed by the overall experimental design and the experimental results.

A. Experimental Purposes

Some questions related to our approach were explored. The intention was to examine the accuracy and the ability to search ridesharing paths. Those questions were examined with Geolife dataset. The questions we explored were:

1) Will the gap between two ridesharing paths be like the ϵ we set?
2) Can our approach find more ridesharing paths than the grid map method?

Question 1 is to find out the distribution of GPS points in two intersected cubes. GPS points are possible in any position in a cube, and some GPS points may be in the corners of a cube which may lead a gap space of two ridesharing paths more than ϵ meters. Question 2 is to find out the ability of our approach to search ridesharing paths is better than the grid map method.

 ISBN: 978-1-61208-4763

B. Experimental Design

This experiments use the Geolife dataset[11] as the test data. The dataset contains 17,621 GPS trajectories with a total distance of 1,292,951 kilometers and a total duration of 50,176 hours by 182 users. Those trajectories were recorded by various GPS loggers and GPS-phones with various sampling rates. 91.5% of the trajectories are logged in a dense sampling rate, e.g. every 1∼5 seconds or every 5∼10 meters per GPS point. Each record in the dataset contains the information of latitude, longitude, altitude and time-stamp. This dataset recorded a broad range of user movements, including life routines like go home, go to work, and some entertainments and sports activities such as shopping, sightseeing, dining, hiking, and cycling. The majority of this dataset is in Beijing, China, few data is in other countries.

To reduce the variation, we only use the GPS points in Beijing city that inside 6th Ring Rd, from coordinate (39.688403, 116.091945) to (40.179632, 116.714733). The experiments were carried out on the mainframe with the specification below:

- CPU: Intel Core i7-4790 3.6GHz
- Memory: DDR3-1600 16GB Non-ECC
- OS: Windows 7 64bits
- Programming Language: C++ on Visual Studio 2013

The experiment design for question 1, Figure 5, uses two different cube sizes, (ϵ=20,τ=600) and (ϵ=100,τ=3600), to show the distribution in all pairs of intersected cubes. To check any pairs of ridesharing paths which found by our approach have a gap distance as we expect, DTW(Dynamic Time Warping) [12][13] was used to verify the gap distance between two ridesharing paths. The DTW method can find the pairwise GPS points on two trajectories. The DTW in this experiment is the dynamic programming edition, whose computational speed is much faster than the recursive edition.

The experiment design of question 2 is shown in Figure 6. To evaluate the ability of our approach to search ridesharing paths, we compare results of our approach with the grid map method. The outputs of either approach are the ridesharing information and the length of each ridesharing path was accumulated. The larger accumulated result, the better ability to find ridesharing paths. Both the block size of grid method and the cube size of our approach are 100 meters width and 3600 seconds height.

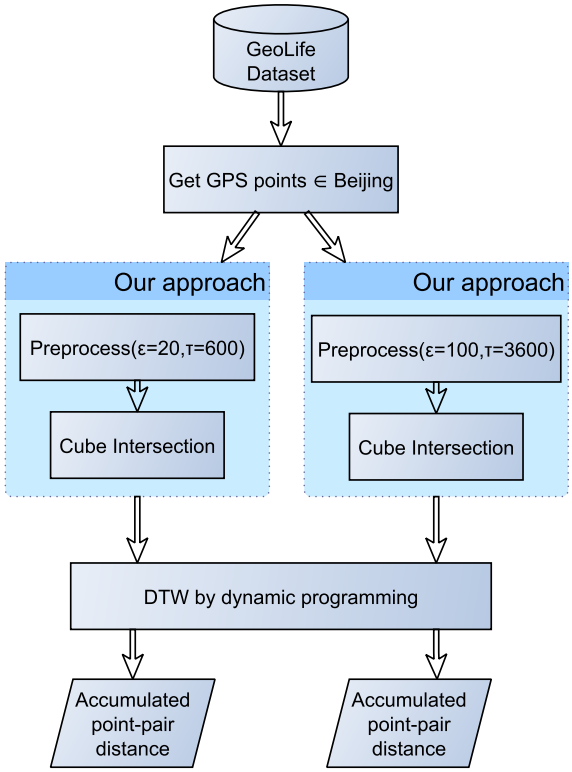

Figure 5. The design of experiment 1

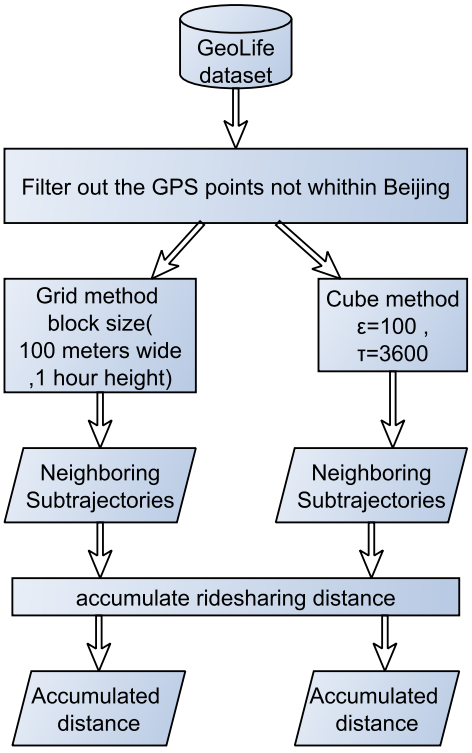

Figure 6. The design of experiment 2

C. Experimental Results

Will the gap between two ridesharing paths be like the ϵ we set? Figure 7 shows a statistic result when ϵ=20 meters. The figure shows that 92% pair-points have a gap distance less than 20 meters. The Figure 8 shows a statistic result when ϵ=100 meters. The result shows that 95% pair-points have a gap distance less than 100 meters. Both results show that the distribution in most intersected cubes is in our expectation. Most of the ridesharing paths have a gap distance less than ϵ.

Can our approach find more ridesharing paths than the grid map method?

The result in Figure 9 shows that the accumulated ridesharing path of our approach is six times that of the grid map method. The finding ridesharing ability of our approach is significantly better than the grid map method. The possible

 ISBN: 978-1-61208-4763

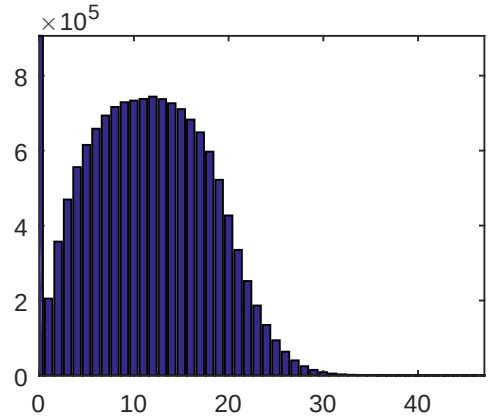

Figure 7. Accumulated results of cube size ϵ=20 and τ=600. X-axis represents the distance between the two corresponding GPS points (in meter), and the. Y-axis dentoes the counts GPS point-pairs within each bin in the X-axis.

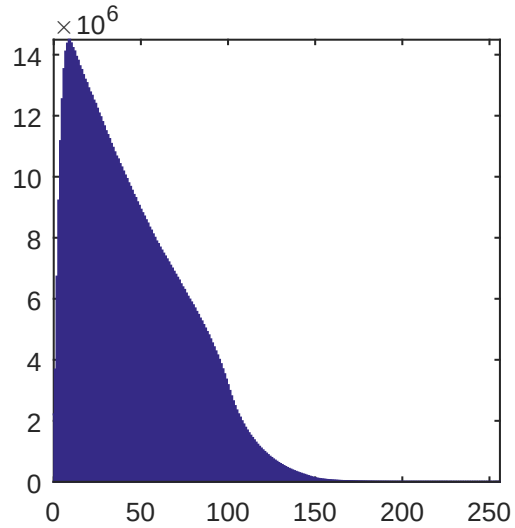

Figure 8. Accumulated results of cube size ϵ=100 and τ=3600. X-axis represents the distance between the two corresponding GPS points (in meter), and the. Y-axis dentoes the counts GPS point-pairs within each bin in the X-axis.

reason is that in the grid map method, only the paths in the same block are considered as a ridesharing path while the path in the next block will not be considered even if there is only a few centimeters gap between them. Our approach can find every ridesharing paths while the gap between two paths is smaller than ϵ. However, the computational time of the cube method is 1960 seconds, which is larger than the grid method, 152 seconds. The possible reason is that our approach needs more time to search wider range and more ridesharing distance. On the other hand, we use a huge array to represent a grid space in the grid map method for random access ability, which any pair of paths can be checked whether they are in the same block in a constant time.

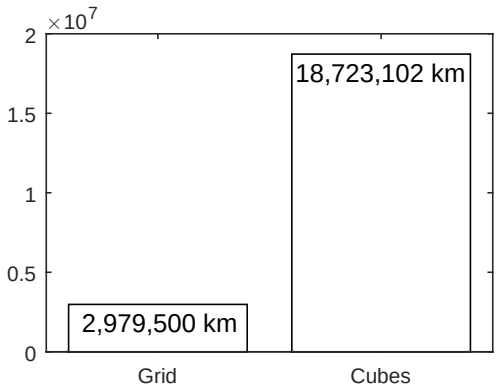

Figure 9. The accumulated ridesharing path distance

V. APPLICATION

Our approach can find ridesharing paths by comparing multiple candidate trajectories with one target trajectory. Here we demostrate an application of the proposed method, by using the trajectories data of User 0 and User 2 (about 170 trajectories for each user) in the Geo-life dataset. The task was set as a query "Compare each trajectory of User 0 to all the trajectories of User 2, and find the trajectory from User 0 that has the most ridesharing paths with User 2". Because our approach is comparing each trajectory to multiple trajectories at once, our approach is performed 170 times if there were 170 trajectories in User 0. The parameters in this example are $\beta = 10$, $\tau = 3600$ seconds, $\epsilon = 100$ meters. For example, by setting $\beta = 10$ and $\epsilon = 100$, the ridesharing paths in L with the length less than 1000 ($\beta \times \epsilon$) meters will be excluded. The results are shown on Figure 10 and Figure 11, the symbols in the graph are original GPS points with timestamps. We set the cube segments from 782 to 816 based on the values of Φ in Figure 12. These segments have most continuous ridesharing paths than other ranges of segments. The main track on the graph represents a target trajectory of User 0. Other tracks on the graph are the trajectories of User 2. Figure 10 is a two-dimensional graph showing the geographical path. Figure 11 is a three-dimensional space with temporal space, shows that there are four paths overlapped in terms of spatial distance but not in terms of temporal gap. Table I demonstrates the one-dimensional trajectories table extracted from L for the application. We found that User 0 has a GPS trajectory on 2009-04-03 that significantly overlapped with the trajectories of User 2 on 2008-11-19, 2008-12-05 and 2009-03-09.

VI. RELATED WORKS

This section introduces some works which are related to our study.

A. Computational Geometry

The rectangles intersection problem has been thoroughly investigated in computational geometry [10] . One research which uses computational geometry to find similar parts of trajectories is Buchin et al. [14] , which also takes temporal data into account. However, the method can become inefficient in practice for large sets of trajectories with many vertices.

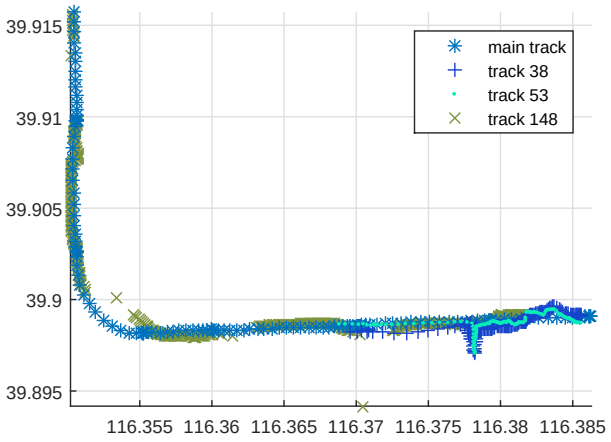

Figure 10. The application in two-dimensional plot. X-axis denotes the longitude, Y-axis the latitude.

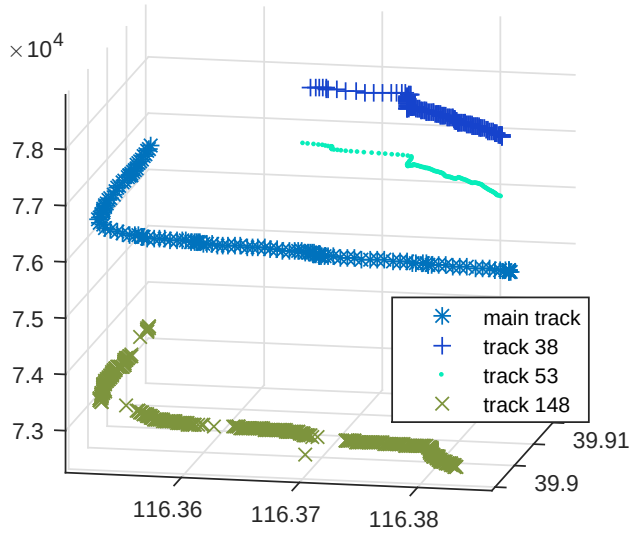

Figure 11. The application in three-dimensional plot. The total time duration is 86400 seconds(24 hours).

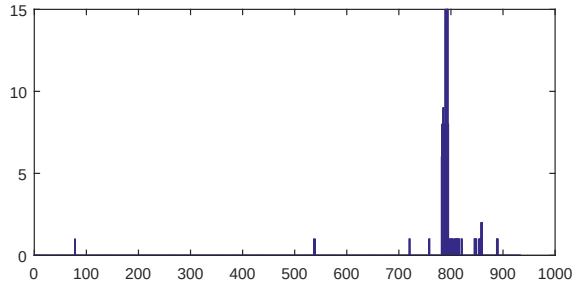

Figure 12. The values in Φ of this application. X-axis denotes the cube ID number. Y-axis denotes how many trajectories intersect with this cube.

TABLE I. ONE-DIMENSIONAL TRAJECTORIES TABLE FROM THE MATRIX L IN THIS APPLICATION.

main track	**track 38**	**track 53**	**track 148**
782	52	1	0
783	52,53	1,2	0
784	53,54	2,3	0
785	54,55	3,4	7
786	55,56	4,5	7,8
787	57	6	8,9
⋮	⋮	⋮	⋮
792	63,64	13	14
793	64,65	14	16
794	65	15	16,17
795	0	0	17,18
796	0	0	18,19
⋮	⋮	⋮	⋮
816	0	0	38

B. *Methods of finding similar trajectories*

The methods based on curve similarities, such as Longest Common Sub-Sequence LCSS [15], Edit Distance on Real sequence(EDR) [16], Hausdorff distance [17] and Dynamic Time Warping(DTW) [12][13] , are essentially not designed for ridesharing. These methods can get a value of similarity of two trajectories. However, they can find only one common sub-trajectory from two trajectories while there are multiple separated common parts. These methods also measure the similarity of the shape of two trajectories which is not required in our approach. The computational cost is high while the methods directly process GPS points without any points reduction methods. Usually, these methods only consider spatial data but spatiotemporal data [18].

C. *Graph-based methods*

Graph-based methods, such as Network Hausdorff Distance(NHD) [6][7], temporal graph-based method [8], grid with road network information method [5][9][19], the 2 synchronization points shortest paths problem [20][21], use road network information to simplify the analysis of trajectories. The road network information is like the nodes and the edges in Traveling Salesman Problem(TSP). The major issue of these methods is they will be useless while there is no road network information. Our approach processes GPS points directly but not mapping the points into POI.

D. *Ridesharing system*

The studies of the ridesharing system, such as taxi ridesharing [5], salient traffic problem [22] and the system of [3][4], show effective results. However, the taxi ridesharing and the salient traffic problem also need road network information. The system of [3][4] uses no road network information but splits user trajectories into a number of segments according to temporal distance and match the segments into grid space. The issue of the system is it may overlook possible candidates in adjacent grids as we explained in the introduction of this paper.

VII. Conclusion and future works

In this study, we designed a different approach to identify possible ridesharing paths for the ridesharing recommendation. The proposed approach can find ridesharing paths between one target trajectory and all other candidate trajectories in spatiotemporal space. The experimental results show that more than 92% ridesharing paths have a gap distance less than ϵ, which represents our cube intersection approach function well. The ability of searching ridesharing paths is significantly better than the grid map method. However, the time cost of our cube method is higher than the grid map method. The possible reason is that our searching program is not yet optimized. We expect to reduce the time cost of this method by incorporating some optimization techniques from computational geometry in future works. Based on the classification of ridesharing system in a previous study [23], our proposed method is categorized as "routing and time" class, which focuses on matching passengers and drivers by checking the pick-up and drop-off locations on the same path and the same time. Other classes such as "Origin-Destination pair" and "Keyword/List" consider only the starting and ending points but ignore the route between the two points, which is also an effective way for a ridesharing system. We leave this issue to future studies.

References

[1] Wikipedia, "Rideshare — wikipedia, the free encyclopedia," 2016, [Online; accessed 15-May-2016]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Rideshare&oldid=714643953

[2] Y. Zheng, L. Capra, O. Wolfson, and H. Yang, "Urban computing: concepts, methodologies, and applications," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 5, no. 3, 2014, p. 38.

[3] W. He, D. Li, T. Zhang, L. An, M. Guo, and G. Chen, "Mining regular routes from gps data for ridesharing recommendations," in Proceedings of the ACM SIGKDD International Workshop on Urban Computing. ACM, 2012, pp. 79–86.

[4] W. He, K. Hwang, and D. Li, "Intelligent carpool routing for urban ridesharing by mining gps trajectories," Intelligent Transportation Systems, IEEE Transactions on, vol. 15, no. 5, 2014, pp. 2286–2296.

[5] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in Data Engineering (ICDE), 2013 IEEE 29th International Conference on. IEEE, 2013, pp. 410–421.

[6] M. R. Evans, D. Oliver, S. Shekhar, and F. Harvey, "Fast and exact network trajectory similarity computation: a case-study on bicycle corridor planning," in Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing. ACM, 2013, p. 9.

[7] ——, "Summarizing trajectories into k-primary corridors: a summary of results," in Proceedings of the 20th International Conference on Advances in Geographic Information Systems. ACM, 2012, pp. 454–457.

[8] J.-R. Hwang, H.-Y. Kang, and K.-J. Li, Spatio-temporal similarity analysis between trajectories on road networks. Springer, 2005.

[9] Y. Zheng and X. Zhou, Computing with spatial trajectories. Springer Science & Business Media, 2011.

[10] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars, Computational Geometry: Algorithms and Applications. Springer-Verlag TELOS, 2008.

[11] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from gps trajectories," in Proceedings of the 18th international conference on World wide web. ACM, 2009, pp. 791–800.

[12] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," Knowledge and information systems, vol. 7, no. 3, 2005, pp. 358–386.

[13] G. A. ten Holt, M. J. Reinders, and E. A. Hendriks, "Multi-dimensional dynamic time warping for gesture recognition," in Thirteenth annual conference of the Advanced School for Computing and Imaging, June 13-15 2007.

[14] K. Buchin, M. Buchin, M. Van Kreveld, and J. Luo, "Finding long and similar parts of trajectories," Computational Geometry, vol. 44, no. 9, 2011, pp. 465–476.

[15] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in Data Engineering, 2002. Proceedings. 18th International Conference on. IEEE, 2002, pp. 673–684.

[16] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in Proceedings of the 2005 ACM SIGMOD international conference on Management of data. ACM, 2005, pp. 491–502.

[17] S. Nutanong, E. H. Jacox, and H. Samet, "An incremental hausdorff distance calculation algorithm," Proceedings of the VLDB Endowment, vol. 4, no. 8, 2011, pp. 506–517.

[18] R. Cheng, T. Emrich, H.-P. Kriegel, N. Mamoulis, M. Renz, G. Trajcevski, and A. Zufle, "Managing uncertainty in spatial and spatio-temporal data," in Data Engineering (ICDE), 2014 IEEE 30th International Conference on. IEEE, 2014, pp. 1302–1305.

[19] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," Knowledge and Data Engineering, IEEE Transactions on, vol. 27, no. 7, 2015, pp. 1782–1795.

[20] A. Bit-Monnot, C. Artigues, M.-J. Huguet, and M.-O. Killijian, "Carpooling: the 2 synchronization points shortest paths problem," in 13th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS), 2013, pp. 12–p.

[21] K. Aissat and S. Varone, "Carpooling as complement to multi-modal transportation," in Enterprise Information Systems. Springer, 2015, pp. 236–255.

[22] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, "Urban computing with taxicabs," in Proceedings of the 13th international conference on Ubiquitous computing. ACM, 2011, pp. 89–98.

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

 ISBN: 978-1-61208-4763