Evaluation of Clustering Techniques for GPS Phenotyping Using Mobile Sensor Data

Zachary S. Tschirhart
Karl W. Schulz
Oden Institute for Computational Engineering and Sciences
Austin, TX, United States
{zachary,karl}@oden.utexas.edu

ABSTRACT

With the ubiquitousness of mobile smart phones, health researchers are increasingly interested in leveraging these commonplace devices as data collection instruments for near real-time data to aid in remote monitoring, and to support analysis and detection of patterns associated with a variety of health-related outcomes. As such, this work focuses on the analysis of GPS data collected through an open-source mobile platform over two months in support of a larger study being undertaken to develop a digital phenotype for pregnancy using smart phone data.

An exploration of a variety of off-the-shelf clustering methods was completed to assess accuracy and runtime performance for a modest time-series of 292K non-uniform samples on the Stampede2 system at TACC. Motivated by phenotyping needs to not-only assess the physical coordinates of GPS clusters, but also the accumulated time spent at high-interest locations, two additional approaches were implemented to facilitate cluster time accumulation using a pre-processing step that was also crucial in improving clustering accuracy and scalability.

KEYWORDS

digital phenotype, GPS, clustering, time locality, machine learning, containerization

ACM Reference Format:
Zachary S. Tschirhart and Karl W. Schulz. 2020. Evaluation of Clustering Techniques for GPS Phenotyping Using Mobile Sensor Data. In *PEARC '20: ACM Symposium, July 26–30, 2020, Portland, OR.* ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

The significantly sharp increase in healthcare costs over the past decades has pushed the technological development of more efficient and effective ways of decreasing these costs through innovation. While the United States has higher health care costs than any other developed country in the world, the quality of service received is classified as marginal by almost any metric [20]. The process of using digital phenotyping to aid in individualizing heathcare offers the potential to lower costs by catering to specific trends and to avoid lumping people into generalized classes of care that risk providing the wrong support.

Digital phenotyping is the collection and processing of data with the purpose of building an individual picture of a participant to help study patterns and trends. Today, data generated passively by a single modern smartphone includes time-stamped location coordinates, accelerometer data, attitude/pressure measurements, and phone usage activity to name a few. While the larger scope of digital phenotyping is to apply the method onto any number of health trends, the purpose of this work is in support of phenotyping observations sampled throughout a woman's pregnancy.

While many sensors are available for use, a significant data stream of interest to characterize a users mobility effectively is their location as a function of time, typically inferred through a combination of GPS and cell phone signal triangulation. The location of a person over time can be used to calculate data such as user mobility, time spent traveling, and general location relative to points of interest. The location data passively generated by mobile phones is also one of the most context aware, densely packed data streams provided. Accurately processing and classifying location data in a scalable fashion is deemed a necessary step to support an ongoing research project targeting a mobile-health study of pregnancy for 1,000 women in the central Texas area [4]. Consequently, the work detailed in this paper is focused on the evaluation and development of analysis techniques for discerning spatial clusters derived from mobile sensor location data from a two month period using the Beiwe Research Platform [1][29].

Since the focus is specifically on user location data, an exploration of clustering methods provided several challenges that were ultimately overcome using a more context-sensitive approach. In order to ascertain time spent at statistically generated high-frequency locations, which the use of off-the shelf clustering methods did not provide directly, a need to relate time and space through some method manifested. General clustering methods motivated the use of pre-processing which increased the quality and scalability of the clustering of user generated location data.

2 CHALLENGES

Location data collected passively from smart phones, from a variety of hardware and operating systems, has a large variance in quality and time-to-log consistency. Determining high-frequency locations and time spent at those locations will vary based on the inaccuracy of the data and whether it is processed or filtered in the final results. Location data in general, and the specific data that was collected for this study, had sources of noises and challenges that can be categorized as follows:

- The collected sample dataset includes international and domestic GPS points, resulting in large errors in most clustering algorithms, especially when using standard normalizing methods.
- The application would stop collecting GPS data if there was not a geo-location refresh event, causing periodic gaps in data collection.
- The number of samples in a time interval varied widely, which made using the number of samples collected as a density function a challenge. The daily variation of sample frequency may be seen in Figure 1.
- The location accuracy and precision varies between mobile devices, which was characterized and captured for each sample, but the estimate of accuracy still varied and had to be taken into account.
- When calculating the distance between two location coordinates, a simple Euclidean distance metric can be used to approximate a flat earth model. Normalizing the raw coordinate data, as is typically done with clustering algorithms, is a possible source of cluster accuracy error, especially when the distance between points are large.

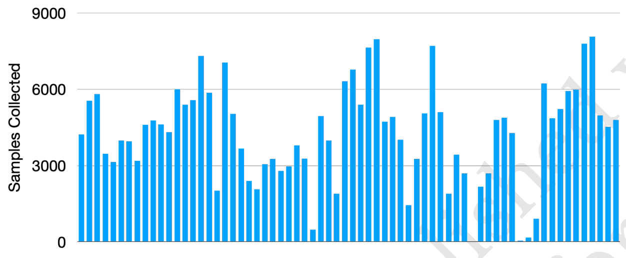

Figure 1: Location data points aggregated by day over the course of the experiment. The number of samples collected per day varies greatly.

3 EXPLORATION OF TRADITIONAL CLUSTERING METHODS

Over the course of the full study more than a thousand participants will be evaluated, and each user's 10 months worth of data will be collected with the goal of building a digital signature. The sample dataset used in this paper is a fraction of what will be processed for the full study, roughly 5000 times smaller. Therefore, choosing a scalable clustering method that used a resource-constrained model and kept accuracy errors to a minimum was a basic requirement. Accuracy evaluation, described in more detail later, must take into account properly ranked frequented locations based on how long is spent at the position. Performance and scalability were key, but the time to develop, continued maintenance, and portability were all equally evaluated and weighted. A decision was made to use containers and develop software using Jupyter notebooks for these reasons, which pushed for the use of high performance libraries developed in Python. Several off-the-shelf clustering algorithms exist in the scikit-learn package for python[24], and makes for efficient exploration and benchmarking on chosen datasets, while meeting many of the requirements listed above. In order to test performance and scalability on the relevant data patterns, a basic approach of simply feeding the location data into the clustering methods provided by scikit-learn produces cluster information immediately and provides guidance on which is likely to meet the rest of the requirements for the study. A brief description of the clustering methods used in this exploration, including their frequently cited benefits and drawbacks, are found in the following list.

Mini Batch K-Means - A variant of K-Means, with the addition of using batches of sub-sampled data to decrease the computation time and memory overhead by reducing distance between points. Using the batch scheme typically converges faster than K-Means, but at a cost of the cluster quality. Another downside to this algorithm, for our purposes, is that it requires a number of clusters to be provided beforehand. The implementation of Mini Batch K-Means in the scikit-learn package also restricts the process to a single process/thread, which may not scale.

Affinity Propagation - A method that clusters by placing points into categories and then updating the central cluster point (and the relationship others) as the algorithm progresses. The time and memory complexity of this algorithm proves to be a challenge for larger data sets, along with the restriction to a sequential process.

Mean-Shift - An iterative sliding-window centroid finding algorithm, where a calculated center point is evaluated and updated. Also, guaranteed to converge to a solution, in addition the number of iterations are typically reduced using a tolerance threshold. Although, the Mean-Shift implementation in scikit-learn also allows for multiple processes to be created which, in theory, will be scalable, but the time complexity is a significant issue to overcome.

Spectral Clustering - The method applies a K-Means clustering method to a projected normalized Laplacian after finding the restricted domain eigenvectors. Since this algorithm uses K-Means, it requires the number of clusters to be provided as input, which is a challenge. Unlike Mini Batch K-Means, the method has a large time and memory complexity, but does allow for parallel tasks be initiated.

Ward Hierarchical Clustering - Is part of a general family of algorithms that build trees of clusters by merging or splitting based on the function criteria. Specifically, Ward minimizes the sum of squares between points within the cluster. The method also does not allow for parallel tasking, which reduces an easy implementation of a scalable function.

Average Agglomerative Clustering - Like the Ward Hierarchical method, it is part of a hierarchical class of algorithms. The Average linkage minimizes the average distance between points within the cluster. Unfortunately, this method also requires that the number of clusters as input. As with Ward, the Average Agglomerative Clustering method also does not allow for parallel tasking.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) - Similar to Mean-Shift in the way that it picks a point in the data and finds the nearest neighbors, and if there are enough points to be considered a cluster, it will start to accumulate densely packed points nearby. While some DBSCAN implementations claim to have a linear memory complexity, the version contained in the version of scikit-learn used in this experiment shows strong quadratic scaling in most cases. Fortunately, the scikit-learn version of DBSCAN does have the ability to parallelize tasks.

OPTICS (Ordering Points To Identify the Clustering Structure) - A generalization of DBSCAN and instead of a full distance matrix, a reachability graph is used to order points. OPTICS is commonly recommended to be used instead of DBSCAN if the sample size is large, as the worst case memory usage is linear since the method only loads in a fraction of the data set at a time. As an aside, HDBSCAN is another alternative to OPTICS and DBSCAN, but displays similar memory complexity as DBSCAN, which is unscalable.

BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) - A Clustering Feature Tree is built by capturing characteristics of the data distribution of the clusters such as number of points in the cluster, linear sum of data points, and the square sum of all points within the cluster. A commonly sited downside to BIRCH is the computational complexity scaling with the number of clusters. BIRCH also requires an input of the number of clusters.

Gaussian Mixture Model - The scikit-learn implementation uses the expectation-maximization algorithm to fit the Gaussian Mixture Model input data set. A known issue with this method is that it diverges in some situations and requires a certain level of heuristics to properly converge. Since GMM models generalize K-means, the number of clusters are a priori knowledge. As well, GMM does not allow for easy parallelism.

3.1 Theoretical Complexity of Standard Clustering Methods

Since clustering arbitrary points is inherently NP-hard, the methods used to solve this problem are complex and performance will be based on many variables found in the context of the input data. Certain datasets will cause memory or time complexity variation, which is outside of the scope of this experiment as we focus on a single salient dataset related to future experiments. As such, a simple study and rough characterization of the theoretical complexities for these common algorithms can be found in Table 1. Note, that the average evaluation for memory and time complexity in the table is more closely related to an average over general datasets, not an average evaluated on location coordinates.

3.2 Normalization of Location Data

In many clustering algorithms, and machine learning methods in general, a normalization of the input data takes place beforehand. By using a standard scaling function, such as $z = (x - u)/s$, where u is the mean and s is the standard deviation, the data can be transformed to use standard clustering techniques applied to many general sets of data. A comparison of using raw versus normalized coordinates was performed, but a normalization method to significantly improve either accuracy or runtimes for the location data provided could not be found. All reported results are specifically created using raw coordinates and a geodesic distance measurement when available, since this provided the best results.

4 USE OF TIME LOCALIZATION

All of the methods described so far had issues producing accurate results in a timely and scalable manner, although each method was restricted to only two dimensions. An attempt to build a custom distance metric, which included time as a weighted input to replace the standard metric for each off-the-shelf clustering algorithm, never improved accuracy or computational performance. Since there was no theory found in literature review to build this monotonically increasing metric based on time and space, while also accounting for challenges listed in section 2, a heuristic pre-processing approach was pursued. Two methods were developed to associate time and space in the context of discrete location data collection. Both methods improved scalability, but section 4.2 covers the algorithm that also increased accuracy on general location datasets drastically. The Intertwined Time and Space pre-Processing method also provided an excellent estimate of accumulated time, without additional computation. The estimate on time spent at a particular location provides an extra level of stability for the proper ranking of frequently visited location.

4.1 Hierarchical Segmentation

Using the assumption that people will follow a daily cycle of movement, one could group samples together by day or week and cluster on that time period. Then using weighted centroids based on number of samples per day or week, an end level clustering algorithm would be used to find clusters on the pre-processed data while accumulating the number of samples and centroids of centroids into the final results.

```
procedure HierarchicalSegmentation
main:
    grouped locations ← SplitByTime(locations)
    goto loop.
    labels ← Cluster(accumulated centroids)
    a ← accumulated centroids
    b ← accumulated samples
    x, y ← CalcCentroids(label, a, b)
    close;
loop:
    labels ← Cluster(grouped locations(i))
    x, y ← CalcCentroids(labels, grouped locations(i))
    accumulated centroids ← append(x)
    accumulated samples ← append(y)
    i ← i + 1
    if i != EOF then return
    else goto loop.
```

4.2 Intertwined Time and Space Pre-processing

Intertwined Time and Space (ITS) pre-processing was approached from the idea of perfect data. If location data collection over time was continuous and without error, a simple difference over any portion of the time parameterized line function would provide the total time for that segment. Since the function provides a surjective map from time onto location, one could also integrate over a restricted space domain to find how much time was spent within that region. Using this theoretical concept, with the addition of realistic

Table 1: Clustering Methods

Clustering Method	Memory Complexity (Worst)	Memory Complexity (Average)	Compute Complexity (Worst)	Compute Complexity (Average)
Mini Batch K-Means[10]	$O(const)$	$O(const)$	$O(K*N*I)$	$O(K*N*I)$
Affinity Propagation[14]	$O(N^2)$	$O(N^2)$	$O(N^2*I)$	$O(N^2*I)$
Mean-Shift [33]	$O(const)$	$O(const)$	$O(N^2*I)$	$O(N^2*I)$
Spectral Clustering[32][16]	$O(N^2)$	$O(N*M)$	$O(N^3)$	$O(N^3)$
Ward Hierarchical Clustering[36]	$O(N^2)$	$O(N^2)$	$O(N^3)$	$O(N^2)$
Agglomerative Clustering[36]	$O(N^2)$	$O(N^2)$	$O(N^3)$	$O(N^2)$
DBSCAN [8] [27]	$O(N^2)$	$O(N*D)$	$O(N^2)$	$O(N*log(N))$
OPTICS [23]	$O(N)$	$O(N)$	$O(N^2)$	$O(N*log(N))$
BIRCH[36][35]	$O(const)$	$O(const)$	$O(N*K)$	$O(N*K)$
Gaussian Mixtures[17][19]	$O(N^2)$	$O(N*G)$	$O(N*G*I)$	$O(N*G)$

1. D represents the average number of neighbors, K represents the number of clusters, I represents the number of iterations, M represents the subset of sampled columns, G represents the number of Gaussian bases.
2. The Mini batch version of K-Means will load a constant subset of data for each batch.
3. The BIRCH algorithm can work with an arbitrary size of memory (within limit), although run time may be affected for smaller memory.

heuristics to reduce incorrect data accumulation from imperfect sources, the ITS pre-process procedure achieves scalable and sophisticated clustering accuracy. Since the pre-processing method also calculates the time difference for each segment, an additional metric is available besides sample accumulation, which proves to be a useful and robust method to accurately rank high-frequency locations. An illustration of the theory can be seen in Figure 2 and a discretization view in Figure 3.

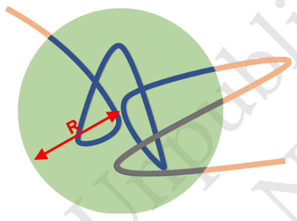

Figure 2: A theoretical continuous line representing the true location of a user, split into segments that are within an arbitrary radius R.

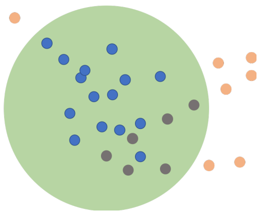

Figure 3: A representation of a discontinuous sampled set of the estimated position of a user, again split into segments within a radius.

```
procedure ITSPreProcess
main:
    sorted locations ← SortByTime(locations)
    current ← 0
    begin ← 0
    prev ← 0
    goto loop.
    labels ← Cluster(relevant locations)
    a ← relevant locations
    b ← number of samples
    c ← total time
    x, y ← CalcCentroids(label, a, b, cn)
    close;
loop:
    x ← sorted locations(current)
    y ← sorted locations(begin)
    z ← sorted locations(prev)
    pxy ← Distance(y, x) < MaxPrecision(x, y)
    md ← Distance(y, x) < max allowable distance gap
    mt ← TimeDiff(x, z) < max allowable time gap
    mtd ← TimeDiff(z, y) > min time difference
    mns ← prev - begin > min number of samples
    if pxy or md or mt then
        if mtd or mns then
            start time ← GetTime(y)
            total time ← TimeDiff(y, z)
            number of samples ← prev - begin
            relevant locations ← sorted locations (prev, begin)
            centroid ← CalcCentroid(relevantlocations)
        begin ← current
    prev ← current
    current ← current + 1
    if current == EOF then return
    else goto loop.
```

5 RESULTS

Overall, the results vary in execution times and accuracy, with a limited number of off-the-shelf algorithms performing well enough to be considered for future use. One major weakness found in most evaluated algorithms was seen in the severe drop in accuracy across the board when clustering international data points as opposed to just domestic in the small dataset. Based on the results, Mini Batch K-Means performed substantially better than any off-the-shelf algorithm, but still suffered from the lack of a time aggregate when identifying high-frequency locations. The largest advantage the ITS pre-process method provided was the estimated time spent at each location, instead of using the number of samples as a weighting factor.

5.1 Experimental Setup

Hardware to run experiments consisted of a single TACC Stampede2 node, using dual 24-core Intel Xeon Platinum 8160 processors and 192 GB of main memory. The operating system environment consisted of Centos 7, running Singularity v2.1 and Jupyter v1.0. The original container file was created for Docker v2.2 and later converted to use Singularity. The software used to run the experiments are Python v3.6.4, Scikit-learn v0.22.1, Pandas v1.0.1, NumPy v1.18.1, and Matplotlib v3.1.3.

5.2 Clustering Methods Runtime

DBSCAN was unable to load the full dataset into memory to begin processing, while Affinity Propagation and Spectral Clustering algorithms failed to produce results due to an execution of over eight hours. There is a large range of variability in the execution time in Table 2. For instance, many of the methods took over an hour to run on a relatively small dataset when compared to what the overall study will be using, while some are under five seconds. The Mini Batch K-Means method also displayed significant variability when presented the two datasets, one of which is a subset of the full dataset and processed for much longer than the full set. The likely explanation of the significant non-linear performance is related to the iterative nature of K-Means and the subsampling of the batch process. The execution times showing significant changes based on input data also encourage the use of a pre-processing method to reduce the number and distance between points. As a note, in the case of the two pre-processor methods, the final ranked locations could not be realized without the use of a clustering algorithms being run on the pre-processed data. Therefore, the full execution time of both pre-processor and clustering method were combined in the case of Hierarchical Segmentation and ITS algorithms.

5.3 Clustering Methods Accuracy

In order to judge accuracy for the particular dataset used in the experiment, a priori knowledge of three ranked high-frequency coordinate locations are used to measure against all centroids produced by the cluster algorithms. For the purposes of this experiment the rank of the locations, based on time spent, is as important as identifying the correct locations. Location differences used a standard GeoPy geodesic conversion from coordinates to meters, and the predicted location must be within a 650 meter radius from the center of the golden samples to be counted as a valid point. The ITS

Table 2: Clustering Methods Execution Time

Clustering Method	Small Dataset (seconds)	Full Dataset (seconds)
`Mini Batch K-Means`	10.13	0.79
`Affinity Propagation`	96.43	DNF
`Mean-Shift`	10.27	5134.53
`Spectral`	61.77	DNF
`Ward Hierarchical`	4.57	720.09
`Agglomerative`	2.42	437.78
`DBSCAN`	3.66	OOM
`OPTICS`	18.03	4181.17
`BIRCH`	0.27	4.77
`Gaussian Mixtures`	0.12	4.60
`Hierarchical Seg`	298.53	7564.12
`ITS`	35.25	692.87

1. Results are the reported Python runtimes.
2. Affinity Propagation and Spectral Clustering had runtimes over several hours when run on the full dataset.
3. DBSCAN was unable to load the full dataset into memory. 4. Bandwidth estimation for Mean-Shift was added into final time.
5. Connectivity matrix generation time for Ward Hierarchical Clustering method was added into final time.
6. 16000 records processed for the small dataset case and 292903 for the full dataset.

pre-process method is split into two rows to highlight the advantages of using an estimate of time as a metric instead of the number of samples. As an aside, the ITS method provided points within a 300 meter radius.

5.4 Rank by Number of Samples or Estimated Time Accumulated

In Figure 3, the only methodology that achieved perfect accuracy on the full dataset was ITS, but only after the results had been sorted based on the estimated time accumulated in that region. For reasons unknown, the number of samples captured during international travel accounted for 4% of the total number of samples in the dataset, while the real time spent at this particular location is estimated to be close to 2%. Compared with the third ranked location used to measure accuracy, the number of samples came in at 4% with a real time of closer to 4%. The number of samples in this situation caused a false prediction of a high-frequency location due to the variation of samples. The ITS pre-processing method provides an accurate method in estimating the time spent at a location as a trivial operation, leading to accumulation of time calculation for clustering methods post-processing and ranking.

5.5 Parallelization

A small subset of clustering algorithms had the ability to easily parallelize the function using a parameter to enable multi-processes, which only included Mean-Shift, Ward Hierarchical Clustering, Agglomerative Clustering, DBSCAN, and OPTICS. As the data in the figure 4 show, none of these algorithms had any noticeable speed-up, and in fact a few displayed a slowdown. Reasonable efforts were

Table 3: Clustering Methods Precision and Accuracy

Clustering Method	Small Dataset	Full Dataset
`Mini Batch K-Means`	2/2	2/2
`Affinity Propagation`	2/1	DNF
`Mean-Shift`	2/2	0/0
`Spectral`	0/0	DNF
`Ward Hierarchical`	2/2	0/0
`Agglomerative`	2/2	0/0
`DBSCAN`	2/2	OOM
`OPTICS`	2/2	1/0
`BIRCH`	2/2	0/0
`Gaussian Mixtures`	2/2	1/1
`Hierarchical Seg`	3/3	0/0
`ITS`	3/3	2/2
`ITS Time Sorted`	3/3	3/3

1. The small dataset only includes domestic location records, while the full data set contains international coordinates.
2. P/Q results may be interpreted as P representing the number of returning centroid coordinates within a 650 meter radius regardless of ordering, while Q is the number of centroid coordinates within a 650 meter radius and predicting the correct time-spent priority order.
3. Affinity Propagation and Spectral Clustering had runtimes over several hours when run on the full dataset.
4. DBSCAN was unable to load the full data set into memory.
5. 16000 records processed for the small dataset case and 292903 for the full dataset.
6. ITS Time Sorted ranks the results by the total time estimates instead of the number of samples.

made to properly scale the out-of-box solutions, but any attempt never resulted in a significant speedup. Thus, the motivation for the proposed pre-processing ITS function as this scales while maintaining accuracy. As mentioned in section 5.2, the execution times and speedup recorded for Hierarchical Segmented and ITS methods in Figure 5 and Table 4 include the final clustering step time as well as pre-processing.

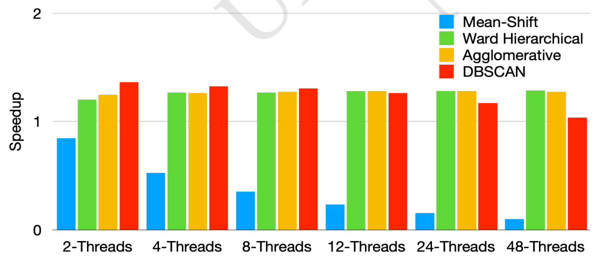

Figure 4: Strong scaling of off-the-shelf algorithms that were trivial to add parallel functionality. Due to the extreme runtimes/memory constraints, the small dataset was used to show speedup.

6 CONCLUSION

Testing several off-the-shelf clustering software solutions motivated the search for an alternative method for consistent accuracy and scalability. By exploiting the temporal locality that is inherently structured within the captured location data, the quality of the classification of high-frequency locations of particular users show increased performance by using the ITS pre-processing method. A more accurate and robust algorithm for calculating ranked high-frequency locations can be had by relying on the estimated time spent within a cluster, which is easily achieved using the ITS pre-processing method, instead of the number of samples normally provided by all other clustering methods. The new functionality does this while keeping a relatively small time complexity and at most $O(N)$ memory consumption. Since the pre-processing method lends itself well to parallelization, the application of using this on several thousand users is possible by scaling the hardware. Future research will focus on adding more digital phenotyping data characterization, with a requirement for scalable solutions.

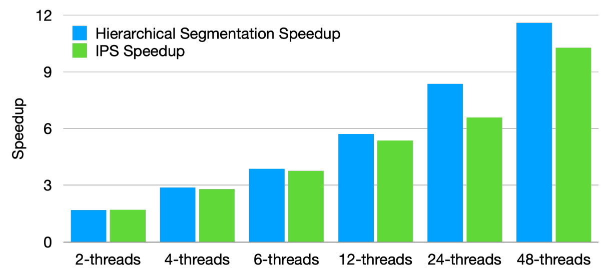

Figure 5: Strong scaling speedup of the total runtime, including sorting and post-clustering steps, for Hierarchical Segmented and ITS using the full dataset.

Table 4: Total Parallel Pre-processing with Final Cluster Runtimes

Processes	Hierarchical Segmented (seconds)	ITS (seconds)
1	7564.12	692.87
2	4482.26	419.15
4	2621.23	257.05
6	1950.88	191.32
12	1320.54	133.26
24	903.37	108.85
48	652.79	69.71

1. Results are the reported Python runtimes.

ACKNOWLEDGMENTS

The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources that have contributed to the research results reported within this paper. This work is also supported in part by the National Science Foundation under Grant No. 1838901.

REFERENCES

[1] [n.d.]. Beiwe Research Platform. https://www.beiwe.org/
[2] [n.d.]. The Cost of Diabetes. https://www.diabetes.org/resources/statistics/cost-diabetes

[3] [n.d.]. HCUPnet. https://hcupnet.ahrq.gov
[4] [n.d.]. Healthy Pregnancy. https://healthy-pregnancy.org/
[5] 2018. Births: Final data for 2016. *National Vital Statistics Reports; vol 67 no 1.* (2018).
[6] Laura Alessandretti, Piotr Sapiezynski, Vedran Sekara, Sune Lehmann, and Andrea Baronchelli. 2018. Evidence for a conserved quantity in human mobility. *Nature Human Behaviour* 2, 7 (2018), 485–491.
[7] Lauren P Alexander and Marta C González. 2015. Assessing the impact of real-time ridesharing on urban traffic using mobile phone data. *Proc. UrbComp* (2015), 1–9.
[8] Domenica Arlia and Massimo Coppola. 2001. Experiments in parallel clustering with DBSCAN. In *European Conference on Parallel Processing*. Springer, 326–331.
[9] Hugo Barbosa, Marc Barthelemy, Gourab Ghoshal, Charlotte R James, Maxime Lenormand, Thomas Louail, Ronaldo Menezes, José J Ramasco, Filippo Simini, and Marcello Tomasini. 2018. Human mobility: Models and applications. *Physics Reports* 734 (2018), 1–74.
[10] Marco Capó, Aritz Pérez, and Jose A Lozano. 2017. An efficient approximation to the K-means clustering for massive data. *Knowledge-Based Systems* 117 (2017), 56–69.
[11] Serdar Çolak, Lauren P Alexander, Bernardo G Alvim, Shomik R Mehndiratta, and Marta C González. 2015. Analyzing cell phone location data for urban travel: current methods, limitations, and opportunities. *Transportation Research Record* 2526, 1 (2015), 126–135.
[12] Merkebe Getachew Demissie, Francisco Antunes, Carlos Bento, Santi Phithakkitnukoon, and Titipat Sukhvibul. 2016. Inferring origin-destination flows using mobile phone data: A case study of Senegal. In *2016 13th International conference on electrical engineering/electronics, computer, telecommunications and information technology (ECTI-CON)*. IEEE, 1–6.
[13] Riccardo Gallotti, Rémi Louf, Jean-Marc Luck, and Marc Barthelemy. 2018. Tracking random walks. *Journal of The Royal Society Interface* 15, 139 (2018), 20170776.
[14] Yangqing Jia, Jingdong Wang, Changshui Zhang, and Xian-Sheng Hua. 2008. Finding image exemplars using fast sparse affinity propagation. In *Proceedings of the 16th ACM international conference on Multimedia*. 639–642.
[15] Shan Jiang, Joseph Ferreira, and Marta C Gonzalez. 2017. Activity-based human mobility patterns inferred from mobile phone data: A case study of Singapore. *IEEE Transactions on Big Data* 3, 2 (2017), 208–219.
[16] Mu Li, Xiao-Chen Lian, James T Kwok, and Bao-Liang Lu. 2011. Time and space efficient spectral clustering via column sampling. In *CVPR 2011*. IEEE, 2297–2304.
[17] Xia Li, Zhisheng Zhong, Jianlong Wu, Yibo Yang, Zhouchen Lin, and Hong Liu. 2019. Expectation-maximization attention networks for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*. 9167–9176.
[18] Hai-Ying Liu, Erik Skjetne, and Mike Kobernus. 2013. Mobile phone tracking: in support of modelling traffic-related air pollution contribution to individual exposure and its implications for public health impact assessment. *Environmental Health* 12, 1 (2013), 93.
[19] Ryan Maas, Jeremy Hyrkas, Olivia Grace Telford, Magdalena Balazinska, Andrew Connolly, and Bill Howe. 2015. Gaussian mixture models use-case: in-memory analysis with myria. In *Proceedings of the 3rd VLDB Workshop on In-Memory Data Mangement and Analytics*. 1–8.
[20] John E. McDonough. 2015. The United States Health System in Transition. *Health Systems & Reform* 1, 1 (2015), 39–51.
[21] Dave Milne and David Watling. 2019. Big data and understanding change in the context of planning transport systems. *Journal of Transport Geography* 76 (2019), 235–244.
[22] Amanda Montañez. 2019. The Cost of Giving Birth in the U.S. https://blogs.scientificamerican.com/sa-visual/the-cost-of-giving-birth-in-the-u-s/
[23] Mostofa Ali Patwary, Diana Palsetia, Ankit Agrawal, Wei-keng Liao, Fredrik Manne, and Alok Choudhary. 2013. Scalable parallel OPTICS data clustering using graph algorithmic techniques. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. 1–12.
[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.
[25] Mario B Rojas IV, Eazaz Sadeghvaziri, and Xia Jin. 2016. Comprehensive review of travel behavior and mobility pattern studies that used mobile phone data. *Transportation Research Record* 2563, 1 (2016), 71–79.
[26] Meead Saberi, Hani S Mahmassani, Dirk Brockmann, and Amir Hosseini. 2017. A complex network perspective for characterizing urban travel demand patterns: graph theoretical analysis of large-scale origin–destination demand networks. *Transportation* 44, 6 (2017), 1383–1402.
[27] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21.
[28] Jameson Lawrence Toole. 2015. *Putting big data in its place: understanding cities and human mobility with new data sources.* Ph.D. Dissertation. Massachusetts Institute of Technology.
[29] J. Torous, M. V. Kiang, J. Lorme, and J.-P. Onnela. 2016. New Tools for New Research in Psychiatry: A Scalable and Customizable Platform to Empower Data Driven Smartphone Research. *JMIR Mental Health* (2016).
[30] Zhenzhen Wang, Sylvia Y He, and Yee Leung. 2018. Applying mobile phone data to travel behaviour research: A literature review. *Travel Behaviour and Society* 11 (2018), 141–155.
[31] Luc Johannes Josephus Wismans, K Friso, J Rijsdijk, SW de Graaf, and J Keij. 2018. Improving a priori demand estimates transport models using mobile phone data: A rotterdam-region case. *Journal of Urban Technology* 25, 2 (2018), 63–83.
[32] Donghui Yan, Ling Huang, and Michael I Jordan. 2009. Fast approximate spectral clustering. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. 907–916.
[33] Changjiang Yang, Ramani Duraiswami, Daniel DeMenthon, and Larry Davis. 2003. Mean-shift analysis using quasinewton methods. In *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, Vol. 2. IEEE, II–447.
[34] Changjiang Yang, Ramani Duraiswami, Daniel DeMenthon, and Larry Davis. 2003. Mean-shift analysis using quasinewton methods. In *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, Vol. 2. IEEE, II–447.
[35] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1996. BIRCH: an efficient data clustering method for very large databases. *ACM Sigmod Record* 25, 2 (1996), 103–114.
[36] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. 1997. BIRCH: A new data clustering algorithm and its applications. *Data Mining and Knowledge Discovery* 1, 2 (1997), 141–182.