Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics
San Antonio, TX, USA - October 2009

A Grid-Based Clustering Method For Mining Frequent Trips From Large-Scale, Event-Based Telematics Datasets

Qing Cao, Bouchra Bouqata, Patricia D. Mackenzie, Daniel Messier, Josheph J. Salvo
Computing and Decision Sciences
GE Global Research Center
One Research Circle, Niskayuna, NY 12309

Abstract— Telematics systems that integrate wireless communications with sensor-based monitoring and location-aware applications have been widely deployed for mobile asset tracking and condition monitoring. In asset tracking field, exploring the data that relate to asset behaviors is critical to understand asset utilization, efficiency, distribution, operation, and many other important aspects in the supply chain. Prior work on analyzing GPS-based patterns has mainly been performed on time-based datasets. In this paper, we describe a scalable clustering algorithm to discover frequently repeated trips from large-scale, event-based telematics datasets collected via a satellite-based tracking system. We first transform GPS traces into a list of trips. Then we present a grid-based hierarchical clustering algorithm to discover frequent spatial patterns among all trips. We evaluate the effectiveness of the proposed algorithm against a large-scale, real-world dataset collected from tracking over a hundred of thousand assets and prove its feasibility. Through these experimental results, we show that the proposed algorithm significantly reduces the computational time needed for clustering as opposed to the traditional hierarchical clustering based on pair-wise comparison.

Keywords— Telematics, data analytics, spatial data mining, clustering algorithms, frequent patterns, large-scale datasets

I. INTRODUCTION AND RELATED WORK

Due to recent improvements and cost reductions in GPS technologies and wireless communications, telematics systems tracking moving objects using embedded GPS devices and sensors have been widely deployed in location-aware applications such as real time navigation, remote diagnosis, security monitoring, asset tracking and other services. Collecting and analyzing GPS and sensor data in these systems are vital to understanding operational efficiencies and asset behaviors, such as asset utilization, distribution and deployment.

Particularly, one important task in asset tracking is trip management, where trips are movements of the assets being tracked. In this paper, we focus on the frequent trip analysis aspect of the trip management. Frequent trips refer to similar trips that have been repeated multiple times by the same or different assets. Frequent trip analysis allows fleet managers and dispatchers to improve the accuracy of estimated time of arrival (ETA). Moreover, by correlating to external events, such as weather and traffic, it helps to detect and understand when and why deviations and anomalies occur.

However, analyzing and extracting meaningful patterns from telematics datasets are challenging due to the noisy nature of GPS sensor readings and the fuzziness of locations represented by GPS coordinates (latitude and longitude). In that effort, many existing data clustering methods have been generalized to handle GPS data and to detect GPS-based patterns of moving objects, such as clustering significant locations (places, districts or mobile communication cells), clustering the moving objects into groups, and mining periodic patterns. Trip extraction from automatically collected, telematics-based, GPS datasets has seen application in areas such as trip surveys, personal routine learning, route prediction, trip arrival time prediction, and transportation mode learning [1~4]. However, prior work on mining trips in telematics datasets has been performed mainly on time-based GPS data, in which data is collected at a predefined time frequency (e.g. 30 seconds, or 1 minute). The event-based GPS data addressed in this work are much more sparse compared to time-based GPS data. A trip can have as few as two data points indicating the trip-start event and the trip-end event, as opposed to tens or hundreds of data points in time-based GPS data scenarios. Thus, algorithms that apply to trip inference and extraction from time-based GPS data are not directly applicable to our case.

One way to cluster GPS dataset is to use the K-Means method. For example, in [1], a variant of K-Means clustering algorithm was used to extract significant locations with small radii from GPS data. After that, a time cut-off parameter was used to determine whether the locations were significant for the user or not. However, existing approaches that use K-Means require a reasonable estimation of the number of clusters k, which is difficult to obtain given the scale of multiple moving assets in our case.

This work presents a hybrid clustering algorithm that combines hierarchical method [5,6] with grid-based method [7]. Hierarchical clustering is particularly useful when it is difficult to determine the best clustering from optimizing certain score functions. The general idea is to create an initial clustering by putting all data point into disjoint set of clusters. The proximity is calculated based on the distance between cluster centroids. Then at each step, clusters that are nearest are

successively merged together, reducing the number of clusters. The iteration stops when there is no merging possible.

However, traditional hierarchical clustering is not practical when applied to a large-scale dataset such as the one in this paper. There exist improved hierarchical clustering methods, such as BIRCH[6], CURE [5] etc. The idea of BIRCH is to pre-cluster all the data and create an initial in-memory CF-tree for indexing. Since only local information stored in the CF-tree is used in clustering, therefore the time complexity is improved. CURE uses random sampling and a constant number of representative points to represent a cluster during partitioning. Both algorithms follow an approach to reduce the computation time by pre-group and index the data. In our case, since GPS datasets are geo-referenced data, naturally we choose to add grid indexing to improve the performance of the hierarchical clustering. By grid indexing data and neighborhood searching to filtering out irrelevant points, the candidate space can be significantly reduced. Grid indexing also allows us to identify neighborhood data points on the fly based on spatial relationship between grid cells, eliminating the need to store a whole data indexing reference in memory like BIRCH.

The performance of the proposed algorithm is then evaluated against a large-scale real world dataset. The dataset consists of real GPS data from over one hundred thousand assets generated over a four month time period, containing a total of 8,375,908 time-stamped location messages, representing over a million trips. Through these experimental results, we show that the proposed algorithm significantly reduces the computational time needed for clustering as opposed to the traditional hierarchical clustering based on pair-wise comparisons.

The rest of the paper is organized as follows. We introduce the application domain and review related work in Section 1. Then in Section 2 we describe the trip model and explain how we segment trips from raw GPS data streams, and define the mining frequent trip problem. In Section 3, we present the grid indexing in detail and the improved hierarchical clustering method. Section 4 shows the experimental results to demonstrate the effectiveness and efficiency of the proposed algorithm. Section 5 summarizes and concludes the paper.

II. PROBLEM DEFINITION

In telematics systems for asset tracking, a mobile asset, such as a trailer, is equipped with a tracking device that has an embedded GPS receiver as well as other sensors, and this tracking device communicates with a central data server via satellite or cellular communication. Location data, vehicle identification information, and timestamps are usually embedded in every telematics message, along with an event code. Due to engineering considerations such as power conservation in the field and limitations associated with communications cost, it is common to have assets send telematics messages from the tracking system only in an event-based messaging mode. Events represent either normal activities of an asset, such as "trip start", "trip end", "door open", "door close", "cargo loaded", and "cargo empty"; or exceptions such as "lost GPS signal", "unit low battery", etc.

In our system, each message generated by the tracking device contains one of these event codes and is annotated with geo-location and timestamp information, indicating where and when the activity or exception happened. Event messages are transmitted over the appropriate communication network, either satellite or cellular, processed by the respective communication service provider, and then received by the telematics data center. Raw messages are stored in a standard relational database from which they are available for further analysis. The system is fully automated – events at the asset level trigger the device to send messages through the network. In addition, when desired, a fleet manager can also initiate a request for information from the asset. Message receipt reliability is extremely high in these systems, >97%. Position accuracy, via GPS, is 10m, more than sufficient for identifying landmarks as start/end locations of trips.

Definition 1: A satellite message in our event-based GPS dataset is defined as $P = \{lat, lon, t, e\}$, where *lat* is latitude, *lon* is longitude, *t* is timestamp and *e* is the event code. Event codes, *e*, can be from the set {"trip start", "trip end", "door open", "door close", "cargo loaded", "cargo empty", "lost GPS signal", "low battery"}.

The GPS data stream from an asset is $\{P_1, P_2, \ldots, P_n\}$, as shown in Fig. 1.

	Latitude	Longitude	Time	Event
P_1	lat_1	lon_1	t_1	e_1
P_2	lat_2	lon_2	t_2	e_2
...	...	...	...	...
P_n	lat_n	lon_n	t_n	e_n

Fig. 1. GPS data stream.

In our system, the tracking device on a mobile asset automatically sends a "trip start" event message when the asset starts a trip, and a "trip end" event when the asset stops moving. Therefore, a basic rule for extracting trips is to represent a trip by the message sequences between consecutive "trip start" events and "trip end" events, as shown in Fig. 2.

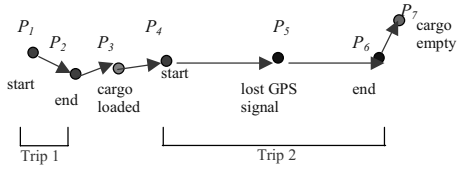

Fig. 2. Trip segments.

Definition 2: A Trip $T = \{P_i, (P_{i+1}, \ldots), P_j\}$, where P_i and P_j are the consecutive "trip start" events and "trip end" events, and $(P_{i+1,\ldots,})$ are possible intermediate non-trip-related event messages.

Intermediate messages are very important to differentiate trips. Given the sparse nature of our data, most of the trip data will have only start and end points, with no information about which route the asset takes on the trip. Locations of those intermediate messages provide additional information about a trip and can be useful to differentiate trips with different routes.

Definition 3: **(Similarity)** Two trips $T_1 = \{P_i^1, (P_{i+1}^1 ...), P_j^1\}$ and $T_2 = \{P_i^2, (P_{i+1}^2 ...), P_j^2\}$ are similar if $|(lat_i^1, lon_i^1), (lat_i^2, lon_i^2)| < d$ and $|(lat_j^1, lon_j^1), (lat_j^2, lon_j^2)| < d$, where d is the pre-determined distance criteria.

Depending on the accuracy of the GPS receiver and the physical size of the start or end location, different GPS coordinates may refer to the same location. For example, point 1 (42.3463, -71.0974), point 2 (42.3464, -71.0975), and point 3 (42.3460, -71.0976) all refer to the same location, "Fenway Park", in Boston, MA. Therefore, to determine whether the start or end locations of two trips are spatially similar, we need to use a small radius d to address the fuzziness of locations represented by GPS coordinates.

Problem Definition: Let L be the total number of assets in the study, Let T_l be the trips of an asset l, then $T = \bigcup_{l=1}^{L} T_l$ is the set of all trips of all assets in the dataset. Given the set of T trips, the mining frequent trip problem is to search for similar trips in T that are frequent and non-redundant with respect to a minimum support *min_sup*.

III. Trip Clustering Algorithm Description

In order to cluster frequent trips, we first apply a traditional hierarchical-based clustering algorithm based on pair-wise comparison of trips. In this baseline trip clustering algorithm, every trip is assigned to an initial cluster. Then iteratively, each cluster is compared with other clusters.

When the distance criterion is satisfied, the two clusters are merged and the centroid is updated. The iteration stops when there are no clusters that can be merged. Finally, the frequency of the trips in each cluster is checked to see if it meets the threshold *min_sup*.

However, this algorithm clearly follows an exhaustive pair-wise comparison of the clusters. This is not practical when applied to a large-scale dataset such as the one in this paper. Therefore, we add improvement to the baseline algorithm by adding grid indexing and neighborhood searching to reduce the time complexity of the hierarchical-based clustering. We describe in detail the improved trip clustering algorithm in next subsections.

A. Spatial Indexing

In spatial data mining literature, grid indexing has been used to achieve faster processing time for spatial accessing methods [7]. A spatial grid divides a spatial area into rectangular cells and the data points are indexed to the cells containing them.

The grid indexing can be repeated on different coarse layers, providing different resolutions of data partition. Grid indexing is particularly helpful in spatial search methods based on the Euclidian distance. Intuitively, grid indexing divides data points into grid cells based on their spatial location. Data points of distant grid cells are less likely to belong to the same cluster.

In Fig. 3, a group of spatial points (defined by longitudes and latitudes) are mapped to a grid space of rectangular cells. Each grid cell has an index number. There are many ways to define the index number; in this work, we use a simple formula to generate numerical index numbers from the grid's row and column number, as in (1).

$$Index = (row - 1) \times (number\ of\ total\ columns) + column \quad (1)$$

As an example, the cell at row 2 and column 3 would be indexed as cell number 7.

Bordering cells are the cells that share at least one vertex of the current cell. The spatial relationship between the bordering cells and the current cell could be left, right, top, down, or diagonal. The bordering cells' indices can be inferred from the spatial relationship in regard to the current cell. For example, if a grid cell is at row r and column c, the bordering cell to the right would be at column *c+1*; the bordering cell to the left is at column *c-1*; the bordering cell to the top is at row *r-1*, etc. For example, as shown in Fig. 3, cell 10 has eight bordering cells: cell 5, 6, 7, 9, 11, 13, 14, and 15. Cells on the edge of the grid space will have fewer bordering cells.

The neighborhood of a data point refers to all bordering cells in addition to its home grid cell. For example, the point marked in red in Fig. 3 is in cell 10. Its neighborhood region includes the grid cells 5, 6, 7, 9, 11, 13, 14, and 15 in addition to its home cell 10. The points in these neighboring cells are called ***neighborhood points*** of the current point.

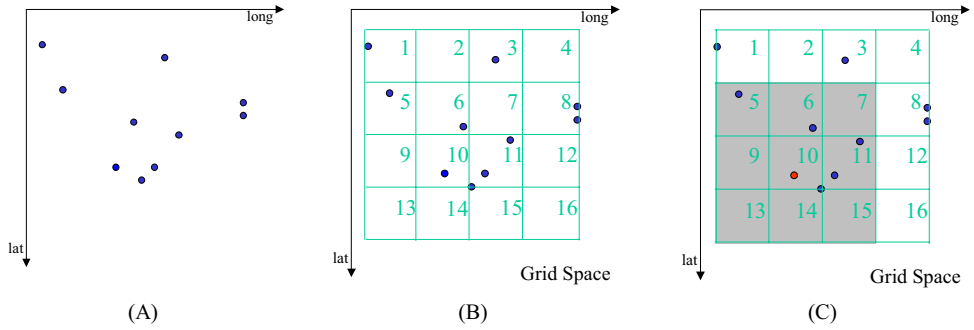

Fig. 3. A) A group of points in the latitude/longitude 2D space. B) Spatial indexing of this group of points using grids. C) Shadowed cells represent neighborhood regions of the red point in cell 10.

B. *Apply Grid Indexing to Trip Datasetss*

To apply the grid indexing to trip data, we index start and end locations into two separate grid spaces. In Fig. 4 (A), a group of trips is shown with their start and end points indexed in two different grid spaces. The start and end grid spaces are in different colors. This is to show that the start points and end points are indexed independently and all the grid indices are regional. An alternative solution is to grid index all points on the same grid space. In that case, the total number of grids needed would increase significantly as the start and end points of any trip could belong to very distant regions and may yield many empty grid cells. Furthermore, different proximity criteria for the start and end points could be used, as it is required in this work.

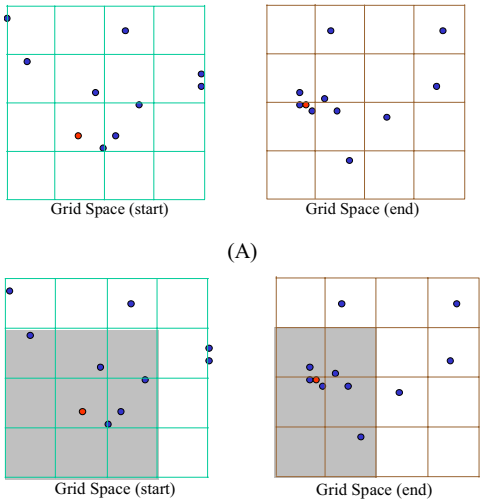

Fig. 4. (A) Indexing of the trip data into two separate grid spaces. (B) Start and end points neighborhoods of a trip are shown in shadowed cells in their respective grids.

A trip's neighborhood consists of two regions: its start point's neighborhood and its end point's neighborhood. The neighborhood query of a trip would require comparisons in both regions. In Fig. 4 (B), a trip's start and end points are highlighted in red. The neighborhood region of the highlighted trip is shown in the shadowed cells in both the start and end grid spaces.

We also applied the grid indexing to the clusters. In Fig. 5, the cluster boundaries are shown in circles. Since we adopted a centroid-based representation of a cluster, a cluster's neighborhood is its centroid's neighborhood, as shown in Fig. 5.

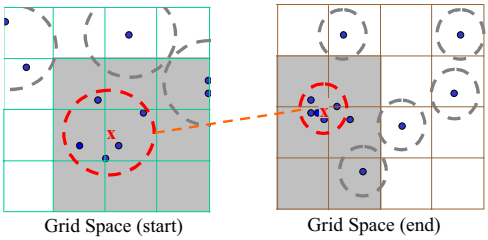

Fig. 5. The highlighted cluster's neighborhood is shown in the shadowed cells.

C. *Improved Clustering Algorithms Based on Grid Indexing*

The grid-based clustering algorithms clustering algorithm is described below. First, every trip is assigned to an initial cluster. The initial clusters are indexed into the grid cells based on their centroids. Next, when searching for clusters that may possibly be merged, instead of an exhaustive comparison with all other clusters, only neighborhood clusters are considered as potential candidates. When two clusters are merged, in addition to updating the centroid, the grid index of the resulting cluster is computed. This process is repeated until there are no clusters to merge. Finally, the frequency number of trips of each cluster is verified against the minimum support *min_sup*, and only the clusters whose frequencies are greater than *min_sup* are kept.

GridTripClustering ($\boldsymbol{T}$, *min_sup*, *d*)

1. For all trips T_i in $\boldsymbol{T}$
2. $C_k = \{ T_i \}$,
3. Let c_k be C_k 's centroid, $c_k \leftarrow T_i$
4. End
5. Grid indexing all clusters based on their centroids;
6. While there are clusters that can be merged
7. For each cluster C_k
8. $W_k \leftarrow$ Find all clusters in C_k 's neighborhood;
9. For each cluster C_k in W_k
10. If $d_{mean}(C_k, C_m) = \|c_k - c_m\| < d$
11. $C_k = C_k \cup C_m$;
12. Discard C_m;
13. End
14. Update c_k;
15. Update the grid index of C_k;
16. End
17. End
18. For each cluster C_k
19. If the number of trips in $C_k > min\_sup$
20. Keep C_k;
21. Else Discard C_k;
22. End

IV. EXPERIMENTAL RESULTS

A. Dataset Description

We extracted about millions of trips from the raw telematics GPS datasets, visualized in Fig. 6. To test our algorithm, we chose a subset of 450,523 trips that satisfy the following criteria: 1) The trips are regional trips, meaning that they start and end in the same region; and 2) the distance between the start and end of the trip has to be at least 10 miles. These trips are further sub-categorized into subsets based on the spatial regions they belong to, resulting in 50 samples, with sizes ranging from 724 to 71,749 trips per sample. These spatial regions can be considered as a rough partitioning layer prior to the grid indexing.

Fig. 6. Visualization of the telematics dataset in this study (A) raw telematics data, (B) trips extracted of an individual asset, and (C) trips extracted of multiple assets.

B. Clustering and Indexing Parameter Settings

The distance criteria used in clustering are defined as: 1-mile diameters for start data points and 0.6-mile diameters for end data points. The different distance criteria for the start and end data points are chosen because different business rules are used for defining GPS locations for "start" and "end" messaging. We chose the grid size such that only data in neighboring grid cells satisfy the distance criteria. However, the cell size should be big enough such that no good data points are filtered. The grid cell size used in grid indexing is 0.1 latitude by 0.1 longitude. This measure is about 7 by 7 in miles, which is sufficiently larger than our distance criteria. We choose the minimum support (min-sup) to be 6. In other words, a trip pattern has to be repeated for at least 6 times to be considered frequent.

C. Effectiveness

We ran both the baseline trip clustering algorithm and the grid-based agglomerative clustering algorithm on trip datasets. The grid-based clustering algorithm is equivalently effective compared to the baseline algorithm. In Table 1 we can see that the clustering result is only slightly different between the two algorithms.

TABLE I. RUNTIME AND CLUSTERING RESULT COMPARISON OF 6 SAMPLES

Sample size		724	5878	6393	12839	16177	38909
Baseline algorithm without grid indexing	***Clusters found***	9	99	122	352	499	1209
	Time (sec)	64.6	3,472	3,134	10,739	17,288	115,077
Grid TripClustering algorithm	***Clusters found***	9	101	118	348	505	1172
	Time (sec)	8.6	308.3	623.6	1,602.1	2,058.3	16,324.6

As an example, in searching for frequent trips in the first sample of 724 trips, both algorithms found the following trip pattern had been mostly repeated by different trailers in different days, as shown in Table 2. This trip pattern starts from (40.9, -75.0) and ends at (40.7, -74.1), the road distance is about 61.7 miles and has been repeated nine times in this sample.

TABLE II. A MOST FREQUENT TRIP CLUSTER FOUND BY BOTH ALGORITHMS

Trip	**Trailer**	**Start Time**	**Duration (hrs)**
1	M73	1/27/07 2:42 AM	2.27
2	B3U	1/31/07 2:39 AM	2.33
3	FW0	2/10/07 9:06 PM	2.27
4	FW0	2/22/07 1:43 AM	3.4
5	R6M	3/16/07 1:36 AM	2.32
6	2ZG	3/19/07 10:05 PM	2.75
7	9BC	3/24/07 10:23 PM	2.23
8	9UX	4/3/07 9:38 AM	2.83
9	X39	4/19/07 11:42 PM	2.85

D. Running Time Comparison

In Table 1and Fig. 7, we compare the execution time of both algorithms on 6 different samples while the grid size is set to 0.1 x 0.1 for all the samples. We see that the grid-based clustering algorithm is much faster than the original clustering algorithm.

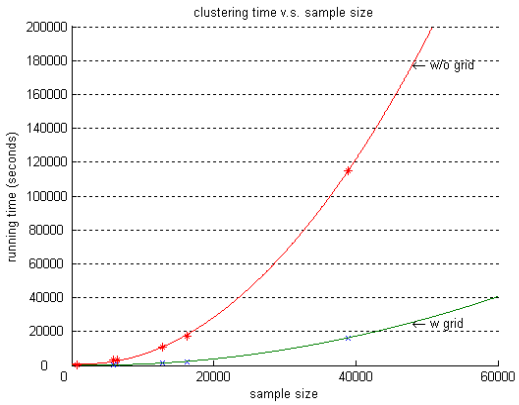

Fig. 7. Runtime comparison of the two algorithms

E. Time Complexity Analysis

Due to the exhaustive pair-wise comparison, the time complexity in each iteration of the basic agglomerative clustering algorithm is $O(n^2)$, where n is the size of the dataset.

The time complexity of the grid-based agglomerative clustering algorithm is correlated with the number of grid indices. Assuming a uniform distribution of data points, the number of data points in a grid cell is $\frac{n}{m}$, where m is the total number of grid indices. Then, the neighborhood query retrieves at most $9\times\frac{n}{m}$ data points. Consequently, the time complexity for a single iteration of the grid-based agglomerative clustering

algorithm is $O\left(\frac{n^2}{m}\right)$. Therefore, choosing *m* such that $m > \frac{n}{\log n}$, the time complexity of the algorithm is: $O\left(\frac{n^2}{m}\right) < O(n \log n)$.

F. Sensistivity to Grid Configuration

The grid cell size matters to both computation speed and accuracy. As mentioned earlier, the total number of grid cells affects the time complexity of the grid-based agglomerative clustering algorithm. Given the same grid space, when the cell size increases, neighborhood search brings more data points back to compare, yielding a more accurate clustering. However, the time complexity increases since more points are being considered for the pair-wise comparison during clustering. Ideally the cell size should be small enough such that the total number of grids.

On the other hand, when a grid gets so small, fewer points are considered for the pair-wise comparison. This is to the advantage of the time complexity but to the disadvantage of the clustering accuracy. In fact, there could be some points that are 'similar' to the current cluster but since they fall in a cell that is not a neighboring cell to the current one, they are not taken into consideration for the pair-wise comparison. A lower bound for grid cell size is $d \times d$ where *d* is the distance criteria for the trip similarity. The intuition behind this measure is that if the cell size is less than *d*, then the points that are in non-neighboring cells to the current cell but have a distance of *d* with respect to the centroid of the current cluster will not be considered as a candidate for clustering.

TABLE III. GRIDTRIPCLUSTERING ALGORITHM RUNNING TIME TO GRID SIZE

Sample size = 724	***Grid Size (Lat x Lon)***			
	0.5 x 0.5	0.25 x 0.25	0.1x0.1	0.05x 0.05
Total number of grids	16	56	320	1209
Clusters found	9	9	9	9
Running time (sec)	44.13	24.23	8.64	8.36

In Table 3, we compare the execution time on a sample of 724 trips. We see that the smaller the grid size is, the faster the grid-based clustering algorithm is. When the grid size is set to 0.1 by 0.1, the total number of grids is 320, which is more than $\frac{n}{\log n}$ (in this case, $\frac{n}{\log n} = 253$).

Data distribution will also affect the clustering result. Usually data has different density distributions over a two-dimensional grid space. Changing grid size will generate slightly different clustering results. However, if the data is uniformly distributed, then different grid sizes will not affect the clustering result.

V. CONCLUSION AND FUTURE WORK

GPS datasets collected from asset tracking systems on moving objects are challenging to analyze due to the enormous amount of data, the data quality (data is often missing and noisy) and the approximate nature of the spatial data type. In this paper, a grid-based hierarchical clustering algorithm has been proposed to discover frequent trip patterns in large-scale GPS datasets. We first indexed the trips based on a grid-indexing method, then during the hierarchical clustering process, instead of an exhaustive pair-wise comparison of all the trips; only trips sharing the same grid neighborhood were compared. The advantage of grid indexing is to significantly decrease the size of the data space needed to run the distance computation. In fact, by filtering those non-neighborhood points, the candidate solution space has been reduced to a much smaller neighborhood space. Given the appropriate grid cell size, this should significantly reduce the amount of distance computation needed for each step without sacrificing the accuracy.

We showed experimentally that the algorithm could significantly reduce the computation time while clustering accurately when working on large-scale sparse real world GPS datasets from tracking hundreds of thousands of moving objects.

One particular interest for future work is to further study the optimal grid cell size. The grid should be small enough to cut off as many irrelevant candidates as possible, yet the grid should be large enough so a true neighbor point will not fall out of the neighboring grids. An optimal grid size could play an important role in the accuracy and time complexity during clustering.

REFERENCES

[1] D. Ashbrook and T. Starner, "Learning significant locations and predicting user movement with GPS", in Proceedings of the 6th International symposium on Wearable Computers, Seattle, WA, 2002.

[2] C. Zhou, L. Terveen, and S. Shekhar, "Discovering personal paths from sparse GPS traces", in Proceedings of the 1st International Workshop on Data Mining in conjunction with 8th Joint Conference on Information Sciences, 2005.

[3] Y. Zhang, L. Liu, L. Wang, and X. Xie. Learning Transportation Mode from Raw GPS Data for Geographic Applications on the Web. In Proceedings of World Wide Web Conference 2008. Beijing, China.

[4] H. Cao, N. Mamoulis, and D. W. Cheung, "Discovery of periodic patterns in spatiotemporal sequences", IEEE Transactions on Knowledge and Data Engineering, vol. 19, 453-467, April 2007.

[5] S. Guha, R. Rastogi, and K. Shim, "CURE: an efficient clustering algorithm for large database", in Proceedings of the 1998 ACM SIGMOD international conference on Management of data, Seattle, WA, 1998, pp.73-84.

[6] T. Zhang, R. Ramakrishnan, and Livny M, "BIRCH: an effective data clustering method for very large databases", in Proceeding of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Canada, 1996, pp.103-114.

[7] W. Wang, J. Yang, and R. Muntz, "STING: a statistical information grid approach to spatial data mining", in Proceedings of the 23rd VLDB Conference, Athens, Greece, 1997, pp.186-195.