Group Detection in Mobility Traces*

Yung-Chih Chen
UMass Amherst, USA
yungchih@cs.umass.edu

Elisha Rosensweig
UMass Amherst, USA
elisha@cs.umass.edu

Jim Kurose
UMass Amherst, USA
kurose@cs.umass.edu

Don Towsley
UMass Amherst, USA
towsley@cs.umass.edu

ABSTRACT

In a number of network scenarios (including military settings), mobile nodes are clustered into groups, with nodes within the same group exhibiting significant correlation in their movements. Mobility models for such networks should reflect this group structure. In this paper, we consider the problem of identifying the number of groups, and the membership of mobile nodes within groups, from a trace of mobile nodes. We present two clustering algorithms to determine the number of groups and their identities: *k-means chain* and *spectral clustering*. Different from traditional k-means clustering, k-means chain identifies the number of groups in a dynamic graph, using a chaining process to keep track of group trajectories over the entire trace. The second approach uses spectral clustering, which uses similarities between node pairs to cluster nodes into groups. We show that the number of groups and node membership can be accurately extracted from traces, particularly when the number of groups is small.

Categories and Subject Descriptors

C.2 [**Computer communication networks**]: Misc.; I.5.3 [**Clustering**]: Algorithms and similarity measures

General Terms

Performance

*This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence and was accomplished under Agreement Number W911NF-06-3-0001. The views and conclusions contained in this document are those of the author(s) and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

Keywords

Group mobility, clustering, performance

1. INTRODUCTION

The modeling and analysis of wireless networks greatly depends on the specific rules that govern the movement of the nodes. One class of mobility model is that of group-oriented or leader-oriented movement, in which the set of moving nodes is divided into several disjoint groups, where the movement of individual nodes within each group is highly correlated. By separating out groups from each other, a more precise analysis of intra-group behavior can be achieved, and inter-group mobility correlation can be observed and analyzed. This group structure can also be used in generative models of group-based node mobility [5].

In this work we present two approaches for dividing mobile units into groups, given a set of candidate values k in advance, with the goal of determining the most suitable value of k and the group memberships. We analyze the ability of these approaches to correctly identify the number and membership of groups, using both real and synthetic mobility traces. Here, a trace consists of a series of snapshots of nodes' locations. In the field of machine learning, clustering tools are most commonly used to assign each point in a data set to one of a set of clusters, the number of which is assumed be known. When the number of clusters is unknown at the outset, selecting the "correct" number of clusters is a difficult problem.

The first approach we propose for group identification, *k-means chaining*, fixes the number of groups at k for each candidate value k, and clusters nodes into k groups at each snapshot. By counting the number of nodes' changing groups between consecutive snapshots, the number of groups is selected to minimize the number of such group changes. The trajectories of the corresponding groups are also produced by the k-means chaining algorithm. The second approach we investigate uses *spectral clustering* methods to group the nodes for each candidate value of k, and uses *clustering stability* measures to determine which value of k best models the system.

The remainder of this paper is structured as follows. After reviewing k-means clustering in Sec. 2, we describe the two different methods used to determine the number of groups in the trace and membership of nodes in these groups in Sec. 3. Then, in Sec. 4 we present the traces used to study and evaluate these two methods. Sec. 5 is devoted to a detailed discussion of the range of tests we ran to evaluate our methods, and we conclude with Sec. 7 where we summarize our

findings and present ideas for future research.

2. K-MEANS CLUSTERING

In the fields of statistics and machine learning, clustering analysis is usually regarded as the problem of dividing a set of objects into two or more groups in such a way that elements in a group are similar to each other in some predefined manner. A popular approach to clustering is the k-means clustering algorithm [1], which is a special case of the classic EM clustering algorithm. In this work, we rely heavily on this algorithm, and so we proceed to define it formally.

k-means clustering is a method used to partition a data set of n points into k different clusters such that similar elements are grouped together. Given k, a set of points $\{x_1, x_2, ..., x_n\}$, and a distance metric, k-means clustering partitions the n points into k different sets: $S = \{S_1, S_2, \ldots, S_k\}$ so as to minimize the sum of squares of distance from the points to the mean μ_i of the distance of each cluster S_i:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x_j \in S_i} \|x_j - \mu_i\|^2 \quad (1)$$

The values μ_i are referred to commonly as the cluster *centroids*. Given k and a (possibly random) starting position of the centroids, k-means uses an iterative procedure to try to determine the μ_i that satisfy (1). There are two concerns here selecting the correct value for k (a parameter that is sometimes known in advance) and dealing with the fact that k-means can select values of μ_i values that result in a local minimum rather than a global minimum.

3. MOBILE GROUP CLUSTERING

In this section we present our two approaches to partition the nodes in a mobility trace into groups, without *a priori* knowledge of the number of groups, k. Our first approach, *k-means chaining*, first selects a value for k and then assigns each node to one of the k groups at each snapshot[1] and then selects the value of k that minimizes the number of times that nodes are partitioned into a different group. The second approach uses *spectral clustering* to generate clusterings for various values of k, and then chooses the value of k most likely to be correct by selecting the clustering exhibiting the greatest *clustering stability*.

3.1 k-means Chains

We begin by introducing the concept of *loyalty* in mobile groups. For expository purposes, we use the example of mobile military units, where each soldier is a unit. Here, each soldier (a mobile unit) is assigned to a group, and remains with this group permanently. We say this soldier is loyal to the group. Thus, consider an algorithm that would apply k-means to each snapshot in a mobility trace for a set of possible k values. For each snapshot a clustering would be produced. If an incorrect value of k was used, units would show a significant number of group changes, and is referred to as *disloyalty* between snapshots. The disloyalty function is defined by enumerating the number of times each unit changes groups between snapshots for a set of possible values of k. We thus choose the number of groups to the value of k for which disloyalty is a minimum[2].

[1]A snapshot is the trace of a specific second, containing information of time stamp, node ID, x, and y coordinates.

[2]We ignore the cases of $k = 1$ or n (no group changes occur).

Though useful in theory, comparing clusterings between snapshots is a complicated and time-consuming process. First, in the standard k-means implementation, the centroids are randomly distributed at the beginning of the clustering process for a given snapshot; in our scenario this can lead to *inconsistent labeling* between snapshots (e.g., the same group can be labeled as group 1 at time t and group 5 at time $t+1$). Due to this inconsistent labeling, clustering comparison is difficult. Specifically, a brute-force approach would require iterating over all possible $k!$ label permutations in search of the best label-alignment between clusterings that minimizes the differences between them. Additionally, a more fundamental challenge is that group identity is not well-defined: if two units were in the same group at time t but not at time $t+1$, which of the two should be considered to have changed its group? Any answer to this question can be, at most, a heuristic, and open for interpretation and dispute.

To address both of these problems, we initialize the k-means algorithm so that the starting position (at time $t+1$) of the centroid of cluster j is close to where we expect cluster j from the *previous* snapshot to have moved to by $t+1$. To do this, we can simply use the centroid location computed for cluster j at time t as the starting position for cluster j at time $t+1$. In this paper, we use a more practical way to initialize the starting position of each group by forcing the centroid of each cluster to be one of the nodes closest to the computed center location. This is because, in some cases, nodes in a group may suddenly move very fast from one point and arrive at another, say a hot spot, at time $t+1$, resulting in the situation that no node at $t+1$ is near the centroid position at time t.

We refer to such sequences of k-means as a "k-means chain", as the behavior at every snapshot is determined by the previous one, creating a dependency chain between snapshots. This method can be thought of as extending the classic k-means algorithm from solely clustering nodes in a single static snapshot, to the case of dynamic, sequential snapshots. Using this approach, we use the following approach for selecting the best value of k: we count the total number of times that nodes change their *assigned group label*, and select the k that minimizes this value. For our implementation here we sum up the number of group label changes for all nodes across snapshots, and refer to this as the combined *disloyalty* of all the units/nodes. The motivation for this algorithm is that since units are loyal to their groups, the disloyalty should be low for the correct value of k, where group changes will occur only in borderline cases when two groups are close to one another.

This technique can also be used to keep track of the movement of group centroids over the trace. Since we know group centroids' movement, during each snapshot, the trajectory of each group can be identified by the k-means chaining process. Furthermore, if the correct value of k is given, nodes' group affiliation can also be identified in an online fashion. For the first snapshot, k-means clustering returns k groups and node affiliation (i.e., each node will be given an assignment of group affiliation: g, where $1 \leq g \leq k$). After setting the start positions for the next snapshot described previously, nodes of the same group should ideally remain in that same group for the entire length of the trace, showing loyalty to its group. A pseudo code for k-means chain algorithm can be found in [10].

3.2 Spectral Clustering

Spectral clustering methods [2] accept as input a set of objects $\{x_1, \ldots, x_n\}$ for which there exists a pairwise similarity (or weight) matrix $S = (s_{ij})$, $i, j \in \{1, \ldots, n\}$, representing the similarity between every pair of objects i and j. From S we extract a graph Laplacian matrix L, and determine the k eigenvectors of L with the lowest eigenvalue, denoted by $e_1, ..., e_k$. These are arranged as columns of a matrix E_k, and each object x_i is mapped to the i^{th} row in E_k, denoted y_i, s.t. $y_i \in \mathbb{R}^k$. Finally, we run k-means over the points $\{y_i\}_{i=1}^n$, generating the desired k groups. The challenges in applying spectral clustering lie in *(i)* determining the value of k, *(ii)* selecting a useful and expressive similarity function, and *(iii)* deciding whether the matrix should contain all the similarity information or just a section of it [2].

We defer our discussion of these issues until Sec. 5, and for now explain how we use *clustering stability* [4] to choose the most suitable value of k. We say that a clustering is *stable* if it is insensitive to the specific sample from the sample space that is input to the clustering algorithm, for samples of equal size. For the case at hand, for each unit we have a sample space consisting of its movement over the length of the trace, and a sample is a part of the trace. Assuming the mobility pattern of the unit does not change between samples, including its association with its group, then the clustering into groups for the "correct" value k would ideally be insensitive to the specific sample used to generate the clustering (for large enough samples), and in such a case we say the clustering is *stable*. Thus, for the problem at hand, we test the stability of a specific k by randomly dividing the available trace into $h \geq 2$ disjoint samples, run spectral clustering on each sample, and compare the resulting clusterings. If the clusterings match, this indicates that the clustering is *stable*, increasing the confidence that k is the actual number of clusters[3].

Using this concept of clustering stability, we run spectral clustering over two equal and disjoint portions of the trace, and compare the results to check that the clusters for each run are the same. Note, however, that clustering stability assumes that the different samples are taken from *the same sample space*. In real-world mobility traces, groups change their behavior patterns over time even if their groups remain intact. For this reason, each sample is a concatenation of small samples (i.e., consecutive snapshots) taken over *entire* trace, so that the behavior of each node is similar in both samples. A more detailed description and pseudo code for spectral clustering is in [10].

4. TRACE DATA

In this section we discuss the traces we used to evaluate our two group-detection algorithms. We use both synthetic traces generated by a mobility model, and a "real-world" trace of a military scenario. Note that we also apply our algorithms on the traces of other scenarios, such as campus area and vehicular networks, as case studies [10].

4.1 Reference Point Group Mobility (RPGM)

In the RPGM model, each group has a logical center (i.e., the leader) whose motion defines the behavior and the trajectory of the whole group [3]. We generate RPGM traces using the *IMPORTANT* tool [5]. *IMPORTANT* takes a compressed description of group leader(s) movement and outputs a full mobility trace for all nodes in all groups. The leader trace consists of a series of checkpoints visited by the leader and the speed at which it moves between successive checkpoints. The resulting trace contains snapshots every second, giving the location of each node at each snapshot. In this paper, we generated 50 RPGM traces for $k = 2, ..., 9$, where each group is comprised of 5 nodes. We fixed the speed deviation and angle deviation in IMPORTANT [5] to a default value[4].

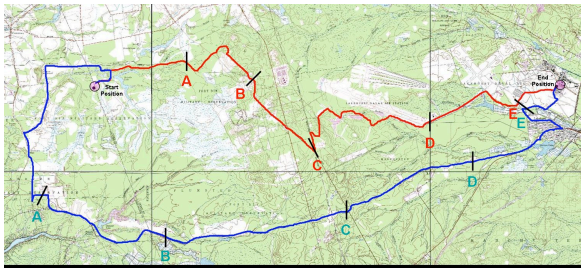

Figure 1: Map of Lakehurst Scenario

4.2 Lakehurst Military Trace

This trace, supplied by the US Army Research Lab (ARL), records the movements and communication between units in a realistic military maneuver that took place at Lakehurst, New Jersey USA. The maneuver was 3 hours long and consisted of 64 ground vehicles, each belonging to one of 9 groups, referred to in this paper as military units, MUs for short (The number and membership of each of these groups was determined by human observation). Vehicles were equipped with GPS receivers so that they could record their positions at intervals of one second. All 9 MUs started and ended the maneuver at same locations, but were divided into two meta-groups, with each following a different route from the starting point to the end point shown in Fig. 1. Along this route there were several checkpoints, with each MU stopping at every checkpoint for some period before moving on to the next checkpoint.

5. TESTS AND RESULTS

We evaluate our group-identification algorithms using both synthetic and military traces. We compared the results of these algorithms to the known correct groups in the traces. In the case of synthetic traces, k was an input parameter. Here in the cast of the military trace, the number of groups was determined by the authors by viewing the movement of groups visually using a mobility simulator.

We define *group number accuracy* as the fraction of the traces where k is correctly chosen by the algorithm. Also, for the correct value of k, we compute the fraction of nodes that are assigned to the correct group in each snapshot, and then determine the average of this fraction over all snapshots in the trace as *group member accuracy*. In the case of RPGM with 5 groups ($k = 5$), each of size 5, 25 nodes are associated to 5 groups. If 22 of these 25 nodes are assigned to the

[3]For discussion regarding the theoretical usefulness of stability as a tool for selecting k, see [6] and [7].

[4]The default value of speed deviation is 5m/sec, and 90° for angle deviation. All simulations were done in a 1000m× 1000m plane. We leave detecting groups under different speed and angle deviation levels as future work.

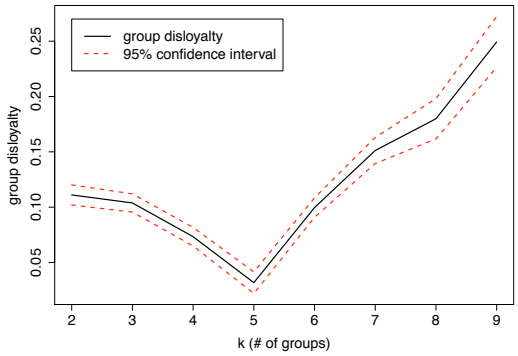

Figure 2: K-means chain minimizes the group disloyalty at $k = 5$, for 50 RPGM traces of 5 groups.

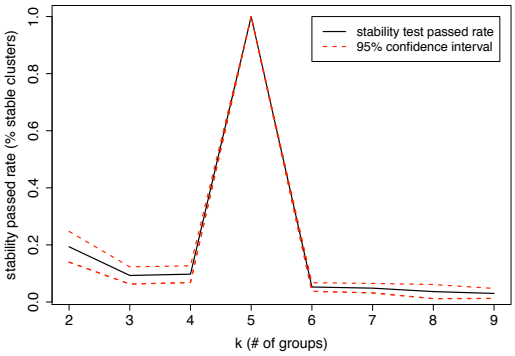

Figure 4: Spectral clustering stability test for 50 RPGM traces of 5 groups.

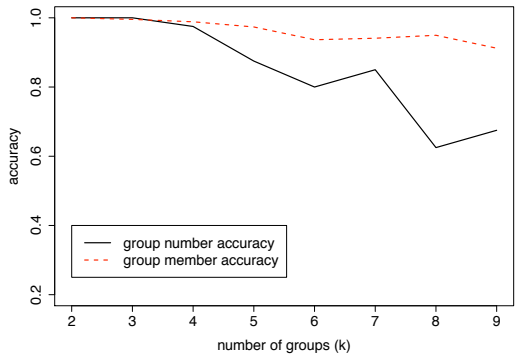

Figure 3: The accuracy of k-means chaining in determining the correct number of groups, and assigning nodes to groups.

correct group during a snapshot, then the accuracy for that snapshot is $\frac{22}{25}$. For spectral clustering, the accuracy of the group member detection is defined as the percentage that nodes are correctly labeled after applying k-means on the similarity matrix.

5.1 K-means chain

As described earlier, the k-means chain algorithm is a heuristic that determines the number of groups in the trace as that number of groups which minimizes the number of *group changes* over the course of the trace. Since this algorithm is just a heuristic, it is of interest to study how well this heuristic performs in practice. In order to capture the spatial and temporal relationships in group mobility, in this paper, we run k-means chain over the data consisting of nodes' geographical positions and velocities (i.e., a node's x, y coordinates and its associated velocity: V_x and V_y).

Fig.2 shows the disloyalty for different values of k for synthetic traces. The disloyalty for a given value of k is the total number of times nodes change group under the clusterings computed at each snapshot, normalized by the sum of all such changes for all values of k. Fig.2 shows that $k = 5$ clearly has the minimum disloyalty; recall that this is precisely the number of groups used by the RPGM trace generator.

The performance of k-means chain over all synthetic traces is presented in Fig.3. Here, we see that k-means chaining is most accurate when there are a smaller number of groups. Its accuracy degrades a bit when the number of groups in the traces increases. This is due to the fact that the simulation environment is constrained to a 1000m×1000m plan in the IMPORTANT tool. When the number of groups is large, groups tend to mix with each other during their movement from one checkpoint to another. This happens especially when two or more groups are moving in opposite directions. Cases like this one make it difficult for k-means chaining to identify groups precisely in one snapshot, and results in a performance degradation. On the other hand, k-means chaining does a good job identifying members of each group when a correct value of k is given. For $2 \leq k \leq 9$, the accuracy of assigning a node to a correct group is above 0.9.

The results for the military trace, shown in Fig.5, indicate that the disloyalty function minimizes the group changing frequency at $k = 9$. This shows the existence of nine operational groups, exactly the number of groups in the scenario described in Sec. 4.

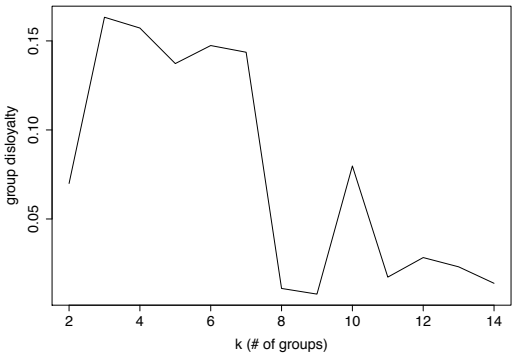

Figure 5: K-means chain with military trace, where the group disloyalty function is minimized at $k = 9$, indicating the existence of nine groups.

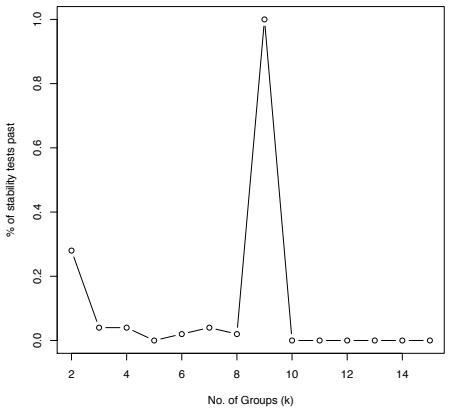

Figure 6: Clustering stability with spectral clustering, for the military trace. Support for a specific k is determined by the number of stability checks that indicate stable clustering.

5.2 Spectral Clustering

To test the spectral clustering approach for solving this problem, we divided each trace into two equal-length disjoint portions, constructed by "concatenation of alternating samples of length 10, taken over entire trace (example see Sec. 3.2). For these samples, we used spectral clustering on each sample 50 times and compared clusterings across samples each time. We then selected as the most suitable k the one for which the clusterings were stable the largest fraction of time. The need to repeat this test more than once arises from the fact that spectral clustering uses k-means, an algorithm that is prone to converging to a local rather than a global minimum. By running spectral clustering several times over the same samples, we can distinguish between the k values that are consistently bad and those that behave well most of the time.

We analyzed two functions for generating the similarity matrix needed by the spectral clustering algorithm: the *mean distance* and *distance variance* between every two units. The first metric was chosen since units in a group consistently remain close together, while the second reflects the fact that the direction of movement and speed within each group are strongly correlated[5]. For both synthetic and real-world traces (Fig. 4 and 6 respectively), the value of k for which the clustering was stable the most times compared to the correct number of groups, and the group association of units was correct every time the clusterings were stable. Furthermore, as indicated in these figures, for incorrect values of k, stability checks failed at least 70% of the time, but almost never for the correct k, indicating that stability is a useful measure for good k selection in this field.

6. RELATED WORK

Research of group mobility mostly follows the *Reference Point Group Mobility (RPGM)* model [3] used in this paper. Some of RPGM's variants, such as the more restrictive *Structured Group Mobility (SGM)* model [8] is also presented. A very similar work we are aware of that attempted to discover groups in a mobility trace is that of Wang et al [9], which presented a partition algorithm called *Sequential Clustering*. Details about the constraints of this algorithm and related comparisons are addressed in [10].

[5]Other metrics might be equally useful. However, note that care needs to be taken when choosing the metric, specifically because trace portions are non-contiguous, and this can cause metrics that are computed over several snapshots at once to malfunction.

7. SUMMARY AND CONCLUSIONS

In this paper, we have presented two algorithms - k-means chains and spectral clustering for identifying the number of groups, and the membership of mobile nodes within groups, in a trace of mobile network nodes We evaluated these algorithms using both synthetically-generated traces with a known group structure and a real-world trace from a military scenario. We found that both algorithms were almost always able to identify the number of groups correctly, and assigned nodes to groups fairly accurately. Accuracy tended to decrease and the number of groups operating within a fixed-size domain became larger.

Much work still remains to be done. It is of interest to understand how the accuracy of group detection changes as the time between snapshot changes (i.e., as the system state changes more between snapshots), and as group structure becomes "looser" (e.g., as a node's affinity for its group leader decreases). We are interested in investigating new group-detection algorithms as well.

8. REFERENCES

[1] J.B. MacQueen, "Some Methods for classification and Analysis of Multivariate Observations," *Berkeley Symposium on Mathematical Statistics and Probability, 1967*

[2] U. von Luxburg, "A Tutorial on Spectral Clustering," *Tech. Report No. TR-149, Max Planck Institute for Biological Cybernetics, 2006*

[3] X. Hong, M. Gerla, G. Pei, C-C. Chiang, "A Group Mobility Model for Ad Hoc Wireless Networks," *IEEE MSWiM, 1999*

[4] T. Lange, V. Roth, M. L. Braun, J. M. Buhmann, "Stability-Based Validation of Clustering Solutions," *Neural Computation, vol. 16, pp. 1299-1323, 2004*

[5] F. Bai, N. Sadagopan, A. Helmy, "The IMPORTANT framework for analyzing the Impact of Mobility on Performance Of RouTing protocols for Adhoc NeTworks," *Ad Hoc Networks, Elsevier 2003*

[6] S. Ben-David , U. von-Luxburg and D. Pńal, "A Sober Look at Clustering Stability," *the 19th Annual Conference on Computational Learning Theory, 2006*

[7] O. Shamir and N. Tishbi, "Cluster Stability for Finite Samples," *Advances in Neural Information Processing Systems (NIPS) 2007-8*

[8] K. Blakely and B. Lowekamp, "A Structured Group Mobility Model for the Simulation of Mobile Ad Hoc Networks," *ACM International Workshop on Mobility Management and WIreless Access (MobiWac), 2004*

[9] K. H. Wang, and B. Li, "Group Mobility and Partition Prediction in Wireless Ad-Hoc Networks," *IEEE International Conference on Communications (ICC), vol. 2, pp. 1017-1021, 2002*

[10] Y.-C. Chen, E. Rosensweig, J. Kurose, and D. Towsley, "Group Detection in Mobility Traces," Tech. Rep. Department of Computer Science, UMass Amherst *http://gaia.cs.umass.edu/networks/papers/GroupDetect10.pdf*