

Scalable Solution for the Anonymization of Big Data Spatio-Temporal Trajectories

Hajlaoui Jalel Eddine(✉)

Lambda Lab, Paris, France
hajlaouijalel_ig@yahoo.fr

Abstract. Regardless of the collection location, mobile traffic data contains information about many aspects of subscribers' lives, including their activities, interests, schedules, travel and preferences. It is precisely the ability to access such information on unprecedented scales that is of critical importance for studies in a wide variety of fields. However, access to such a rich source also raises concerns about potential infringements on the rights of mobile customers regarding their personal data: among other things, individuals can be identified, their movements can be modified, their movements can be tracked and their mobile stage fright can be monitored. As a result, regulators have been working on legislation to protect the privacy of mobile users. In this optic, we provide a scalable solution to anonymize Big Data Spatio-temporal Trajectories of mobile users.

Keywords: Big Data · Anonymization · Privacy

1 Introduction

Mobile traffic transmits information about people's movements, interactions and mobile service consumption on unprecedented scales. Traditional techniques data collection, e.g., census, population surveys or phone interviews, cannot provide a perspective, even from a comparable distance, on human activities. As a result, as soon as the rise of the mobile communications became obvious - about a decade ago - sociologists, epidemiologists, transport and telecommunication experts have seen in the collected data set by mobile network operators a clear opportunity to perform their analyses while maintaining a high level of detail for each individual [1–4]. The first landmark studies using large mobile traffic datasets were conducted in the United States towards 2006. Since then, the growth of mobile traffic analysis has been spectacular and has grown at an annual rate of 90%. The main reason for this phenomenon is the nature of mobile traffic databases, allowing large-scale research relevant to different disciplines [5]. However, other factors contributed to the success of telephony mobile traffic analyses. The increase in the volume of workload is primarily based on the growing availability of datasets. As mentioned above, mobile operators have always monitored mobile traffic in their networks, for maintenance, efficiency and billing purposes. Yet they have traditionally been very careful about sharing the collected data. This attitude has

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022, LNCS 13375, pp. 465–476, 2022.
https://doi.org/10.1007/978-3-031-10522-7_32

changed in recent years, as more and more operators are inclined to open their data to analytical processing in the broad sense. On the one hand, operators driven by the value that mobile traffic data can have, are still deploying more advanced probes in their networks allowing more detailed measurement of subscriber activities. On the other hand, mobile services have been developed from simple calling and texting to permanent applications based on Cloud applications [6,11]: this leads to much more frequent interactions between users (or their devices) and the network, which leads to a significantly higher granularity of activity samples recorded at the operator's side. A third key element, resulting from the first two preceding ones, is the emergence of a very active company and an interdisciplinary community which brings together researchers and industrial players. The cohesion of university-industry partnerships have important consequences: for example, mobile operators encourage fundamental and applied research on mobile traffic in response to targeted challenges. As examples, we can cite challenges of Telecom Italia[1].

In this paper we implement our scalable solution of spatio-temporal trajectories merging based on our Cartesian Product k-merge (FCP-k-merge) algorithm which extends the k-merge algorithm. In Sect. 2, we discuss the taxonomy of mobile traffic analysis approaches. In Sect. 3, we illustrate the architectural aspect of our application while focusing on the optimization of the algorithm for calculating the aggregation cost of the movements of mobile users by generalizing their spatio-temporal trajectories. In Sect. 4, we conclude the paper and highlight perspectives of our work.

2 Taxonomy of Mobile Traffic Analysis Solutions

The adopted classification [12] of traffic analysis works revolves around the study topics, each of which includes multidisciplinary contributions. The roadmap of the proposed hierarchy is illustrated in Fig. 1. In the upper layer, authors explicit three macro-research topics across multiple domains: they cite the social analysis, mobility analysis and network analysis, respectively. Then, a topic tree is detailed within each macro-topic. Below, we show a taxonomy of the topics included in the classification:

- *Social analysis* explore the relationships between mobile traffic and a broad set of social features. The research focuses on characterizing the social structure of mobile user interactions, and on studying the impact of demographic, economic or environmental factors on the way of mobile services consumption. This topic includes works which exploit social characteristics, inferred from mobile traffic, for the characterization and mitigation of epidemics.
- *Mobility analysis* deal with the extraction of mobility information from mobile traffic. Mobility is considered here in its largest sense, and focus on generic human movements, both individual and collective, as well as specialized patterns followed by specific users, for example trips on transport systems.

[1] http://www.telecomitalia.com/bigdatachallenge/.

This branch also defines the reliability of mobile traffic data as a source of mobility information.

– *Network analysis* have a more technical perspective, as they focus on understanding the dynamics of mobile traffic demand and how to evolve the mobile network infrastructure to better accommodate it. The work in this category therefore relates either to the characterization of the uses of mobile services, or to the operation of this knowledge to implement different sophisticated technological solutions. The wide majority of the categories described above are interdisciplinary in nature. In the lower part of Fig. 1, authors provide a representation of the importance of five main areas for the different topics of mobile traffic analysis. The relationships are not probably clear cut, but it can be seen that mobility studies are the ones that capture the tremendous attempts to contribute. Most of the categories have sense for two or three disciplines. The only non-multidisciplinary subjects is devoted to develop solutions for mobile networks: being fairly specific and very technical subjects, it is evident that they entice contributions only from the networking community.

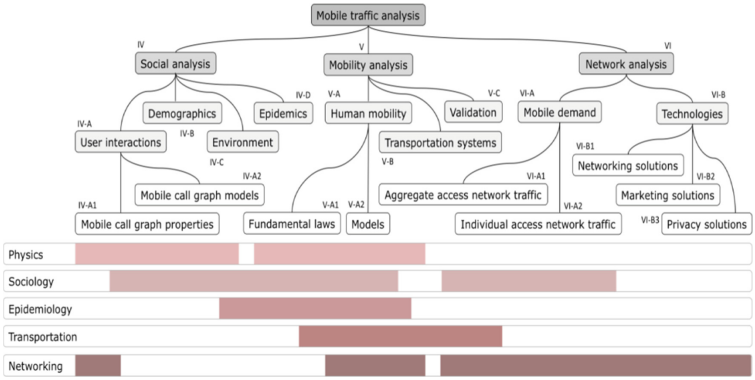

Fig. 1. Functional architecture of the mobile data clustering application.

3 Spatio-Temporal Big Data Clustering Solution

In this section, we present our contribution to the anonymization of user mobility data within λ_*Company*[2], this massively collected data makes it possible to trace the movement of people. The anonymization of these Big Data consists of aggregating them using clustering techniques so as not to be able to re-identify people from call logs or probe data and anonymize their movements. Thus, we apply our technique of aggregation/fusion of spatiotemporal trajectories in a Big Data ecosystem with the Spark platform and the Scala language.

[2] For the sake of anonimity of the company owning the data set.

3.1 Reference Architecture

Data storage focuses on how to efficiently store a huge volume of data, allowing faster read/write speeds during processing. The storage layer must take care to store all types of data and continue to evolve with the exponential growth of data. In our case, the source data is csv files stored in HDFS and transformed into Spark for batch processing for offline analysis. We have added a streaming module for real-time processing in order to extend the scope of the reference architecture. A real-time processing system connects directly to the data ingestion layer and is applicable for online analysis; it should provide low latency processing results. The cluster manager is trained by resource managers to accommodate resource provisioning and scheduling throughout the distributed system. Communication between Spark applications and transactional nodes, masters and workers, and intercommunication between drivers and Spark executors can be secure (i.e., encrypted). Encryption must be configured individually on each node of the cluster (Fig. 2).

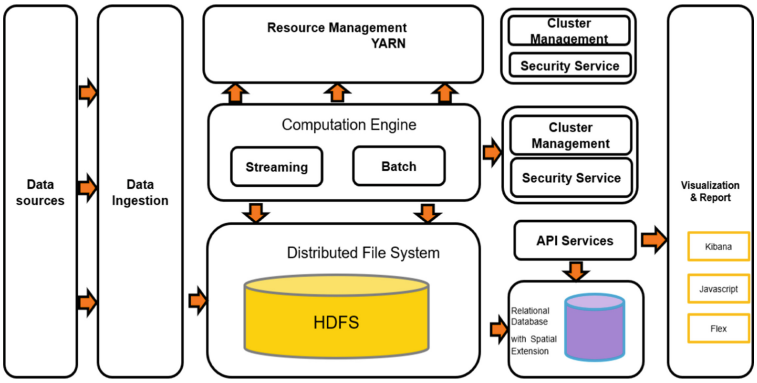

Fig. 2. Reference architecture of the mobile data clustering application.

The data visualization layer is the presentation layer. Dashboards can be created to help the user perform visual analysis directly from the ingested source or processed data. In Big Data environments - Hadoop [13] and Spark [14] echosystem - there are no integrated components for data visualization. Flex, JavaScript or Kibana can be used as a visualization tool to present the dashboards in real time.

Apache Hadoop, introduced by Yahoo in 2005, is the open source implementation of the MapReduce programming paradigm. The main characteristic of Hadoop is to use hardware base mainly distributed for the parallel processing of batch jobs. The heart of Hadoop is its fault-tolerant Hadoop Distributed File Systems (HDFS) file system which can be explicitly set to span many machines. In HDFS, the data block is much larger than a traditional file system (4 KB vs. 128 MB). Therefore, it reduces the memory required to store the metadata

on the locations of the data blocks. In addition, it reduces the search operation in large files. In addition, it dramatically improves network usage because only fewer network connections are required for random read operations. In the architecture of HDFS, there are mainly two types of nodes: Name Node and Data Node. The Name Node contains the metadata of HDFS blocks and the Data Node is the location where the actual data is stored. By default, three copies of the same block are stored on data nodes to make the system fault tolerant. Hadoop's resource manager is called Yarn. It is composed a central resource manager that resides in the master node and many node managers that live on the slave nodes. When an application is submitted to the cluster, the application master negotiates resources with the resource manager and starts the container (where the actual processing takes place) on the slave nodes. The main disadvantage of Hadoop was that it stored intermediate results on disk. Therefore, for intensive shuffle operations (the rearrangement of data between machines, by keys, so that entire operations can be handled) like iterative machine learning, a huge amount of data is stored on disk and transferred. On the network, which represents a significant overload on the entire system.

Apache Spark is one of the world's premier big data processing platforms. It is a large scale, open source data processing framework. It primarily focuses on high-speed server cluster computing and provides scalable and interactive analysis through high-level APIs. Spark supports batch or streaming data analysis, machine learning, and graph processing. It can also access various data sources such as HDFS, HBase, Cassandra, etc. and use Resilient Distributed Dataset (RDD), Data Frame (DF), or Data Set (DS) structures for data abstraction. Compared to Hadoop system tasks, Apache Spark can perform most in-memory calculations and offers better performance for some applications such as iterative algorithms. When the results do not fit in memory, the intermediate results are written to disk. Spark can run locally on a single machine, in a local cluster, and in the cloud. It runs on Hadoop Yarn, Apache Mesos, and the standalone cluster manager by default. Applications or jobs are divided into several sets of tasks called stages that are interrelated. All of these steps form a directed acyclic graph (DAG), where each step is executed one after the other.

3.2 Functional Dimension

The data pre-processing step includes flexible procedures that are applied before storing and analyzing the trajectory data, such as data format transformation and associating the stamped movement sequences to each user. After being properly treated, all trajectories are kept in memory to support efficient analysis and rapid restitution of results. A (raw) sample of a space-time trajectory represents the position of a subscriber at a given time, and we model it with a real vector of length $3_s = (t(s), x(s), y(s))$. Since a data set is characterized by a finite granularity in time and space, the sample is actually a niche covering a minimum of time and spatial intervals. A spatiotemporal (sub)trajectory describes the movements of a single subscriber during the duration of the occurrence of the movements in the data set. Formally, each trajectory is an ordered vector

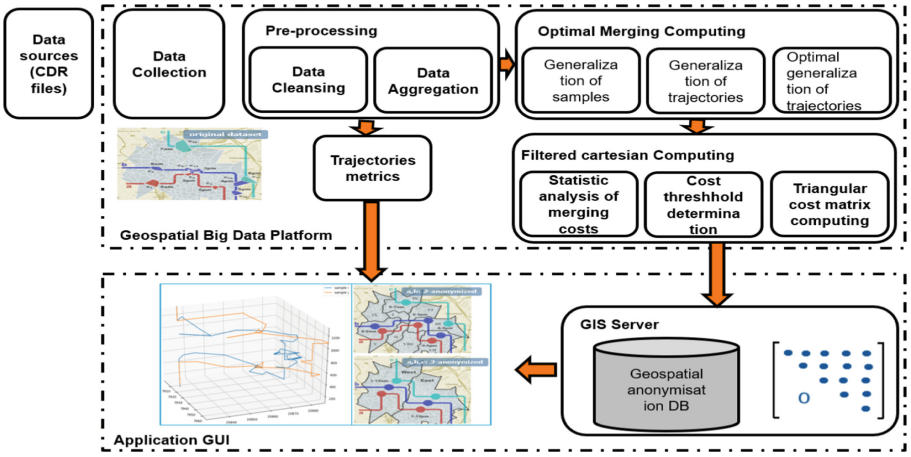

Fig. 3. Functional architecture of the mobile data clustering application.

of samples $S = (s_1, ..., s_N)$, where the order is induced by the time coordinate, i.e., $t(si) < t(si')$ si et seulement si $i \leq i'$.

A generalized trajectory, obtained by merging different trajectories, is defined as an ordered vector of generalized samples $G = (G_1, ..., G_Z)$. Here, the order is more subtle, and based on the fact that the time intervals extended by the generalized samples do not overlap, a property that will be called temporal consistency. Once able to optimally merge the (sub) trajectories, the calculation of the cost of merging trajectories is a combinatorial treatment that involves all combinations of the trajectories one by one. The computation result is a matrix $n \times n$ where n is the number of trajectories. We perform a statistical computation in order to determine a threshold (i.e., a mean cost value). The resulting matrix is a triangular matrix where the cost values are not duplicated. The results of the merged trajectories are stored in a geospatial anonymization database (Fig. 3).

3.3 Algorithm for Merging Trajectories

To calculate the cost between trajectories to merge, we implemented $k - merge$ [15–17] an algorithm to efficiently search the set of raw samples S, extract the subset of elementary partitions, $K^{\times}$, and identify the optimal partition $G^{\times}$. The k-merge algorithm, detailed in Algorithm 1, starts by filling a set of raw samples S, whose items $s_{i,j}$ are sorted according to their time value $t(s_{i,j})$ (lines 1 to 3). Then, it processes all the samples according to their time order (line 6). More precisely, the algorithm tests, for each sample s_θ in position θ, all the sets $\{s_{\theta'}, ..., s_\theta\}$, with $\theta' \leq \theta$, as follows. The first loop ignores incomplete sets that do not contain at least one sample of each input path (line 8). The second loop runs until the first non-elemental set is encountered (line 10). Here, the algorithm generates the current set (complete and elementary) $\{s_{\theta'}, ..., s_\theta\}$ to G, and checks if G reduces the total cost of merging down to s_θ. If so, the cost is updated by

adding $c(G)$ to the cumulative cost up to $s_{\theta'-1}$, and the resulting S (partial) score that includes G is stored (lines 11 through 14). Once out of the loops, the cost associated with the last sample is the optimal cost, and just navigate to arrive in the structure of the partition to retrieve the associated $G^{\times}$ (lines 16–17). Note that, in order to update the inclusion cost of the current sample s_θ (line 13), the algorithm checks only the previous samples in time. Therefore, the optimal decision up to s_θ should not depend on any of the samples in the initial trajectories that come later than s.

Algorithm 1: `k-merge` algorithm pseudocode.

```
   input : Trajectories S_1, ..., S_k, where S_i = (s_{i,1}, ..., s_{i,N_i})
   output: Generalized sample set G*, Cost C(G*)
1  foreach i ∈ [1, k] do
2  |  S_i = ⋃_{j=1}^{N_i} {s_{i,j}};
3  S ← timesort (S_1 ∪ ··· ∪ S_k);
4  Cost ← (0, ∞, ..., ∞);
5  Partition ← (NULL, ..., NULL);
6  foreach s_θ ∈ S do
7  |  θ' = θ − 1;
8  |  while incomplete (s_{θ'}, ..., s_θ) do
9  |  |  θ' = θ' − 1;
10 |  while elementary (s_{θ'}, ..., s_θ) do
11 |  |  G ← generalize (s_{θ'}, ..., s_θ);
12 |  |  if Cost[θ] > c(G) + Cost[θ' − 1] then
13 |  |  |  Cost[θ] ← c(G) + Cost[θ' − 1];
14 |  |  |  Partition ← (θ' − 1, G);
15 |  |  θ' = θ' − 1;
16 G* ← visit (Partition);
17 C(G*) ← Cost[|S|];
```

3.4 Filtered Cartesian Product k-merge (FCP-k-merge) Algorithm

The Filtered Cartesian Product k-merge (FCP-k-merge) algorithm allows optimizing the execution of the k-merge algorithm which, once executed with a consequent set of users, runs in exponential time. For every 1000 users, k-merge completes its execution in about 6 h in λ_*Company*'s private cluster. The idea is to limit the cost calculation only for pairs of trajectories whose merging cost does not exceed a certain threshold λ. As a result, we avoid an exhaustive cartesian product and thus reduce the processing space in the triangular cost matrix. The FCP-k-merge algorithm allows us to determine an average threshold by performing the steps shown in Algorithm 2.

First, this algorithm accumulates the number of costs using a spark *accumulator* that retrieves the costs of each worker node of the cluster. Then, the final number of costs is assigned to each bin or segment of the vector. Finally, the threshold value λ is calculated and in the ideal case it corresponds to the histogram's median. In the case where the last bin contains an important value, i.e. the distribution of bins values is unbalanced, we repeat the execution with a higher bin width and a higher number of bins (lines 8–10) (Fig. 4).

Algorithm 2: Filtered cartesian computing

```
Cost matrix: Cn * n,
n_bins: bins number
w_bin: bin width
Threshold λ
/* initialisation                                                    */
λ = 0 // threshold
Max_bin_value = n_bins * w_bin // Maximum value
bin_accunt[ ] = Array(Int) of size nbin + 1
//step 1:
for (i := 0 to n ) do
    For (j = 0 to n)
    if (Sij ≥ max_bin_value) then
        do: bin_accum[nbin+1]+=1
    else
        do: index=(int) ((Sij)/wbin)
        bin_acc[index]+=1
    if bin_acc[n + 1] ≫ medium_value then
        repeat step 1 with w_binNew ≻ w_bin and n_binsNew ≻ n_bins
    else
        return λ = Median(bin_acc)
```

3.5 Used Mobility Data

Mobile traffic is most often in the form of CRA (Call Detail Record), or CDR (Call Detail Record). It can include several fields but for mobility studies only 3 are essential:

- User ID (cell phone number or SIM card number).
- Time stamp of connection to the mobile network.
- Mobile network identifier (CELL_ID).

This third element is composed of 2 sub-identifiers (Figs. 5 and 6):

- LAC = Location Area Code
- CI (Cell Identification) or SAC (Service Area Code) or ECI (E-UTRAN Cell Identifier) depending on the technology (2G, 3G or 4G)

On 2G and 3G networks both elements (LAC and CI/SAC) are needed to identify an antenna (CI and SAC codes are guaranteed to be unique within a LAC). On the 4G network the ECI code is unique for each antenna and is therefore sufficient to identify it. An example of CRA data is shown as depicted in the Fig. 7:

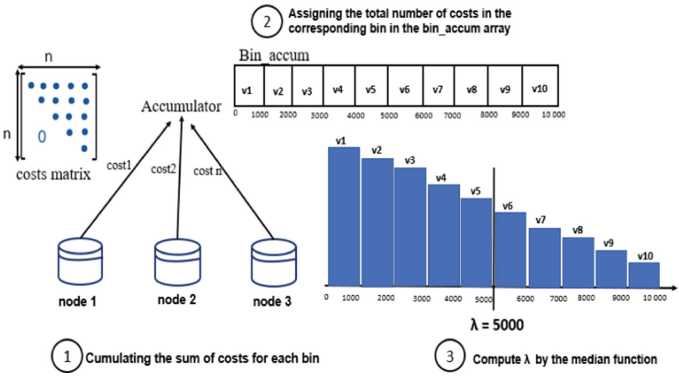

Fig. 4. Steps to calculate the λ cost threshold.

```
User_id;datetime;LAC;CELL
935b0906b5cdb56a4407c4098ae754ac846911e7;2015-09-15 12:32:35;1054;19851
15637b4580e2a5e8f6a0cc2c62d50f74b67d6b3c;2015-09-15 12:32:05;1054;6775
e384f05b9f7b6a240e6161b203b80d02cfe36b15;2015-09-15 12:32:36;1054;49616
15637b4580e2a5e8f6a0cc2c62d50f74b67d6b3c;2015-09-15 12:32:25;1054;6771
c8444c5e849ef6a03e948f3e1a863c139790b4fd;2015-09-15 12:32:37;1054;56057
4355c1a7e14a1eb898dc71937ed92e2653e13747;2015-09-15 12:32:21;1054;37453
4355c1a7e14a1eb898dc71937ed92e2653e13747;2015-09-15 12:32:41;1054;37453
```

Fig. 5. CRA mobile data.

The datetime field can be given in clear as in the example but also in unix epoch (number of seconds since 1/1/1970). The data are in no particular order. To build a trajectory, it is therefore necessary to extract the data from a user and sort them according to the datetime field. An example of reference data for the datetime is shown in the Fig. 8: The X/Y positions are given in Lambert 2 extended projection. It is an Euclidean projection so to calculate the distances between the points, we can use the Pythagorean theorem.

```
CELL ;LAC ;SITE ;X ;Y
19851;1054;00000493H3;0;3G;795782;2081272
6775;1054;00000119H3;150;3G;792940;2082960
```

Fig. 6. Network antenna data.

3.6 Experimentations

The experiments are performed on a local Dell computer with 32 GB RAM and an Intel(R) Core(TM) i7-8700, 3.2 GHZ, 6-core processor. Let the following

test set containing 10 users, we want to calculate the cost matrix corresponding to them:

- U_1: sorted([(58, 7906, 20835), (141, 7906, 20835), (321, 7906, 20835)])
- U_2: sorted([(50, 7906, 20835), (141, 7906, 20835), (315, 7906, 20835), (870, 6972, 20868), (874, 7926, 2044)])
- U_3: sorted([(55, 7906, 20835), (147, 7906, 20835), (318, 7906, 20835)])
- U_4: sorted([(210, 4916, 12532), (366, 7916, 12330), (371, 5902, 20835), (380, 7906, 20868)])
- U_5: sorted([(331, 4916, 12532), (512, 7906, 10731), (515, 7906, 20835), (524, 7906, 20868)])
- U_6: sorted([(221, 1549, 22125), (226, 7104, 73122), (229, 8506, 20835)])
- U_7: sorted([(82, 5221, 10212), (86, 7906, 10731), (89, 7916, 10728), (524, 7916, 10788)])
- U_8: sorted([(918, 7906, 20835), (944, 1045, 30442), (946, 1054, 71754), (952, 1054, 178)])
- U_9: sorted([(775, 1080, 12532), (760, 1080, 7980), (764, 1080, 2563), (766, 1080, 219)])
- U_10: sorted([(1220, 32519, 12362), (1221, 76, 10731), (1224, 1706, 235), (1228, 1902, 868)])

$$C = \begin{array}{c} \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \\ U_7 \\ U_8 \\ U_9 \\ U_10 \end{array} \begin{array}{c} \begin{array}{cccccccccc} U_1 & U_2 & U_3 & U_4 & U_5 & U_6 & U_7 & U_8 & U_9 & U_10 \end{array} \\ \begin{bmatrix} 0 & 19089 & 15 & 2361162 & 6130776 & 6408512 & 2951165 & 70202010 & 19704074 & 62114524 \\ 0 & 0 & 11017084 & 12208579 & 10364685 & 23532079 & 9708766 & 70847574 & 17045958 & 62577783 \\ 0 & 0 & 0 & 2426703 & 6170160 & 6270386 & 3081974 & 70437324 & 19786403 & 62273656 \\ 0 & 0 & 0 & 0 & 1932290 & 2818218 & 2609514 & 58286864 & 15557076 & 54085463 \\ 0 & 0 & 0 & 0 & 0 & 21082096 & 1462174 & 48788436 & 12226820 & 47663146 \\ 0 & 0 & 0 & 0 & 0 & 0 & 13194158 & 58857192 & 44583150 & 1.06173648e+08 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 68328208 & 13290100 & 051122937 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 15138534 & 32350220 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 20991033 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Fig. 7. Cost matrix for 10 mobile users.

From the cost matrix C depicted in Fig. 7, we notice that users U_1 and U_3 have trajectories whose merging cost is equal to 15 because they look very similar. Users U_6 and U_{10} have trajectories whose merging cost is very huge and is equal to 1.06173648e+08 because the trajectories are quite far in time and geographical coordinates. Figure 8 shows the execution time of the k-merge algorithm for 100 users combined with a number of users ranging from 100 to 1500 users. The maximum execution time is 9888 ms or less than one second. With the FCP-k-merge, this time is likely to be half optimized.

Currently, we are conducting deep experiments on the private $\lambda_Company$ cluster to see the behavior of the FCP-k-merge algorithm in terms of the execution time considering that with the k-merge algorithm, we can reach about 6 h for a combination of 1000 users.

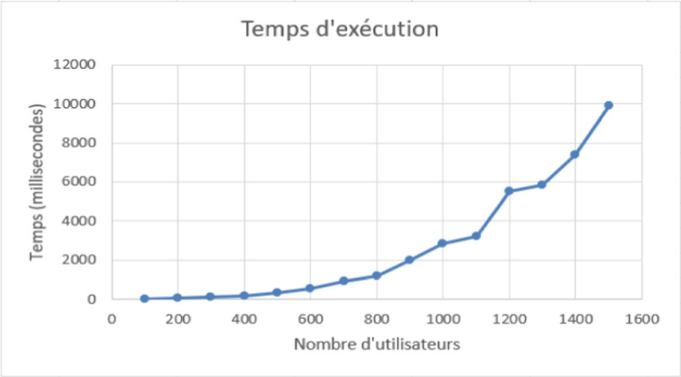

Fig. 8. Execution time of the FCP-k-merge algorithm.

4 Conclusion

This work, focused mainly on privacy, ultimately allowed the development of a proof of concept application, in order to help organizations concerned with the privacy of mobile users by anonymizing their trajectories. The main objective has therefore been the implementation of the k-merge algorithm under Spark with the Scala language. There are still a number many points to improve, especially at the level of partitioning, but the solution is functional and fulfills the objectives that we have set, to perform the clustering of spatiotemporal trajectories. The secondary objective is the optimization of k-merge by the Filtered Cartesian Product k-merge (FCP-k-merge) algorithm which help to optimize execution time by offloading memory with additional processing. In the future we plan to deploy our solution in an elastic Kubernetes infrastructure for the sake of the orchestration of our containerized infrastructure and its high scalability.

References

1. Zhang, C., Fiore, M., Ziemlicki, C., Patras, P.: Microscope: mobile service traffic decomposition for network slicing as a service. In: Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, pp. 1–14, September 2020
2. Pullano, G., Valdano, E., Scarpa, N., Rubrichi, S., Colizza, V.: Evaluating the effect of demographic factors, socioeconomic factors, and risk aversion on mobility during the COVID-19 epidemic in France under lockdown: a population-based study. Lancet Digit. Health **2**(12), e638–e649 (2020)
3. Zhao, Y., Luo, Y., Yu, Q., Hu, Z.: A privacy-preserving trajectory publication method based on secure start-points and end-points. Mob. Inf. Syst. (2020)
4. Bennati, S., Kovacevic, A.: Privacy metrics for trajectory data based on k-anonymity, l-diversity and t-closeness. arXiv preprint arXiv:2011.09218 (2020)
5. Tan, R., Tao, Y., Si, W., Zhang, Y.-Y.: Privacy preserving semantic trajectory data publishing for mobile location-based services. Wirel. Netw. **26**(8), 5551–5560 (2019). https://doi.org/10.1007/s11276-019-02058-8

6. Hajlaoui, J.E., Omri, M.N., Benslimane, D.: Multi-tenancy aware configurable service discovery approach in cloud computing. In: 2017 IEEE 26th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 232–237. IEEE, June 2017
7. Hajlaoui, J.E., Omri, M.N., Benslimane, D., Barhamgi, M.: QoS based framework for configurable IaaS cloud services discovery. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 460–467. IEEE, June 2017
8. Azouzi, S., Hajlaoui, J.E., Ghannouchi, S.A., Brahmi, Z.: E-Learning BPaaS discovery in cloud based on a structural matching. In: SoMeT, pp. 176–189, September 2019
9. Khemili, W., Hajlaoui, J.E., Omri, M.N.: Energy aware fuzzy approach for placement and consolidation in cloud data centers. J. Parallel Distrib. Comput. (2021)
10. Mokni, M., Yassa, S., Hajlaoui, J.E., Chelouah, R., Omri, M.N.: Cooperative agents-based approach for workflow scheduling on fog-cloud computing. J. Ambient Intell. Human. Comput. 1–20 (2021)
11. Mokni, M., Hajlaoui, J.E., Brahmi, Z.: MAS-based approach for scheduling intensive workflows in cloud computing. In: 2018 IEEE 27th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), pp. 15–20. IEEE, June 2018
12. Naboulsi, D., Fiore, M., Ribot, S., Stanica, R.: Large-scale mobile traffic analysis: a survey. IEEE Commun. Surv. Tutor. **18**(1), 124–161 (2015)
13. Zeebaree, S.R., Shukur, H.M., Haji, L.M., Zebari, R.R., Jacksi, K., Abas, S.M.: Characteristics and analysis of hadoop distributed systems. Technol. Rep. Kansai Univ. **62**(4), 1555–1564 (2020)
14. Zaharia, M., et al.: Apache spark: a unified engine for big data processing. Commun. ACM **59**(11), 56–65 (2016)
15. Gramaglia, M., Fiore, M., Tarable, A., Banchs, A.: $k^{\tau,\epsilon}$-anonymity: Towards privacy-preserving publishing of spatiotemporal trajectory data. arXiv preprint arXiv:1701.02243 (2017)
16. Gramaglia, M., Fiore, M., Tarable, A., Banchs, A.: Preserving mobile subscriber privacy in open datasets of spatiotemporal trajectories. In: IEEE INFOCOM 2017-IEEE Conference on Computer Communications,pp. 1–9. IEEE, May 2017
17. Fiore, M., et al.: Privacy in trajectory micro-data publishing: a survey. arXiv preprint arXiv:1903.12211 (2019)