# Scalable Mining of Common Routes in Mobile Communication Network Traffic Data

Olof Görnerup

Swedish Institute of Computer Science (SICS), SE-164 29 Kista, Sweden
olofg@sics.se

**Abstract.** A probabilistic method for inferring common routes from mobile communication network traffic data is presented. Besides providing mobility information, valuable in a multitude of application areas, the method has the dual purpose of enabling efficient coarse-graining as well as anonymisation by mapping individual sequences onto common routes. The approach is to represent spatial trajectories by Cell ID sequences that are grouped into routes using locality-sensitive hashing and graph clustering. The method is demonstrated to be scalable, and to accurately group sequences using an evaluation set of GPS tagged data.

**Keywords:** Cellular networks, mobility, data mining, location privacy

## 1 Introduction

Mobile communication networks have in recent years been recognised as large-scale ubiquitous sensor networks capable of producing massive amounts of valuable mobility data. This data has for instance been used to statistically characterise human mobility [13, 24, 25], in crisis management [5], for traffic modeling [20], for detecting anomalous events [2, 9], to identify common locations [14, 17], for mobility prediction [6, 26], and to determine low-level properties, such as if a terminal is in motion [15] or mode of transportation [23]. There is also an abundance of work where other types of sensors are used to characterise human mobility, predominantly GPS, such as in [1, 3, 19, 21]. Although GPS data has a higher precision than data acquired from mobile communication networks, the former also has several disadvantages, such as not functioning indoors or underground (e.g. in the subway), or having high energy consumption. The penetration rate of GPS equipped devices is currently also significantly lower than for mobile phones in general.

In this paper a scalable probabilistic method for inferring common routes from sequences of Cell IDs from 2G and 3G networks is presented. Similar sequences are related using locality-sensitive hashing, and grouped using graph clustering. Resulting common routes are constituted by the clusters of Cell ID sequences output by the method. The approach is neither limited to short scale (e.g. Markovian) prediction as it captures long correlations in data, nor is it dependent on GPS trajectories as it operates on network data alone.

In a recent related study, Becker et al. have demonstrated that Cell ID handover sequences are stable over routes [4]. Using Cell IDs in concurrence with additional data (e.g. temporal information and cell tower locations), they utilise this to accurately classify sequences with respect to routes given at the outset, although not explicitly addressing scalability. The method presented in this paper, however, infers the routes from data, which is necessary if routes are dynamically changing over time, or not even known (such as during crisis management, where population movements are unforeseen, perhaps on a national scale [5]). Automatically being able to infer common mobility patterns from existing network data is also valuable in several other application areas, such as for facilitating traffic and transport planning, to improve emergency response, or for enabling more accurate empirically grounded epidemic models.

Furthermore, mining mobility patterns has computational advantages. In order to cope with the massive amount of mobility data available the data has to be coarse-grained for viable storage and analysis. The method presented in this paper enables projection of raw Cell ID sequences onto substantially smaller sets of common routes. These projections also result in anonymisation, cf. [10], as information about individual sequences is removed and hence subscribers may "hide in the crowd" [1, 12].

## 2 Methods

**Data Set** A mobile communication network consists of cells, where each cell is given by a geographic area covered by a base station. Cells in turn are grouped into location areas. When a terminal moves between location areas, the new location area and the id of the new cell is communicated to the network. Information about cell handovers within a location area, however, is not transmitted to the network. The current Cell ID is also known by the network when the terminal is used, e.g. during a call or when messaging. The network also acquires cell information by occasionally paging the terminal. This is done periodically, but infrequently, such as once per hour. Since each cell is correlated with geographic position, one may use sequences of Cell IDs to represent geographic trajectories. Such a sequence is denoted $S = (c_1, c_2, ..., c_a)$, where $c_i$ is the $i$:th Cell ID, and $a$ is the length of the sequence.

To enable experiments on empirical data, Cell ID sequences have been collected using terminals that record cell handovers. Handovers within location areas are also acquired. To simulate data as seen by the network, sequences are therefore sparsened by only keeping Cell IDs that occur immediately after a location area change. This corresponds to the "worst case" scenario when a terminal is in idle mode and not being paged.

**Relating Sequences** The strategy is to identify common routes by clustering Cell ID sequences under the assumption that two trajectories that share the same route have similar Cell ID sequences. When relating sequences to each other the order of Cell IDs is disregarded and only Cell ID occurrences are considered. The

similarity between two sequences $S_i$ and $S_j$ is quantified as the Jaccard index $J_{ij} = |\mathcal{C}_i \cap \mathcal{C}_j|/|\mathcal{C}_i \cup \mathcal{C}_j|$, where $\mathcal{C}_i$ and $\mathcal{C}_j$ are the sets of occurring Cell IDs in $S_i$ and $S_j$, respectively. $0 \leq J_{ij} \leq 1$, where $J_{ij} = 0$ when $S_i$ and $S_j$ do not share any Cell IDs, and $J_{ij} = 1$ when the two sequences have every Cell ID in common.

**Locality-Sensitive Hashing** An exhaustive pairwise comparison of sequences is not feasible due to the potentially huge number of comparisons required. Instead sequences are related using a technique based on the Min-hash [8] and locality-sensitive hashing [16] schemes. Let $\mathrm{h}(\cdot)$ be a pseudo-random uniform hash function that maps a Cell ID to a unique integer $i \in \{1, 2, ..., d\}$, where $d$ is the number of distinct cells. Furthermore, let $\mathrm{H}(\cdot)$ be a function that maps a Cell ID set to the minimum hash value of the elements in the set: $\mathrm{H}(\mathcal{C}) = \min_{c \in \mathcal{C}} \mathrm{h}(c)$. Then $\mathrm{H}(\cdot)$ has the convenient property that $\mathrm{Prob}[\mathrm{H}(\mathcal{C}_i) = \mathrm{H}(\mathcal{C}_j)] = J_{ij}$ [8]. This property can be utilised to identify related sequences simply by putting each sequence $S_i$ in a bucket given by $\mathrm{H}(\mathcal{C}_i)$. Sequences in the same buckets are then likely to have a high degree of Jaccard similarity. However, this approach results in many false positives, such that buckets contain unrelated sequences. To remedy this, the *two* smallest hash values of a sequence, given by a function $\mathrm{H}'(\cdot)$, specify its bucket. That is, $\mathrm{H}'(\mathcal{C}) = (\mathrm{H}(\mathcal{C}), \mathrm{H}(\mathcal{C}'))$, where $\mathcal{C}' = \mathcal{C} \setminus \{c\}$ for element $c$ such that $\mathrm{H}(\mathcal{C}) = \mathrm{h}(c)$. Then

$$\begin{aligned} &\mathrm{Prob}[\mathrm{H}'(\mathcal{C}_i) = \mathrm{H}'(\mathcal{C}_j)] = \\ &\mathrm{Prob}[\mathrm{H}(\mathcal{C}_i) = \mathrm{H}(\mathcal{C}_j)] \cdot \mathrm{Prob}[\mathrm{H}(\mathcal{C}'_i) = \mathrm{H}(\mathcal{C}'_j)] \approx \\ &\mathrm{Prob}[\mathrm{H}(\mathcal{C}_i) = \mathrm{H}(\mathcal{C}_j)]^2 = J_{ij}^2, \end{aligned} \tag{1}$$

for $|C| \gg 2$, since $\mathrm{h}(\cdot)$ is pseudo-random uniform hash function[1]. $\mathrm{H}'(\cdot)$, termed a hash signature, drastically reduces the number of false positives, but will also result in numerous false negatives, where similar sequences have different hash signatures. To reduce the number of false negatives the procedure is repeated $n$ times with different hash functions, $\mathrm{H}'_i$, and at each iteration the pairs of sequences that have the same $\mathrm{H}'_i$ value are noted.

**Graph Clustering** Sequences are related in a graph, where the weight of an edge between two vertices, constituted by sequences, is given by the number of times the corresponding sequences have the same $\mathrm{H}'_i$ value. The procedure for building a sequence graph is described in pseudo-code in Algorithm 1. Given the graph, sequences are grouped into common routes using a graph clustering algorithm. There are several such algorithms that can efficiently handle huge networks, c.f. [18]. Here the Louvain method is used [7], which is a good compromise between accuracy and computational complexity. The method hierarchically clusters vertices by iteratively forming higher order clusters that in turn are clustered; see ref. [7] for details.

[1] Another approach is to use two different hash functions [8]. This is more accurate, but also less efficient since one has to hash twice.

**Algorithm 1** Sequence graph construction. Graph $G$ is initialised to an ordered pair of vertex and edge sets, $(\{S_i\}_{j=1}^{m}, \mathcal{E})$, where $m$ is the number of sequences, and $\mathcal{E}$ is the set of all possible edges, where each edge has weight 0.

```
Initialise graph G
for i = 1 to n do
    Clear buckets.
    for all sequences S_j do
        Put S_j in bucket H'_i(C_j)
    end for
    for all buckets b do
        for all unordered pairs (S, S') such that S, S' ∈ b and S ≠ S' do
            Increment weight of edge (S, S') ∈ E with 1
        end for
    end for
end for
```

## 3 Results

**Accuracy** A set consisting of 87 GPS tagged sequences, denoted $\mathcal{V}$, is used to evaluate the accuracy of the method. The sequences are taken from three main routes (with 27, 29 and 31 sequences) given by a highway, a smaller road and a railroad. The GPS trajectories of the sequences are shown in Figure 1. Since it is given which route each sequence belongs to, one may compare the correct partition of $\mathcal{V}$ with the partition given by the method. The degree of accuracy is quantified by the Rand index [22], a similarity measure between partitions. This index, denoted r, is the fraction of pairs of elements where both elements are either in the same partition element in both partitions (counting to $c_{00}$), or where both are *not* in the same partition elements in both partitions (counting to $c_{11}$). That is,

$$\mathrm{r} = (c_{00} + c_{11})\binom{m}{2}^{-1} \tag{2}$$

where $m$ is the number of sequences. $\mathrm{r} \in [0, 1]$, where $\mathrm{r} = 1$ when two partitions are identical. As expected the accuracy improves with increasing number of hash signatures since larger $n$ results in a more accurate sequence graph. The choice of $n$ is therefore a trade-off between speed and accuracy, where the latter increases rapidly for small $n$ and then approaches convergence, cf. Figure 2. The method performs well, reaching an accuracy of $\mathrm{r} \approx 0.87$ for a moderate number of hash functions. In other words, sequences are equally related in both partitions (in terms of belonging to the same partition element or not) in approximately 87% of all possible sequence pairs.

**Scalability** Since the graph clustering algorithm used is known to be scalable [7], we turn our attention to the scalability of Algorithm 1. The runtime of generating a similarity network is measured for different sequence set sizes. Again the

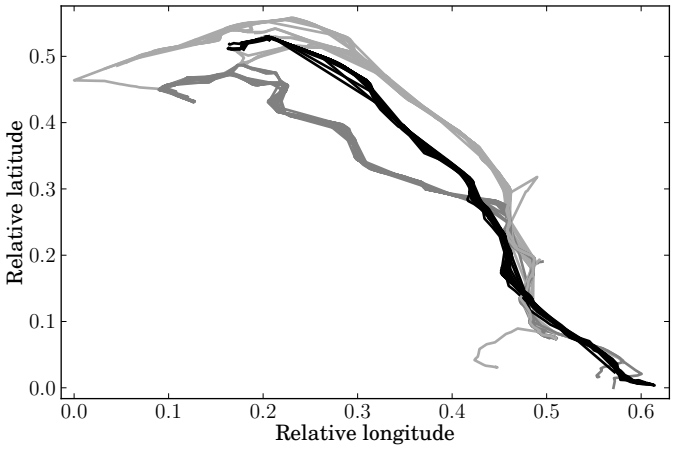


**Fig. 1.** Evaluation set: 87 GPS trajectories constituting three main routes (coded in black, grey and light grey).

GPS tagged sequences are used, in addition to 503 sequences that approximately start and end in the same areas as the GPS sequences.

At each of the $n$ iterations the algorithm does one pass through the sequences, where calculating the hash signature of a sequence $S_i$ takes $\mathcal{O}(|\mathcal{C}_i|)$ time, in total $\mathcal{O}(q)$, where $q = \sum_{i=1}^{m} |\mathcal{C}_i|$ for all sequences. The runtime of updating edge weights scales with the number of pairs of sequences that have the same hash signature, which has an expectation value of $u$. Since there are $n$ iterations, the total runtime of Algorithm 1 is $\mathcal{O}(n(q+u))$. Since $q \gg u$, the runtime is dominated by $q$ at each iteration. This can also be seen in Figure 3, where the runtime scales almost linearly with the number of sequences $m$ for a given number of iterations, implying that $q \sim m$. Note that the number of required iterations $n$ does not grow with increasing $m$ in order to build an accurate similarity network, since the probability that two sequences have the same hash signature at a given iteration is independent of the number of sequences $m$.

## 4 Discussion

The method is scalable since one only considers relevant strong sequence similarities that are relatively limited in number compared to all possible sequence relations. Although several hash functions are utilised in order to capture most strong sequence relations, there may still be false negatives due to the stochasticity of the method. However, even though a strong link between two sequences is missed, it is likely that it is indirectly captured in the graph representation, since sequences with a high Jaccard similarity often also share several neighbour-

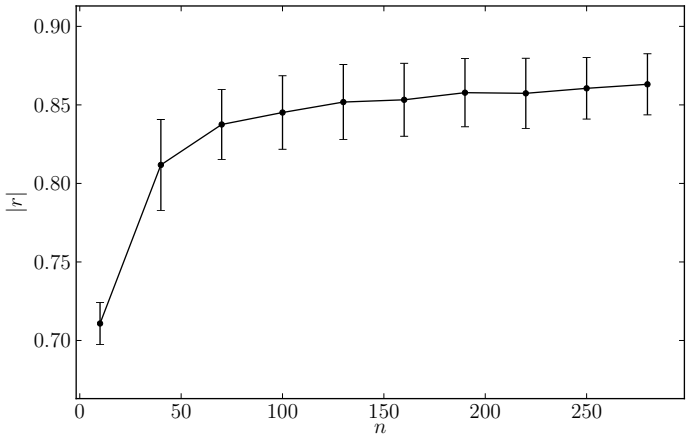


**Fig. 2.** Mean accuracy for different number of hash signatures, n. Standard deviation displayed by error bars.

ing nodes. The reason for this is that the Jaccard distance, denoted and defined as $J_D = 1 - J$, is a proper metric. This results in that the graph clustering algorithm can correctly assign two sequences to the same route, even though the sequences are only implicitly linked.

Although it is demonstrated that the method provides a high degree of accuracy, a few caveats are in place regarding data. Firstly, the method has been evaluated using a dataset collected by the author, coworkers and students – a sample of people that certainly does not represent a cross section of all cell phone users. However, this bias is also an advantage since it enables us to extract a larger evaluation set between a given origin and destination given the limited data available. That is, if the data would be an equally sized uniform sample of Cell ID sequences, very few sequences would be available between a given origin and destination. The second caveat is to note that the method is evaluated on a single set of routes at a given length scale and density of base stations. Further studies are required in order to evaluate the methods performance under other conditions, such as when base stations are more sparsely distributed and trajectories shorter.

Since calculating the hash signatures of sequences can be done independently for each sequence, the method is suitable for parallelisation. One could for instance further speed up calculations by using the MapReduce scheme [11], where the *map* function calculates the hash signatures, and where sequences with identical signatures are aggregated (i.e. put in the same bucket) in the *reduce* step.

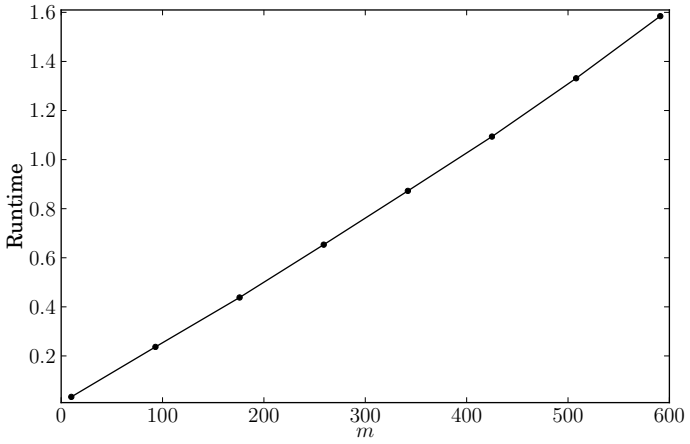


**Fig. 3.** Runtime in seconds (of a Python implementation run on a laptop) of Algorithm 1 for $n = 200$ and different number of sequences.

## 5 Conclusions

To summarise, a probabilistic method for grouping Cell ID sequences from mobile communication networks into common routes has been presented. Sequences are related in a graph, where edge weights are given by locality-sensitive hashing. The method has been demonstrated to be scalable, and to accurately group sequences in an evaluation set of GPS tagged sequences. Even though further investigations are required in order to evaluate the methods performance under conditions not captured by the evaluation set, such as for other types of mobility patterns or on other length scales, the results presented here are highly encouraging.

**Acknowledgements** The author thanks the participants of the Consider8 project for fruitful discussions, John Krumm and the anonymous reviewers for valuable feedback, and everybody that contributed with the Cell ID data used in this paper. This work was supported by the Swedish Governmental Agency for Innovation Systems (VINNOVA).

## References

1. Abul, O. et al.: Never walk alone: Uncertainty for anonymity in moving objects databases. In: In ICDE. pp. 376–385. IEEE (2008)
2. Alec P. et al.: Anomaly detection in a mobile communication network. In: Proceedings of the NAACSOS. pp. 407–422 (2006)

3. Ashbrook, D., Starner, T.: Learning significant locations and predicting user movement with gps. In: ISWC. pp. 101–108 (2002)
4. Becker, R. A. et al.: Route classification using cellular handoff patterns. In: Proceedings of the 13th international conference on Ubiquitous computing. pp. 123–132. UbiComp '11, ACM (2011)
5. Bengtsson, L. et al.: Improved Response to Disasters and Outbreaks by Tracking Population Movements with Mobile Phone Network Data: A Post-Earthquake Geospatial Study in Haiti. PLoS Med 8(8) (2011)
6. Bhattacharya, A., Das, S.: LeZi-update: an information-theoretic approach to track mobile users in PCS networks. In: MobiCom '99. pp. 1–12 (1999)
7. Blondel, V.D. et al.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008(10) (2008)
8. Broder, A.Z. et al.: Min-wise Independent Permutations. Journal of Computer and System Sciences 60, 327–336 (1998)
9. Candia, J. et al.: Uncovering individual and collective human dynamics from mobile phone records. Journal of Physics A: Mathematical and Theoretical 41(22) (2008)
10. De Mulder, Y. et al.: Identification via location-profiling in GSM networks. In: WPES '08. pp. 23–32 (2008)
11. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. pp. 137–150
12. Domingo-Ferrer, J. et al.: Privacy-preserving publication of trajectories using microaggregation. pp. 26–33. SPRINGL '10 (2010)
13. González, M.C. et al.: Understanding individual human mobility patterns. Nature 453(7196), 779–782 (2008)
14. Hightower, J. et al.: Learning and Recognizing the Places We Go UbiComp 2005: Ubiquitous Computing. vol. 3660, p. 903 (2005)
15. Ian S. et al.: Algorithms for detecting motion of a gsm mobile phone. In: In ECSCW 2005 (2005)
16. Indyk, P., Motwani, R.: Approximate nearest neighbors: Towards removing the curse of dimensionality. In: STOC. pp. 604–613 (1998)
17. Laasonen, K. et al.: Adaptive On-Device Location Recognition. In: Pervasive Computing, vol. 3001, pp. 287–304 (2004)
18. Lancichinetti, A., Fortunato, S.: Community Detection Algorithms: A Comparative Analysis. Physical Review E 80(5), 056117+ (2009)
19. Liao, L. et al.: Learning and inferring transportation routines. Artif. Intell. 171, 311–331 (2007)
20. Massey, W., Whitt, W.: A stochastic model to capture space and time dynamics in wireless communication systems. Probability in the Engineering and Informational Sciences 8, 541–569 (1994)
21. Patterson, D. et al.: Inferring High-Level Behavior from Low-Level Sensors. vol. 2864, pp. 73–89 (2003)
22. Rand, W.: Objective Criteria for the Evaluation of Clustering Methods. Journal of the American Statistical Association 66(336), 846–850 (1971)
23. Sohn, T. et al.: Mobility Detection Using Everyday GSM Traces. pp. 212–224 (2006)
24. Song, C. et al.: Limits of predictability in human mobility. Science 327(5968), 1018–1021 (2010)
25. Song, C. et al.: Modelling the scaling properties of human mobility. Nature Physics 6(10), 818–823 (2010)
26. Yavas, G. et al.: A data mining approach for location prediction in mobile environments. Data Knowl. Eng. 54, 121–146 (August 2005)