University of Louisville

ThinkIR: The University of Louisville's Institutional Repository

Electronic Theses and Dissertations

5-2018

Network science algorithms for mobile networks.

Heba Mohamed Elgazzar
University of Louisville

Follow this and additional works at: <https://ir.library.louisville.edu/etd>

Part of the [Computer Engineering Commons](#), and the [Computer Sciences Commons](#)

Recommended Citation

Elgazzar, Heba Mohamed, "Network science algorithms for mobile networks." (2018). *Electronic Theses and Dissertations*. Paper 2997.
<https://doi.org/10.18297/etd/2997>

This Doctoral Dissertation is brought to you for free and open access by ThinkIR: The University of Louisville's Institutional Repository. It has been accepted for inclusion in Electronic Theses and Dissertations by an authorized administrator of ThinkIR: The University of Louisville's Institutional Repository. This title appears here courtesy of the author, who has retained all other copyrights. For more information, please contact thinkir@louisville.edu.

NETWORK SCIENCE ALGORITHMS FOR MOBILE NETWORKS

By
Heba Mohamed Elgazzar
B.Sc., Zagazig University, 1999
M.Sc., University of Louisville, 2007

A Dissertation
Submitted to the Faculty of the
J.B. Speed School of Engineering of the
University of Louisville
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy
in Computer Science and Engineering

Department of Computer Engineering and Computer Science
University of Louisville
Louisville, Kentucky

May 2018

NETWORK SCIENCE ALGORITHMS FOR MOBILE NETWORKS

By

Heba Mohamed Elgazzar
B.Sc., Zagazig University, 1999
M.Sc., University of Louisville, 2007

A Dissertation Approved on

April 6, 2018

by the following Dissertation Committee:

Dr. Adel Elmaghraby

Dr. Suraj Alexander

Dr. Nihat Altiparmak

Dr. Dar-Jen Chang

Dr. Ibrahim Imam

DEDICATION

This dissertation is dedicated to my dear late parents

Mr. Mohamed Elgazzar and Mrs. Sania Elgazzar

You were really the big support in my life. I live until
now with your prayers.

ACKNOWLEDGMENT

First, I am so thankful to Almighty God, Most Gracious, Most Merciful, without his blessing this research would not have been done. I would like to express my deepest thanks and appreciation to my advisor, Dr. Adel Elmaghraby, for his outstanding guidance, active support, cooperation, and encouragement to bring this work to success. My deepest thanks are extended also to Dr. Suraj Alexander, Dr. Nihat Altiparmak, Dr. Dar-Jen Chang, and Dr. Ibrahim Imam for serving on my dissertation committee.

I would like to thank my great late father (God rest his soul) and my great late mother (God rest her soul) for their prayers, encouragement, and support during my study and during all my life. My special thanks should be extended also to my great husband, Dr. Sherif Rashad, for his love, support, and encouragement during this research. I would like also to thank my lovely adorable sons, Mohammad and Khalid, who are really the joy and pleasure of my life. Finally, I would like to thank my brothers, Hani, Wael, and Ahmed for their love and support. And last but not least, I would like to thank my lovely aunts for their prayers, love, and support.

ABSTRACT

NETWORK SCIENCE ALGORITHMS FOR MOBILE NETWORKS

Heba M. Elgazzar

April 6, 2018

Network Science is one of the important and emerging fields in computer science and engineering that focuses on the study and analysis of different types of networks. The goal of this dissertation is to design and develop network science algorithms that can be used to study and analyze mobile networks. This can provide essential information and knowledge that can help mobile networks service providers to enhance the quality of the mobile services.

We focus in this dissertation on the design and analysis of different network science techniques that can be used to analyze the dynamics of mobile networks. These techniques include evolutionary clustering, classification, discovery of maximal cliques, and evolutionary centrality algorithms. We proposed evolutionary clustering and evolutionary centrality algorithms that can be used to dynamically discover clusters and central nodes in mobile networks. Overall, the

experimental results show that the proposed evolutionary algorithms are robust to short-term variations but reflects long-term trends and can be used effectively to analyze the dynamics of mobile networks.

TABLE OF CONTENTS

ACKNOWLEDGMENT.....	iv
ABSTRACT.....	v
LIST OF FIGURES.....	ix
CHAPTER I: INTRODUCTION.....	1
1.1 Background and Research Motivation	2
1.2 Goals and Objectives of Our Research	3
1.3 Plan of Research Activities	3
1.4 Organization of the Dissertation	5
CHAPTER II: BACKGROUND AND RELATED WORK.....	7
2.1 Network Science	7
2.2 Classification Algorithms	12
2.3 Evolutionary Clustering	17
2.4 MIT Reality Mining Dataset	23
CHAPTER III: PROPOSED EVOLUTIONARY CLUSTERING ALGORITHMS.	25
3.1 Evolutionary K-Medoid Clustering	25
3.2 Results of Evolutionary K-Medoid Clustering	29
3.3 Evolutionary DBSCAN Clustering	32
3.4 Results of Evolutionary DBSCAN Clustering	37
3.5 Analysis of Accuracy of K-Medoid results	40
3.6 Evolutionary Hierarchical Clustering	43
3.7 Results of Evolutionary Hierarchical Clustering ...	47
3.3 Analysis of Accuracy of Hierarchical Clustering	54
3.8 Working with Different Time Windows	55
CHAPTER IV: CLASSIFICATION ALGORITHMS IN MOBILE NETWORKS.	60
4.1 Classification Algorithms for Mobile Networks	60
CHAPTER V: MAXIMAL CLIQUES AND CENTRALITY IN MOBILE NETWORKS.....	65

5.1	Finding Maximal Cliques in Mobile Networks	65
5.2	Maximal Cliques Experimental Result	67
5.3	Centrality in Social Networks	76
5.4	Centrality Measures for Mobile Networks	81
5.4.1	Degree Centrality:.....	81
5.4.2	Closeness Centrality:.....	82
5.4.3	Betweenness Centrality:.....	82
5.4.4	Eigenvector Centrality:.....	84
5.4.5	PageRank Centrality:.....	85
5.5	Evolutionary Centrality	91
CHAPTER VI: CONCLUSION AND FUTURE WORK		96
REFERENCES.....		99
Appendix A: SUMMARY OF MIT REALITY MINING DATA FEATURES.		106
CURRICULUM VITAE.....		110

LIST OF FIGURES

Figure 1. Example of network representation.....	9
Figure 2. K-medoids Clustering Algorithm.....	26
Figure 3. Evolutionary K-medoids Clustering Algorithm....	28
Figure 4. K-medoids results for K=2.....	30
Figure 5. Evolutionary K-medoids results for K=2 and $\alpha=0.8$	31
Figure 6. DBSCAN Clustering Algorithm.....	33
Figure 7. Evolutionary DBSCAN Clustering Algorithm.....	36
Figure 8. Static DBSCAN results.....	38
Figure 9. Evolutionary DBSCAN Results.....	39
Figure 10. Accuracy Results of the K-Medoid.....	40
Figure 11. Average Accuracy Results for $\alpha =0.5$ and $\alpha =0.8$	42
Figure 12. Hierarchical Clustering Algorithm.....	44
Figure 13. Evolutionary Hierarchical Clustering Algorithm	46
Figure 14. Static Hierarchical Clustering from t=1 to t=4	49
Figure 15. Evolutionary Hierarchical Clustering from t=2 to t=5 for $\alpha=0.8$	51
Figure 16. Static Hierarchical Clustering.....	52
Figure 17. Evolutionary Hierarchical Clustering for $\alpha=0.8$.	53

Figure 18. Accuracy Results of Hierarchical Clustering...	54
Figure 19. Accuracy Results Evolutionary K-Medoid Clustering with different time values for alpha.....	56
Figure 20. Percentage of common nodes in the previous two- time slots.....	57
Figure 21. Accuracy Results Evolutionary K-Medoid Clustering	58
Figure 22. Results of Classification of Mobile users based on the Predictability Level using Visited Destinations.....	62
Figure 23. Results of Classification of Mobile users based on the Predictability Level using survey data.....	63
Figure 24. Examples of cliques.....	66
Figure 25. Bron-Kerbosch Algorithm.....	67
Figure 26. Sample Results of Finding the Maximal Cliques using Bron-Kerbosch Algorithm.....	74
Figure 27. Frequency of Clique Size.....	75
Figure 28. Centroids in K-medoid Evolutionary Clustering.	78
Figure 29. An Example of Four Maximal Cliques in One Time Slot.....	80
Figure 30. Degree Centrality Scores for different time slots	86
Figure 31. Closeness Centrality Scores for different time slots.....	87

Figure 32. Betweenness Centrality Scores for different time slots.....	88
Figure 33. Eigenvector Centrality Scores for different time slots.....	89
Figure 34. PageRanker Centrality Scores for different time slots.....	90
Figure 35. The results applying the evolutionary centrality for Eigenvector Centrality at time slot 46 with $\alpha = 0.5$..	92
Figure 36. Evolutionary Eigenvector Centrality Scores for different time slots with $\alpha = 0.5$	94
Figure 37. Evolutionary PageRank Centrality Scores for different time slots with $\alpha = 0.5$	95

CHAPTER I

INTRODUCTION

Network Science focuses on the study and analysis of different types of networks such as mobile networks, social networks, computer networks, and biological networks. Recent advances in information and communication technologies have shown dramatic improvement and unparalleled development of our networked world [3]. Networking is the main feature of many large complex systems such as urban transportation systems, electric power markets and grids, the Internet, wired and wireless communication, and the Web [3]. The capability of storing large datasets for complex networks and the need to understand these complex networks is an important motivation for researchers in this field.

In the first section of this chapter, we give a background and the research motivation. The objectives of this research are given in the second section. In section three, a plan of research activities is given including the current and the future research activities. Finally, the organization of the dissertation is given in section four.

1.1 Background and Research Motivation

Network science is an emerging, highly interdisciplinary research area that aims to study the foundations of network structure and behavior and to develop theoretical and practical approaches and techniques to increase our understanding of natural and manmade networks [1, 2]. We can say that we have a huge global system where almost all humans are interconnected and all objects of our environment are networked [3, 4].

Finding communities of densely connected nodes is a fundamental but challenging topic in network science [5, 6]. The focus of this dissertation is on the mobile networks where users are connected using cellular networks. We focus on the design and analysis of different network science techniques that can be used to analyze the dynamics of mobile networks. There is a need to understand the behavior of mobile networks and to find communities within mobile users. This can enhance the provided services for mobile users and it can help service providers to introduce new services that meet the need of discovered communities in mobile networks. We use different techniques which include classification, evolutionary clustering, and discovery of maximal cliques algorithms.

1.2 Goals and Objectives of Our Research

The goal of this research is to analyzing mobility data of users in mobile networks using network science techniques to understand the dynamics behavior of the mobile networks and to find communities within mobile users and to discovery communities of mobile users. Network science techniques include classification, evolutionary clustering, and discovery of maximal cliques algorithms are designed and implemented to analyze mobile networks. The main objective of our research is to:

- Analyze the predictability of mobile users
- Discover similarities between mobile users
- Understand the behavior of mobile users in mobile networks
- Understand the dynamics of mobile networks over time
- Discover communities of mobile users
- Find centrality in mobile networks

1.3 Plan of Research Activities

Our research activities aim to design and develop network science techniques understand the dynamics of mobile networks and discover communities in mobile networks. The

current introduced research in this dissertation aims to design and implement classification algorithms, evolutionary clustering algorithms, discovery of maximal cliques algorithms, and centrality algorithms that can be used effectively to analyze mobile networks.

The research activities are summarized in the following points:

1. Understand the basic concepts of network science
2. Prepare a survey study about network science algorithms, evolutionary clustering, and discovering of communities.
3. Introduce the proposed evolutionary clustering algorithms
4. Introduce the proposed classification algorithms
5. Introduce the proposed discovery of maximal cliques algorithms
6. Discuss the MIT mobile networks dataset that was used to test the proposed techniques.
7. Design and Implement the proposed technique.
8. Design and perform experiments to test the proposed algorithms.
9. Design and perform experiments to compare the performance of the proposed technique with the other techniques.

10. Enhancing the current technique and compare it with other technique.
11. Use R for network visualization and graph representation of network clusters and communities.
12. Study other algorithms that can be used to discover communities in mobile networks.
13. Introduce the enhanced techniques to solve the problem of discovering communities in mobile networks.
14. Implement the proposed solution.
15. Design and perform experiments to test the enhanced technique to discover communities in mobile networks and compare its performance with other techniques.
16. Apply different centrality algorithms to find centrality in mobile networks.

The future research activities include enhancing the proposed techniques, design and perform experiments to test the enhanced techniques to discover communities in mobile networks, and to compare its performance with other techniques.

1.4 Organization of the Dissertation:

The remainder of this dissertation is organized as follows. Chapter 2 gives an overview of network science

algorithms, evolutionary clustering, and discovery of maximal cliques algorithms that have been proposed in the literature. A description of the MIT mobility dataset that was used in this research is also given at the end of chapter 2. Chapter 3 introduces the proposed evolutionary clustering algorithms and its application in mobile networks. Chapter 4 discusses the classification algorithms that are used in this research to study and analyze the behavior and the dynamics of for mobile networks. Chapter 5 discusses the maximal cliques algorithms and the centrality algorithms that were used to analyze the network behavior and the dynamics of mobile networks. Finally, conclusions and the plan of the future work are given in Chapter 6.

CHAPTER II

BACKGROUND AND RELATED WORK

2.1 Network Science

Network Science is a field in computer science that focuses on the study of the theoretical foundations of network structure, network dynamics, network behavior, and the application of network science to different types of networks [1]. Network science techniques can be applied to different types of networks such as mobile networks, social networks, biological networks, power grids and other types of networks. We can represent any network in terms of nodes, links or edges between the nodes, and algorithms that can be applied to the nodes. Network nodes represent the main entities in the network that interact with each other. An edge or a link represents an interaction or a relationship between two nodes. We can define any network as shown below [1]:

$$G(t) = \{ N(t); L(t); f(t) : J(t) \} \quad (1)$$

Where:

$N(t)$: is nodes or vertices at time t

$L(t)$: is links or edges at time t

$f(t)$: $N \times N$ mapping function that connects node pairs

$J(t)$: Algorithm for describing behaviors of nodes and links versus time.

This above definition gives an important representation for the topology of the networks which can be changed dynamically with the time and it connects the different entities in the networks. Algorithms can be used to understand the behavior and the dynamics of the networks. In this dissertation, we will focus on mobile nodes in mobile networks where the network nodes are communicating and the topology is changing with time because of the nodes are not stationary and the communication links are dynamically changing between nodes. An example of this network of mobile users is shown in Figure 1. This network of mobile users is a dynamic network where nodes and links (edges between the nodes) can be added and removed dynamically over time and the topology of the network can change. Network nodes are the main entities in the network that interact with each other. A link (edge) represents an interaction or a relationship between two nodes. These interactions can change over time in dynamic networks and the change in nodes and links represent this dynamic interaction over time.

Figure 1. Example of network representation

This dissertation introduces evolutionary clustering algorithms that can be applied to dynamic networks to identify different clusters within this dynamic network over time. The problem in this case can be defined as follows:

Given a network $G(t_i)$ at time slot i , it is required to apply evolutionary clustering algorithms to find a set of clusters,

$C(t_i)$, at that time slot. We define the dynamic network, $G(t_i)$, as a set of nodes, $N(t_i)$, set of links, $L(t_i)$, with specific mapping function, $S(t_i)$, that connects pair nodes together at time slot i . Another element that can be added to analyze the dynamic network is the algorithm $J(t_i)$ that will be applied on this network at the time slot i to describe the network and to understand the relationship between the nodes. In our problem the nodes represent mobile users, the link between any two nodes represents a friendship between these two users, and the similarity matrix is the mapping function and it represents the degree of friendship between any two users at the given time slot. $J(t_i)$ will be the evolutionary clustering algorithm that will be applied to analyze the network. This can be written as:

$$G(t_i) = \{ N(t_i); L(t_i); S(t_i) : J(t_i) \} \quad (2)$$

Where:

$N(t_i)$: is the set of nodes at time slot i , where every node represents a mobile user.

$L(t_i)$: is the set links between the nodes at time slot i , where every link represents a friendship between two mobile users.

$S(t_i)$: is an $M \times M$ similarity matrix that represents the degree of friendship between pairs of users at time slot i . The degree of friendship between any pair of users is represented by the number of intervals where they were in physical proximity at time slot t_i .

$J(t_i)$: The evolutionary clustering algorithm that will be applied at time slot i to describe behaviors of users within the network by obtaining clusters that evolve smoothly overtime.

M : is the number of nodes (users) in the network.

t_i : is the time slot I and every time slot represents a week.

We will focus in this dissertation on and classification and clustering algorithms for network science and on using mobile networks to study the dynamics of the networks. New evolutionary clustering techniques will be proposed to study the dynamics of the networks. We will use the R programming language to generate several network graphs that can represent the dynamics of networks. Also, one of the cliques algorithms will be applied to discover cliques. In the next sections, we give a review of previous work related classification and clustering.

If the number of clusters is K , then the problem of evolutionary clustering can be described as follows based on the above definitions:

Given a dynamic network $G(t_i)$ at time slot i (week i), it is required to find the set of clusters $C(t_i) = \{C_1, C_2, \dots, C_K\}$ by applying the evolutionary clustering algorithm $J(t_i)$ on the network represented by the similarity matrix $S(t_i)$ at the current time slot and by utilizing the information from the similarity matrix at the previous time slot $S(t_{i-1})$. The goal is to generate clusters that evolve smoothly overtime by producing clustering that is similar to the clustering at the previous time step but reflects the data arrived at the current time step.

2.2 Classification Algorithms

We analyzed the mobile networks data using different classification techniques. The goal is to learn from the recorded data and be able to classify users based on the predictability. A classifier classifies a data item into one of several predefined classes, and typically known as supervised learning. The classes are typically non-overlapping and partition the input database. We discuss in this section the classification algorithms that were used to

analyze the mobile networks. The classification process is divided into two steps:

1. Creating a model using a training data set for which the class labels have already been specified, this done by applying several techniques and comparing their performance to select the one that has the best results. These techniques include Naïve Bayes, and other techniques which will be presented in the next section [7-9].
2. Using the created model to classify new data (test data) [7-9].

There are many problems associated with classification such as missing data. Usually missing data are ignored, or some prediction techniques are used to predict the missing values. Another issue is how to make a judgment about the models or how to decide which model is the best. This usually depends on the user and what they really expect from the data classification. The performance of a classifier can also be evaluated by using information retrieval metrics. Four metrics describe the output of the classifiers (in case of two classes: positive and negative) and can be used to evaluate the performance, these values are:

- TP (true positive), means the group of data classified in a class and it actually belongs to it in other words TP means number of positive samples that are classified as positive.
- FP (false positive), means the group of data classified in a class and it actually doesn't belong to it in other words it means number of negative samples that are classified as positive.
- TN (true negative), means the group of data that not classified in a class and it actually doesn't belong to it in other words number of negative samples that are classified as negative.
- FN (false negative), means the group of data that not classified in a class and it actually doesn't belong to it in other words case number of samples that are positive and classified as negative.

Table 1 shows a 2-class confusion matrix that contains the four values, TP, FP, TN, and FN [7-9].

Table 1. Confusion matrix

	Predicted positive	Predicted negative
Actual positive	TP	FN
Actual negative	FP	TN

From the above four metrics, we can interpret the results using some ratios that are called accuracy, precision and recall. These ratios will be explained as follows [7, 8]:

- **Accuracy:** the percentage of predictions that are correct

$$Accuracy = (TP+TN) / (TP+FN+FP+TN)$$

- **Precision:** The percentage of correctly classified positive cases relative to the cases classified as positive

$$Precision = (TP) / (TP+FP)$$

- **Recall:** The percentage of positive cases that were successfully classified as positive

$$Recall = (TP) / (TP+FN)$$

Another graphical interpretation of the results is the Receiver Operating Characteristic curve (ROC curve), which shows the relationship between false positives and true

positives. Another curve shows the relationship between precision and recall, and is the one used predominantly in this work.

Below, will give an overview of the classifiers used in the present work. These classifiers include sequential minimal optimization (SMO), logistic regression, Naïve Bayes, and Bayesian network classifiers.

- **Naïve Bayes:** is a classification technique based on Bayes theorem [9] which is:

$$P(H|X) = \frac{P(X|H)P(H)}{P(X)} \quad (3)$$

Where X is the data tuple and is considered the evidence and H is a hypothesis that X belongs to a class C . To solve a classification problem we want to determine $P(H|X)$, the probability that the hypothesis H holds given the data tuple X or the probability that X belongs to class C if we know X . $P(H|X)$ is called the posterior probability, while $P(H)$ is known as the prior probability since it is independent of X . Finally $P(X|H)$ is the probability of X conditioned on H , and $P(X)$ is the prior probability of X . To use Naïve Bayes, assume that we have a training data D and each tuple of data is

represented by a vector $X = (x_1, x_2, \dots, x_n)$, and suppose that we have m classes $(C_1, C_2, \dots, C_m)$, then we have to find the maximum posterior probability $P(C_i|X)$

using Bayes theorem [6], given by $P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$.

Since $P(X)$ is constant for all classes C_i , then we need only maximize $P(X|C_i)P(C_i)$. A simple assumption (known as "naïve" assumption) is that all the attributes are independent [9], meaning that there is no relation between them. Thus, we obtain:

$$P(\mathbf{X}|C_i) = \prod_{k=1}^n P(x_k | C_i) = P(x_1 | C_i) \times P(x_2 | C_i) \times \dots \times P(x_n | C_i)$$

Finally, the class label of X can be predicted as C_i if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \forall i \neq j \quad (4)$$

2.3 Evolutionary Clustering

Clustering is the process of grouping similar objects in one group called cluster. The problem of clustering become more complex when the objects to be clustered evolve overtime and a clustering result is desired at each time step [10]. This situation is found in many network science applications and there is a need to introduce new clustering methodologies

that can reflect long-term trends while being robust to short-term variations [10].

Evolutionary clustering is one of the new clustering methodologies that attracted many researchers recently. Chakrabarti et al. [11] introduced the problem of evolutionary clustering in 2006 in. A generic framework to solve the problem of evolutionary clustering was introduced in [11]. The authors in [11] discussed evolutionary versions of two widely-used clustering algorithms within this framework: k-means and agglomerative hierarchical clustering. In [11], the concepts of the cluster snapshot quality and the cluster history cost were introduced. The snapshot quality of cluster C^t (a cluster at time t) measures how well C^t represents the data at time step t . The history cost of the clustering is a measure of the distance between C^t and C^{t-1} , the clustering used during the previous time step [11]. Typically, the snapshot quality is defined in terms of the data points themselves, while the history cost is a function of the cluster models [11]. The overall cost of the clustering sequence is a combination of the snapshot quality and the history cost at each time step [11] and the total quality of a cluster sequence can be calculated based on the snapshot quality and the history cost functions. The general framework for evolutionary clustering was introduced in [11] by adding

a temporal smoothness penalty to a static clustering method. They show in [11] that a stable and consistent clustering is obtained even when the data exhibits noisy behavior, and a smooth sequence of clusters with very low history cost can be obtained for only a small reduction in total snapshot quality.

Another approach for evolutionary clustering was introduced in [12, 14] which can be considered as an expansion for the proposed evolutionary clustering in [11]. The authors in [12, 14] introduced two frameworks to add temporal smoothness in evolutionary spectral clustering. The proposed frameworks are based on a general function contains two costs to measure the quality of a clustering result on evolving data points [12, 14]. The first cost is the snapshot cost which measures the quality of the current clustering at time t with respect to the current data features at time t [12, 14]. The second cost is the temporal cost which measures the quality of the current clustering at time t with respect to historic data features or historic clustering results at time $t-1$ [12, 14]. The proposed two frameworks are called PCQ (preserving cluster quality) and PCM (preserving cluster membership) frameworks [12, 14]. The main difference is in the definition of the temporal cost [12, 14]. In the PCQ framework, the current clustering is applied to historic data and the resulting cluster quality determines the temporal

cost [12, 14]. In the PCM framework, the temporal cost is based on the difference between the current clustering and the historic clustering [12, 14].

The authors in [10] introduced a different methodology for evolutionary clustering. They consider it as a problem of tracking and then they used static clustering [10]. At each time step t , they observe the proximity matrix (W^t) between the different objects. The proximity matrices will be observed at each time step and they are based on the features vector. The proximity matrix can be a similarity or a dissimilarity matrix according to the nature of the problem to be solved. They model the proximity matrix as a linear combination of a true proximity matrix (ψ^t) and a zero-mean noise matrix (N^t) [10]. Therefore we can model the proximity matrix as shown below [10]:

$$W^t = \psi^t + N^t \quad (5)$$

The goal is to estimate the true proximity matrix, ψ^t , at each time step [10] and then perform a static clustering on ψ^t [10]. If W^t is used as an approximation to ψ^t , then the obtained clustering can be unstable and inconsistent with clustering results from previous time steps [10]. A better approximation is to estimate ψ^t as [10]:

$$\psi^t = \alpha^t \psi^{t-1} + (1 - \alpha^t) W^t \quad (6)$$

This is called the smoothed proximity matrix where $t \geq 1$ and $\psi^0 = W^0$. The parameter α^t is called the forgetting factor and it controls the rate at which the past proximities are forgotten [10]. The authors in [10] discussed how to estimate the optimal forgetting factor. Based on that they introduced a framework called the Adaptive Forgetting Factor for Evolutionary Clustering and Tracking (AFPECT). The main advantage of the proposed framework is its flexibility to work with the dynamic changes in the number of clusters and the objects to be clustered.

There is a close relationship between evolutionary clustering and clustering data streams. Clustering data streams was introduced before evolutionary clustering and many researchers worked clustering data streams [22, 23, 24, 25].

Both of evolutionary clustering and clustering data streams work on data that evolve over time. In clustering data streams, we have large ordered sequence of points that must be accessed in order and arrive at high rate. Each reading of the sequence is called a pass or a scan [24]. We can say that data streams introduce a one-pass constraint on

the design of the clustering algorithms [22]. It is not practical in clustering data stream to scan the data multiple times or to store all the data in memory [12]. The focus of research in clustering data stream is on how to cluster large datasets using limited memory resources and using one-pass scanning of data [22,24,12,25]. Clustering algorithms in this case need to find a way to maintain a substantial amount of information without losing the essential details [22].

In evolutionary clustering, the focus is not on scalability or the one-pass access issue, but it focuses on obtaining clusters that evolve smoothly overtime [22]. In evolutionary clustering, the data is clustered at each time step to produce a clustering that is similar to the clustering at the previous time step but reflects the data arrived at the current time step [11]. The first framework to solve the problem of evolutionary clustering was introduced in [11] as we discussed earlier. The concepts of the cluster snapshot quality and the cluster history cost were introduced. The snapshot quality of cluster C_t (a cluster at time t) measures how well C_t represents the data at time step t . The history cost of the clustering is a measure of the distance between C^t and C^{t-1} , the clustering used during the previous time step [11]. In evolutionary clustering, we need to have a high quality clustering at any time step without

changing the clustering dramatically from one time step to the next [11,22].

2.4 MIT Reality Mining Dataset

MIT reality mining dataset [19] was used in our experiments in the next chapters to test the proposed evolutionary techniques in this dissertation. This dataset has records for 94 users and it was recorded for a period of 9 months. The dataset was available as MATLAB data. This data has a variety of information including mobility data survey data collected for mobile users. MATLAB toolboxes and AFFECT [10] toolbox was used to help in implementing the evolutionary clusters.

We considered the mobility features in the datasets to find friendship among users. These data include MAC address data, location data, and time stamps. These data are recorded for all mobile users. Our analysis in next chapters is based on the fact that the MAC address can be discovered by a periodic Bluetooth scan performed by another phone and this is indicative of the fact that the two users' phones are within 5-10 meters of each other. A similarity matrix was used to record the similarity between users which represents the number of intervals where they were in physical proximity

for different weeks. The evolutionary clustering algorithm, maximal cliques algorithm, and classification algorithms in the next chapter were tested using this dataset. Appendix A [19] gives more details related to the MIT Reality Mining dataset.

CHAPTER III

PROPOSED EVOLUTIONARY CLUSTERING ALGORITHMS

3.1 Evolutionary K-Medoids Clustering

K-medoids clustering is similar to k-means clustering but the centroid of any cluster should be one of the clustered points. This is very useful in the mobile environment where these points will help in routing functions and other mobility applications in ad-hoc networks. The distance between points will be used to construct the similarity matrix.

K-medoids clustering is more robust to noise and more resistant to outliers and it does not need to calculate the Euclidean distances for every iteration like the K-means algorithm [17].

Partitioning around medoids (PAM) is one of the well-known techniques that can be used for K-medoids clustering [16]. We developed an evolutionary version of the K-medoid clustering where the similarity measures at any week is based on the similarity matrix of the current week and the previous week. We used a variable, α , to determine the percentage we consider from both weeks (α and $(1 - \alpha)$). A temporal smoothing

was used after the first time step. At each time step after the first time step, we identify the new objects in current similarity matrix that has been introduced. These objects will need to be included in the clusters. No temporal smoothing is performed for the new objects because they are recently introduced. Figure 2 shows the basic steps of the K-medoids algorithm.

```
KMedoids
{
Inputs:
     $W_t[n \times n]$ : similarity matrix for the n points to be clustered at time t
    k: number of clusters

Current Centroids= select k points as centroids
    For every non-centroid point:
        Find the distance between the points and the centroids
    Calculate the overall error for Current Centroids

Continue until no other possible combinations:
    New Centroids = select a new combination of k points as centroids
    For every non-centroid point:
        Find the distance between the points and the centroids
    Calculate the overall error for New Centroids

    If overall error for New Centroids < overall error for Current Centroids:
        Current Centroids = New Centroids
}
```

Figure 2. K-medoids Clustering Algorithm

The K-medoids algorithm starts by randomly selecting k points of the data points as medoids. These medoids will

represent the centers of the initial clusters. The data points will be associated with the closest medoids. Then the trials will continue to find the best association of these points based on the similarity measure. The process will include several iterations to find the best clustering.

Figure 3 shows the basic steps of the evolutionary K-medoids algorithm. It starts by randomly selecting k points of the data points as medoids. The clustering at the initial time will be based only on the data of the similarity matrix at the initial time ($t=1$) since there is no previous information related to the similarity from the previous time. Therefore a standard version of K-medoids algorithm will be used at the initial time ($t=1$). The evolutionary K-medoids algorithm will be applied at the next time step (at $t=2$). A temporal smoothing will be used in this case to utilize the clustering information that was generated from the previous time step.

The evolutionary algorithm starts by identifying the rows and the columns of new objects in the current similarity matrix. Then a smoothed similarity matrix is built based on the similarity matrix of the current time step and the similarity matrix of previous time step. The smoothing will not be applied to the new objects. The smoothing will be built on a smoothing factor α :

$$w'_t = (1 - \alpha)w_t + \alpha w_{t-1} \quad (7)$$

Where w_t is the similarity matrix at time t and w'_t is the updated similarity matrix after temporal smoothing.

```

EvolutionaryKMedoids
{
  Inputs:
     $W_t[n \times n]$ : similarity matrix for the  $n$  points to be clustered at time  $t$ 
     $k$ : number of clusters

    if  $t=1$ :
      call KMedoids
    else:
      find new objects at time  $t$ 
      for each non-new object
        Smooth the similarity matrix using parameter  $\alpha$ 
      call KMedoids

  KMedoids
  {
    Current Centroids= select  $k$  points as centroids
    For every non-centroid point:
      Find the distance between the points and the centroids
      Calculate the overall error for Current Centroids

    Continue until no other possible combinations:
      New Centroids = select a new combination of  $k$  points as centroids
      For every non-centroid point:
        Find the distance between the points and the centroids
        Calculate the overall error for New Centroids

      If overall error for New Centroids < overall error for Current Centroids:
        Current Centroids = New Centroids
  }
}

```

Figure 3. Evolutionary K-medoids Clustering Algorithm

The data points will be associated with the closest medoids. Then the trials will continue to find the best association of these points based on the similarity measure. The process will include several iterations to find the best clustering.

3.2 Results of Evolutionary K-Medoids Clustering

In this section we start by introducing the results without considering the evolutionary approach and compare it with the results of the introduced evolutionary approach for different cases.

Figure 4 shows the results for time slots from 1 to 4 for $k=2$ and using a static K-medoids clustering. Figure 5 shows the results time slots from 1 to 4 for $k=2$ and $\alpha=0.8$ using evolutionary K-medoids clustering. The thickness of the edges in these graphs represents the similarity between the points (nodes). For example, Figure 4 shows that X2 is more similar to X9 than X22 at time slot 4.

Figure 4. K-medoids results for K=2

Figure 5. Evolutionary K-medoids results for $K=2$ and $\alpha=0.8$

The results show that the evolutionary clustering was able to identify a number of outlier nodes and it was able to keep the same correct clustering for many points by taking information from previous time slot. For example, at time slot 4 nodes 16 and 92 were not moved to a new cluster and they were assigned to the green cluster with a medoid at X2 because we did not ignore the similarity at time slot 3. Also, X15 was identified to be an outlier because there because we considered also similarity from previous time slots.

3.3 Evolutionary DBSCAN Clustering

DBSCAN clustering is based on the density of points. In this case every group of users will be grouped automatically in one cluster if they are close (density) to each other based on a minimum number of close points and a maximum distance.

DBSCAN clustering is based on two important parameters. The first parameter is the maximum distance between two points to be considered neighbor (ϵ). The second parameter is the minimum number of neighbor points required to form a cluster. In our case we use pairwise distances between the points to be clustered. The process will need a number of iterations to find the clusters based on the maximum distance and the minimum number of points.

The DBSCAN algorithm is shown in Figure 6(a). The the distances between all point at every time slot is calculated based on the given similarity matrix where:

$$D_t[n \times n] = 1 / W_t[n \times n] \quad (8)$$

The algorithm starts by visiting every point that was not visited in previous iterations at the time t and try to find the list of neighbor points based on the maximum distance and the minimum number of points. This point will be marked as visited and we find all neighbor points to this point.

```

DBSCAN
{
Inputs:
     $D_t[n \times n]$ : distance matrix for the  $n$  points to be clustered at time  $t$ 
     $\epsilon$  : maximum distance between two points to be considered neighbor
    MinPoints: minimum number of neighbor points required to form a cluster

Cluster_num=0 // initialize the number of clusters
For every point:
{
    If the point  $i$  is not visited:
        Mark  $i$  as visited
         $iNeighbors = \text{FindNeighbors}(D_t, i, \epsilon)$ 
         $Num\_iNeighbors = \text{size}(iNeighbors)$ 
        if  $Num\_iNeighbors < MinPoints$ : // (this is a noise point)
            Mark  $i$  as a noise point
        else: // this is a core point
            Cluster_num = Cluster_num + 1; // we have a new cluster
            Expand_Cluster( $i, iNeighbors, Cluster\_num, MinPoints, visited, \epsilon$ )
    }
}

```

Figure 6(a). DBSCAN Clustering Algorithm

```

Expand_Cluster
{
Inputs:
    i: a core point
    iNeighbors: list of neighbor points
    Cluster_num: cluster number
    MinPoints:    minimum number of neighbor points required to form a cluster
    visited: list of visited points

Add point i to the current cluster Cluster_num
For every point i' in iNeighbors:
    {
    If the point i' is not visited:
        Mark i' as visited
        iNeighbors' = FindNeighbors( $D_t$ , i',  $\epsilon$ )
        Num_iNeighbors=size(iNeighbors)
        if Num_iNeighbors >=MinPoints:    //this is a core point
            Add iNeighbors' to iNeighbors

    If point i' is not clustered :
        Add point i to the current cluster Cluster_num
    }
}

```

Figure 6(b) . DBSCAN Clustering Algorithm

If the number of neighbor points is greater than the minimum threshold then this point will be considered as a core point and it will form a new cluster. Otherwise, it will be considered as a noise point and there is no need to continue to expand the cluster based on this point.

If this is a core point, then we will need to expand the new created cluster based on this point and we call

another function to expand the new cluster. This function is shown in Figure 6(b). This function will add the current point i to the new cluster and it will expand our search to the neighbor points of any neighbor points starting from the current point. The search will continue to add new neighbor points until we could not expand it any more. Any expanded neighbor point to any neighbor point will be added to the cluster until we could not add any more neighbors. Each visited cell will be marked as visited to avoid any redundant iteration.

One of the main advantages of this algorithm is its capability to determine the number of clusters during the clustering process. It will form a new cluster at every time we visit a new point that was not visited before if this point has enough number of neighbors. It is important to do several iterations to find the best thresholds for the minimum number of points and the maximum distance.

Based on this, we developed an evolutionary version of the DBSCAN clustering where the maximum distance will be changed in a dynamic way to reflect the change in the new cluster and it will reflect the distance from the previous time slot as well. The basic algorithm is shown in Figure 7. A variable was used to determine the percentage we consider from distances in the previous time slot (previous week in

this case) and $(1 - \alpha)$ represents the percentage we consider from distances in the current time slot.

```

EvolutionaryDBSCAN
{
  Inputs:
    Dt[n×n]:      distance matrix for the n points to be clustered at time t
    MinPoints:    minimum number of neighbor points required to form a
                  cluster

    if t=1:
       $\varepsilon_t = \text{median}(\text{unique}(D_t))/2$ 

    else:
       $\varepsilon_t = (1 - \alpha) \text{median}(\text{unique}(D_t))/2 + \alpha \text{median}(\text{unique}(D_{t-1}))/2$ 

    call DBSCAN
}

DBSCAN
{
  Cluster_num=0          // initialize the number of clusters
  For every point:
    {
      If the point i is not visited:
        Mark i as visited
        iNeighbors = FindNeighbors(Dt, i,  $\varepsilon$ )
        Num_iNeighbors=size(iNeighbors)
        if Num_iNeighbors < MinPoints:      //(this is a noise point)
          Mark i as a noise point
        else:                               //this is a core point
          Cluster_num = Cluster_num +1; // we have a new cluster
          Expand_Cluster(i, iNeighbors, Cluster_num, MinPoints,
            visited,  $\varepsilon$ )
    }
}

```

Figure 7. Evolutionary DBSCAN Clustering Algorithm

We used the median distance of unique distances at every time slot t to set the maximum distance at this time slot before using the evolutionary approach:

$$\varepsilon_t = \begin{cases} \text{median}(\text{unique}(D_t))/2 & t = 1 \\ (1 - \alpha) \text{median}(\text{unique}(D_t))/2 + \alpha \text{median}(\text{unique}(D_{t-1}))/2 & t > 1 \end{cases} \quad (9)$$

This will enable the evolutionary algorithm to change dynamically for every time slot to reflect the new change in the clustering but without ignoring the information from the clustering at the previous time slot and without smoothing the distance matrix.

3.4 Results of Evolutionary DBSCAN Clustering

In this section we start also by introducing the results for DBSCAN algorithm without considering the evolutionary approach and compare it with the results of the introduced evolutionary approach for different cases. Figure 8 shows the results for time slots from 3 to 6 for DBSCAN clustering for $\text{MinPoints} = 1$. Figure 9 shows the results time slots from 3 to 6 using evolutionary DBSCAN clustering.

Figure 8. Static DBSCAN results

Time Slot # 3

Time Slot # 4

Time Slot # 5

Time Slot # 6

Figure 9. Evolutionary DBSCAN Results

The results show that the evolutionary clustering was able to keep the correct clustering for many points by taking information from previous time slot. For example, at time slot 4 nodes 14 and 19 were not moved to a new cluster and they were assigned to the same white cluster because we did not ignore the similarity at time slot 3. Also, outliers were identified at different time slots. For example, four outliers were discovered at time slot 5 that we were not discovered before using the static clustering. The evolutionary clustering was able to identify these outliers by taking information from current and previous time slots.

3.5 Analysis of Accuracy of K-Medoid results

Figure 10 shows an analysis of the K-Medoid results for different values of α including the static K-Medoid.

Figure 10. Accuracy Results of the K-Medoid

The results for $\alpha=0$ represents the static K-Medoid where no information of previous time slots are considered. The results for the evolutionary K-Medoid at $\alpha=0.5$ and $\alpha=0.8$ are also considered. The results in Figure 10 shows clearly the enhancement in the results by applying the evolutionary clustering at $\alpha=0.5$ and $\alpha=0.8$ compared to the results of static clustering at $\alpha=0$. For example, we can see that the results are significantly enhanced in different cases such as the time slot 13. Also, the overall results are better compared to the static clustering. A further study can be conducted to find the best time windows need to be considered from previous time slots and best value for α for evolutionary clustering. In our case, we considered only the immediate previous time slots and $\alpha=0.5$ and $\alpha=0.8$. The time window value can be also changed to consider the previous two time slots.

Additional experiments were conducted to compare between the proposed evolutionary K-Medoid clustering algorithm and the evolutionary K-Means clustering algorithm. The average accuracy results overall the time steps for $\alpha =0.5$ and $\alpha =0.8$ are shown in Figure 11 for both algorithms.

The results show that the proposed evolutionary K-Medoid clustering algorithm outperforms the evolutionary K-Means algorithm for both cases. The results show also an enhancement

in the results was achieved at $\alpha=0.8$ for the evolutionary K-Medoid clustering compared to $\alpha=0.5$ while we could not achieve this enhancement using the evolutionary K-Means clustering. The results show that the proposed evolutionary K-Medoid clustering is promising. It is more robust to noise and more resistant to outliers and it does not need to calculate the Euclidean distances for every iteration like the K-means algorithm.

Figure 11. Average Accuracy Results for $\alpha = 0.5$ and $\alpha = 0.8$

3.6 Evolutionary Hierarchical Clustering

Hierarchical clustering is another methodology to find clusters among datasets of objects. It groups objects into clusters at different levels and a multilevel clustering is produced. We consider in our research agglomerative hierarchical clustering. We start by finding the distances between every two objects. Based on the distances, we start to group and link objects at different levels. We continue to group objects and we stop when all objects are grouped in one cluster. A dendrogram will be produced as an output to show the resultant clusters at different levels. We need in our application to cluster the mobile users into two clusters and we can stop in this case at two clusters. Figure 12 shows the basic steps of the Hierarchical Clustering algorithm. The algorithm starts by finding the distances between all point at every time slot based on the given similarity matrix where:

$$D_t[n \times n] = 1 / W_t[n \times n] \quad (10)$$

Then it will consider every point at time t as a cluster and then find the inter-cluster distances between the points. The next step will be Sorting the clusters based on the distances and then we start to merge the clusters that are close to each other. We will continue to do this process on the new formed clusters in each iteration until we have all points in one cluster.

HierarchicalClustering {

Inputs:

$W_t[n \times n]$: similarity matrix for the n points to be clustered at time t

Convert the similarity matrix to a distance matrix $D_t[n \times n] = 1 / W_t[n \times n]$

Place each point in its own cluster.

Find the list of inter-cluster distances between all clusters using single linkage distance

Sort this list of clusters in ascending order based on the distance.

Continue until all data are in one cluster:

New list of clusters = merge pairs of clusters in the current list that are close to each other based on the distance threshold

Figure 12. Hierarchical Clustering Algorithm

Figure 13 shows the basic steps of the evolutionary hierarchical clustering. The algorithm starts by finding the distances between all point at every time slot based on the given similarity matrix as before. The clustering at the initial time will be based only on the data of the distance matrix at the initial time ($t=1$) since there is no previous information related to the distance from the previous time. Therefore, a standard version of hierarchical algorithm will be used at the initial time ($t=1$). The evolutionary hierarchical algorithm will be applied at the next time step (at $t=2$). A temporal smoothing will be used in this case to

utilize the clustering information that was generated from the previous time step.

It starts by identifying the rows and the columns of new objects in the current similarity matrix. Then a smoothed distance matrix is built based on the distance matrix of the current time step and the distance matrix of previous time step. The smoothing will not be applied to the new objects. The smoothing will be built on a smoothing factor α :

$$w'_t = (1 - \alpha)w_t + \alpha w_{t-1} \quad (11)$$

Where w_t is the similarity matrix at time t and w'_t is the updated similarity matrix after temporal smoothing.

Then the algorithm will continue to merge the closest clusters based on the distances. The process will include several iterations to find the best clustering.

```

EvolutionaryHierarchicalClustering
{
Inputs:
     $W_t[n \times n]$ : similarity matrix for the n points to be clustered at time t

    Convert the similarity matrix to a distance matrix  $D_t[n \times n] = 1 / W_t[n \times n]$ 

    if t=1:
        call HierarchicalClustering
    else:
        find new objects at time t
        for each non-new object
            Smooth the similarity matrix using parameter alpha
        call HierarchicalClustering

HierarchicalClustering {
Inputs:
     $W_t[n \times n]$ : similarity matrix for the n points to be clustered at time t

    Convert the similarity matrix to a distance matrix  $D_t[n \times n] = 1 / W_t[n \times n]$ 

    Place each point in its own cluster.

    Find the list of inter-cluster distances between all clusters using single
    linkage distance

    Sort this list of clusters in ascending order based on the distance.

    Continue until all data are in one cluster:
        New list of clusters = merge pairs of clusters in the current list that are
        close to each other based on the distance threshold

}

```

Figure 13. Evolutionary Hierarchical Clustering Algorithm

3.7 Results of Evolutionary Hierarchical Clustering

In this section we start by introducing the results without considering the evolutionary approach and compare it with the results of the introduced evolutionary approach for different cases. Figure 14 shows the results for time slots from 1 to 4 for using a static hierarchical clustering. Figure 15 shows the results for time slots from 2 to 5 for $\alpha=0.8$ using evolutionary hierarchical clustering. The results show that the evolutionary clustering was able to keep the same correct clustering for many points by taking information from previous time slot. For example, at time slot 2 clustering was enhanced significantly by taking information from time slot 1. Nodes 13 and 92 were considered in one cluster and the algorithm tried to keep the clusters at different levels and at the same time consider the similarity information at the new time slot.

Figure 16 shows the results for the first two time slots after considering only two clusters and using the static hierarchical clustering. Figure 17 shows the results for the first two time slots after considering only two clusters and using the evolutionary hierarchical clustering with $\alpha=0.8$. This shows clearly that we try to keep the correct two clusters at time slot 2 by taking the similarity information

from the previous time slot and the current time slot as well which significantly improved the clustering process.

Time Slot #1

Time Slot #2

Time Slot #3

Time Slot #4

Figure 14. Static Hierarchical Clustering from $t=1$ to $t=4$

Time Slot #2

Time Slot #3

Time slot #4

Time Slot #5

Figure 15. Evolutionary Hierarchical Clustering from $t=2$ to $t=5$ for $\alpha=0.8$

Time Slot # 1

Time Slot # 2

Figure 16. Static Hierarchical Clustering

Figure 17. Evolutionary Hierarchical Clustering for $\alpha=0.8$

3.3 Analysis of Accuracy of Hierarchical Clustering

Figure 18 shows an analysis of the accuracy for different values of α including $\alpha=0$ at different time slots and by considering two clusters.

Figure 18. Accuracy Results of Hierarchical Clustering

The results for $\alpha=0$ represents the static clustering where no information of previous time slots are considered. The results for the evolutionary clustering at $\alpha=0.5$ and $\alpha=0.8$ are also considered. The results in Figure 18 show clearly the enhancement in the results by applying the evolutionary

clustering at $\alpha=0.8$ compared to the results of static clustering at $\alpha=0$. For example, we can see that the results are significantly enhanced at time slot 2 and time slot 21. A further study can be conducted to find the best time windows need to be considered from previous time slots and best value for α for evolutionary clustering. In our case, we considered only the immediate previous time slots and $\alpha=0.5$ and $\alpha=0.8$. The time window value can be also changed to consider the previous two time slots.

3.8 Working with Different Time Windows

We have considered only a time window of width 1 where we considered only the immediate previous time slot. We will study in this section the effect of using time window of width two. In this case, we consider the previous two time slots. We start by identifying the rows and the columns of new objects in the current similarity matrix. Then a smoothed distance matrix is built based on the distance matrix of the current time step, the distance matrix of previous time step at $t-1$, and the distance matrix of previous two time steps at $t-2$. The smoothing will not be applied to the new objects. The smoothing will be built on smoothing factors α_1 and α_2 :

$$w'_t = (1 - \alpha_1 - \alpha_2)w_t + \alpha_1 w_{t-1} + \alpha_2 w_{t-2} \quad (12)$$

$$\text{and} \quad \alpha_1 + \alpha_2 = \alpha \quad (13)$$

Where w_t is the similarity matrix at time t and w'_t is the updated similarity matrix after temporal smoothing. The accuracy results for evolutionary K-Medoid clustering is shown in Figure 19 for different time slots and different values of alpha.

Figure 19. Accuracy Results Evolutionary K-Medoid Clustering with different time values for alpha

The results show that we can still obtain a good clustering results at different time slots and the results are enhanced at different time slots such as 8,12, 13, and 15.

A further study was performed to see the correlation between the number of common nodes in the previous time slots and the obtained accuracy for evolutionary clustering at current time slots. Figure 20 shows the percentage of the common nodes at previous two time slots for different time slots and Figure 21 shows the accuracy results for evolutionary K-Medoid clustering with $\alpha_1=0.6$ and $\alpha_2=0.2$.

Figure 20. Percentage of common nodes in the previous two-time slots

Figure 21. Accuracy Results Evolutionary K-Medoid Clustering

The percentage of common nodes at previous time slots are considered 1 for the first two time slots. After time slot 2, the percentage of common nodes (P_c) at the previous two time slots is calculated by finding the number of common nodes at t , $t-1$ and $t-2$ and calculate the percentage with respect to time t and it is given by:

$$P_c = \frac{|nodes(t) \cap nodes(t-1) \cap nodes(t-2)|}{|nodes(t)|} \quad (14)$$

Where $nodes(t)$: is the set of all nodes at time t .

The results in Figures 20 and 21 show clearly that there is a strong correlation between the percentage of the common nodes in previous time slots and the accuracy of clustering. The accuracy will improve if the percentage of the common nodes in previous time slots is high. This is shown for most of the time slots and it is clearly shown for time slot 8 as an example where all nodes are common and the accuracy is the highest.

CHAPTER IV

CLASSIFICATION ALGORITHMS IN MOBILE NETWORKS

4.1 Classification Algorithms for Mobile Networks

In this section, we discuss the analysis of the MIT datasets to have a better understanding of the dynamics of mobility. The data survey included dyadic questions regarding the average reported proximity and friendship with the other users, as well as questions concerning the individual's general satisfaction with his or her work group. This will be useful in our network analysis.

We started by performing the required preprocessing for the data. We will focus in this section on the analysis the data using classification techniques.

The goal of these experiments was to learn from the recorded data and be able to classify users based on the predictability. WEKA tool [18] was used to compare between different classification techniques.

We converted these data from Matlab format to arff format which is needed by WEKA. In the first analysis, we considered the visited destinations at different times of the day to

classify users to one of three levels of predictability (not predictable, somewhat predictable, or very predictable).

We considered the time of visiting home, work, and other places based on the recorded data. Then we applied different classification techniques using different time windows. The results that were obtained for different classification algorithms are shown below.

In the second analysis, we considered the survey data and their relation to the predictability of mobile users. We applied attribute ranking algorithm to extract the important attributes that should be considered for further analysis.

We applied several classification algorithms including sequential minimal optimization (SMO), Naive Bayes, and Bayesian network classifiers.

Table 2 shows a summary of sample data that was used for classification. This sample shows data based on the 24-hour time slots with information related to percentage of the time at home and work locations for every hour time slot. A summary of the results is shown below in Figures 22 and 23.

Table.2 Data used for Classification: Visited destinations at different times of the day

User ID	Hour 1 - home	Hour 2 - home	...	Hour 24 - home	Hour 1 - work	Hour 2 - work	...	Hour 24 - work
1	% of time he was home	% of time he was home		% of time he was at home	% of time he was at work	% of time he was at work	...	% of time he was at work
2	% of time he was home	% of time he was home		% of time he was at home	% of time he was at work	% of time he was at work	...	% of time he was at work
...	...	...	...	...	...	...	...	...
106	% of time he was home	% of time he was home		% of time he was at home	% of time he was at work	% of time he was at work	...	% of time he was at work

Figure 22. Results of Classification of Mobile users based on the Predictability Level using Visited Destinations

Figure 23. Results of Classification of Mobile users based on the Predictability Level using survey data

These results represent the percentage of correctly classified mobile users based on predictability level. Figure 22 shows the classification results for the three algorithms based on the predictability level using visited destinations while Figure 23 shows the classification results for the three algorithms based on the predictability level using survey data.

Sequential minimal optimization (SMO) classifier shows the best performance for classification of the predictability based on visited distention as shown in Figure 22.

Bayesian network classifier shows the best performance for classification of the predictability based on survey data as shown in Figure 23.

These results show that there is a high level of predictability between mobile users and this information can be helpful for mobile service provider to enhance current services or to introduce new services. Further study can include different time slots and test different algorithms.

CHAPTER V

MAXIMAL CLIQUES AND CENTRALITY IN MOBILE NETWORKS

5.1 Finding Maximal Cliques in Mobile Networks

Finding clusters that present communities in social networks is one of the important problems [26-30]. In this section, we study the applicability of maximal cliques algorithms in the MIT reality mobile social network. The goal is to find the maximal cliques at different time slots.

A clique in social networks represents a group of people or nodes that are connected in a complete graph where every node is connected to each other. An example of clique is shown in Figure 24. A k -clique is a clique of order k if it has k nodes. The clique shown in Figure 24 (a) is an example of a 6-clique while the clique shown in Figure 24(b) is an example of a 13- clique.

A maximal clique is a clique that is not a subset of any other clique. The graph shown in Figure 23 is a maximal clique while any subgraph of this graph is a clique.

Figure 24. Examples of cliques

The well-known Bron-Kerbosch algorithm was used in this problem in a recursive manner to find the maximal cliques. The general algorithm requires an undirected graph with no self-edges which is applicable in mobile social networks where the link between two nodes is undirected and there is no edges from one node to itself.

Suppose that there are three sets R , P , and X where:

R : represents the set of vertices that represent a maximal clique or can be extended to a maximal clique.

P : represents the set of vertices that are connected to all vertices in R and can be added to R to make a larger clique.

X: represents the set of vertices connected to all vertices in R but excluded from being added to R.

$N(v)$: returns a list of vertices that are neighbors to v.

The Bron-Kerbosch algorithm starts by initializing R and X to empty sets and P to the set of the vertices in the graph. The algorithm makes recursive calls to consider every vertex in P in turn. The general algorithm is shown in Figure 25.

This algorithm is one of the most effective algorithms to find clique communities and this is why it was selected to find the maximal cliques in our problem.

```
BronKerbosch(R, P, X):  
    if P = X =  $\varnothing$ :                                // if P and X are empty  
        Report R as a maximal clique  
  
    for each vertex v in P:  
        BronKerbosch( $R \cup \{v\}$ ,  $P \cap N(v)$ ,  $X \cap N(v)$ )  
         $P := P \setminus \{v\}$   
         $X := X \cup \{v\}$ 
```

Figure 25. Bron-Kerbosch Algorithm

5.2 Maximal Cliques Experimental Result

We applied the Bron-Kerbosch algorithm for the MIT reality mining dataset at different time slots to find the maximum cliques. The results are shown below. Figure 26 shows the initial results with different cliques. The size of the

generated clique is based on the time slot of that clique and the number of nodes that interact at this time slot.

(a)

(b)

(c)

(d)

(i)

(j)

(k)

(l)

(m)

(n)

Figure 26. Sample Results of Finding the Maximal Cliques using Bron-Kerbosch Algorithm

The results in Figure 27 show the frequency of the discovered maximal cliques at different times for different clique sizes. A total of 12875 were discovered. The 10-clique is the most frequent maximal clique and the 20-clique is the least frequent as shown in Figure 27.

Figure 27. Frequency of Clique Size

The results in Figures 26 and 27 show the dynamic of the different nodes at different time slots. The cliques at this point were discovered independently at every time slots without considering any information from previous time slot. Although we were able to discover most of the cliques, there still a possibility to enhance the results by considering cliques from previous time slots. This approach will be considered in future work.

5.3 Centrality in Social Networks

Centrality in social networks is a measure of the importance of the node in the network. This is a very useful measure and it has important applications [31-35]. This can be used to discover central nodes that can has important effect on other nodes. One application in healthcare is discovering these central nodes that transmit disease among the nodes.

The proposed evolutionary K-Medoid clustering and the evolutionary DBSCN clustering algorithms can be extended to form dynamic clusters that will identify highly connected individuals(nodes) at the center of social networks. The goal in this case is to identify highly connected individuals (nodes) at the center of the social network based on location

and time information. This approach can have a wide range of applications including early detection of contagious outbreaks [20, 21], broadcasting information, and protecting the network. In the case of contagious outbreaks, these highly connected individuals at the center of the social network are used as sensors for early detection of outbreak. The authors in [20] studied this approach by using nominated friends as social network sensors to monitor the spread of flu at Harvard College [20].

The evolutionary K-Medoid clustering algorithm that is proposed in this dissertation can be extended directly and applied effectively in this case to automatically identify highly connected nodes at the center of social networks based on location and time information. This is because the centroid (medioids) of any cluster in the K-medoid clustering should be one of the clustered points with high connectivity to the other points in same cluster. In this case, the algorithm will be extended to focus on finding the centroids over a time window that could be used to represent the highly connected nodes (individuals) at the center of the social network. In this case, an optimization technique can be applied initially at every time slot to find the optimum number of clusters (or central points in this case) that can represent the nodes. One of the possibility is to use the

genetic algorithm to find the number of centroids. K-medoid evolutionary clustering will be applied after that to find these centroids. These centroids will be considered as candidates to represent the central points over a period of time as shown below in Figure 28.

Figure 28. Centroids in K-medoid Evolutionary Clustering

A time window (consists of a sequence of time slots) will be used in this case to identify centroids that frequently continue to represent the network points during this time window. These centroids are the nodes that can be used as sensors for early detection of outbreak.

The proposed evolutionary DBSCAN clustering algorithm can be extended as well and applied effectively to automatically identify highly connected nodes at the center of social networks based on location and time information. In the DBSCAN clustering, every group of users are grouped automatically in one cluster if they are close(density) to each other based on a minimum number of close points and a maximum distance. The evolutionary DBSCAN clustering algorithm will be extended in this case to find the central node in every generated cluster that represents the highly connected nodes (individuals) within that cluster. These central nodes can be found by finding the centroid of every cluster based on the nodes in this cluster and then finding the closest node to this centroid to represent a central node. A time window can also be used as well to identify the central nodes that frequently continue to represent the points in clusters during this time window. These central nodes are the sensor nodes that can be used for early detection of outbreak.

Another approach is to utilize the maximal cliques algorithm that was used in this dissertation to find the central nodes based on the discovered maximal cliques at every time slot. In this case, the proposed algorithm will be extended to find and generate a list of the common nodes in the discovered maximal cliques at every time step. These nodes

are highly connected to most of the other nodes in this case and they can be considered as central nodes. A time window can be applied in this case as well to find the nodes that frequently continue to be central nodes during this time window. These are the nodes that can be used effectively for early detection of outbreak. Figure 29 shows an example of four discovered maximal cliques at the same time slot. In this example, X2 can be considered as a central point because it is connected to all other nodes in the four cliques and it can be used in this case as a sensor node to detect an outbreak.

Figure 29. An Example of Four Maximal Cliques in One Time Slot

5.4 Centrality Measures for Mobile Networks

There are several techniques that can be used to measure the centrality in social networks. We focus here in this dissertation on five similarity measures that can be used effectively for mobile networks where the network is changing dynamically over time [31-36].

5.4.1 Degree Centrality:

This is a simple measure and it is based on counting the number of neighbor nodes. This degree centrality, C_D , of a node N can be expressed as the degree of the vertex N in the connected graph that represents the network:

$$C_D(N) = \deg(N) \quad (15)$$

Where $\deg(N)$ returns the number of neighbor nodes that are connected to the node N .

The results of the scores for the degree centrality is shown in Figure 30 for different time slots. The results show that central nodes can be detected in the network directly and a threshold can be used to determine the effective central nodes.

5.4.2 Closeness Centrality:

This measures the centrality based on the distance from the given node to all other nodes. The sum of shortest distance to all other nodes is used. This measure is useful to show how fast it will take to reach other nodes. This is very useful in broadcasting and healthcare applications.

$$C_c(N) = \frac{1}{\sum_{v \neq N} d(N, V)} \quad (16)$$

Where $d(N, V)$ is the shortest distance between nodes N and V .

The results of the scores for the closeness centrality is shown in Figure 31 for different time slots. The results show that central nodes that can be used to reach other nodes in a fast way based on the distances.

5.4.3 Betweenness Centrality:

This measures the centrality based on the number of times the node act as a bridge between other two nodes based in the distances between these nodes. The betweenness centrality for a node N can be calculated according to the following steps:

1. Stat with $C_B(N)=0$

2. For each two other nodes (N_1 and N_2) find the shortest path between them:
 - a. If this shortest path passes through N , find p , the percentage of that path that passes through N .
 - b. $C_B(N) = C_B(N) + p$
3. Repeat step 2 for all pairs of nodes.

The results of the scores for the betweenness centrality is shown in Figure 32 for different time slots. The results show that central nodes that can be used effectively to connect other pairs of nodes.

The overall results for the three different measures are consistent and it helps directly to detect the central nodes. This is also consistent with the results from K-Medoid clustering where the medoid nodes are also central nodes. For example, at time t_1 , node 63 and node 2 are the main central points using the three different centrality measures according to Figures 30, 31, and 32.

These results are consistent with the results of the K-Medoid results as shown below. The results at $t=2$, shows that node 63 is a central point for all centrality measures and node 2 for the degree centrality measure and node 93 for the

closeness and betweenness centrality measures. Node 63 and node 2 are consistent of the results with K-Medoid as well. Node 93 was a central node based on the other two measures because it can be used as bridge between many nodes and it connects the two clusters.

5.4.4 Eigenvector Centrality:

This measure is based on the idea of considering a node as an important node if it is connected to other important nodes. it is based on counting the number of neighbor nodes. The vector of centralities, C , can be found based on the adjacency matrix A and a constant λ as follows:

$$\lambda.C = A.C \quad (17)$$

where A is an $n \times n$ adjacency matrix and n is number of all nodes in the network. Any element in A , a_{ij} , represents the connection (edge) between the nodes i and j . Element a_{ij} will be zero if there is no edge between nodes i and j . C is the eigenvector of A with an eigen value of λ .

The results of the scores for the eigenvector centrality is shown in Figure 33 for different time slots. The results show that central nodes that can be used effectively to connect other important nodes.

5.4.5 PageRank Centrality:

This centrality measure algorithm was developed by google to determine the importance of webpages based on the counting the number of links to the page and the quality of these links [36].

The PageRank, $PR(N)$, for any node, N , in this algorithm is based on PageRank, $PR(J)$, for every other node J connected to N and the number of links (edges) of J . This can be expressed in a simple form as:

$$PR(N) = \sum_{\text{for all nodes } J \text{ conncted to } N} \frac{PR(J)}{L(J)} \quad (18)$$

Where:

$PR(N)$ is the PageRank of the node N .

$PR(J)$ is PageRank of the node J that is connected to N .

$L(J)$ is the number of links of the node J .

In this case, the PageRank measure for each node will be initialized to $1/n$ where n is the total number of nodes in the network. The results of the scores for the PageRank centrality is shown in Figure 34 for different time slots. This algorithm is very applicable to dynamic networks because it takes into account the updated links.

Figure 30. Degree Centrality Scores for different time slots

Figure 31. Closeness Centrality Scores for different time slots

Figure 32. Betweenness Centrality Scores for different time slots

Figure 33. Eigenvector Centrality Scores for different time slots

Figure 34. PageRanker Centrality Scores for different time slots

5.5 Evolutionary Centrality

This section introduces an evolutionary approach to enhance the process of finding central nodes. The evolutionary centrality will be calculated based on the connections between the nodes in the network at the current time slot (t) and at the previous time slot ($t-1$). This evolutionary approach considers the dynamics of the network and it takes into consideration the central nodes from previous time slots. The factor α will be used to calculate the updated the adjacency matrix between the nodes (which represents the connections between edges) by considering the connections from current and previous time slot:

$$A'_t = (1 - \alpha)A_t + \alpha A_{t-1} \quad (19)$$

Where A_t is the adjacency matrix at time t and A'_t is the updated adjacency matrix after temporal smoothing.

Figure 35 (c) shows the results of applying the evolutionary centrality using the eigenvector centrality measure for time slot 46 by considering the network connections at time slot 45 and time slot 46. Figure 35 (a) shows the eigenvector centrality at time slot 45 and Figure 35 (b) shows the eigenvector centrality at time slot 46 before applying the evolutionary centrality.

(a)

(b)

(c)

Figure 35. The results applying the evolutionary centrality for Eigenvector Centrality at time slot 46 with $\alpha = 0.5$

The results of the centrality scores in Figure 35(c) shows a significant enhancement in finding the central nodes using the evolutionary centrality. For example, Figures 33 (b and c) shows that the centrality score for nodes 17 was improved significantly compared to other nodes at the current time slot ($t=46$) by considering the network connections at the previous time slot ($t=45$). Figure 35(a) shows the centrality for node 17 and it was the main central point at that time slot. Also, the centrality score for node 44 was improved significantly because it continues to represent a high level of centrality over time as indicated in Figure 35.

Several experiments were conducted to apply the evolutionary centrality for the MIT mobility dataset. The results of evolutionary centrality using the Eigenvector Centrality measure is shown in Figure 36 and the results for different time slots. The results of evolutionary centrality using the PageRank Centrality measure is shown in Figure 37 and the results for different time slots. These results are shown for α factor of 0.5. The results show a significant improvement in finding central nodes by applying the evolutionary centrality approach. The nodes centrality measures were adjusted to reflect the centrality over time for different nodes. This is very important in many applications where the network is changing dynamically.

Figure 36. Evolutionary Eigenvector Centrality Scores for different time slots with $\alpha = 0.5$

Figure 37. Evolutionary PageRank Centrality Scores for different time slots with $\alpha = 0.5$

CHAPTER VI

CONCLUSION AND FUTURE WORK

This work explores the design and implementation of network science algorithms to analyze networks. The goal of this dissertation is to design and develop network science algorithms that can be used to study and analyze mobile networks. This will enable service providers of mobile networks to understand the needs of mobile users and to enhance the quality of the mobile using the essential information and knowledge generated through network science algorithms.

We focus in this dissertation on the design and analysis of different network science techniques that can be used to analyze the dynamics of mobile networks. These techniques include evolutionary clustering, classification, discovery of maximal cliques, and evolutionary centrality algorithms.

The problem of evolutionary clustering was explored in this dissertation. The goal is to introduce new clustering methodologies that are robust to short-term variations but reflects long-term trends. We introduced three evolutionary

clustering algorithms that can be used in mobile networks. These evolutionary algorithms include evolutionary K-medoids clustering, evolutionary DBSCAN clustering, and evolutionary Hierarchical clustering. We conducted several experiments to analyze the performance of the proposed techniques. The experimental results show that these clustering algorithms were able to capture the dynamics of the network where the objects to be clustered evolve overtime. The experimental results show that the proposed algorithms were able to reflect long-term variations and they are robust to short term variations. We conducted several experiments to analyze the performance of using two time slots. The experimental results show that there is a strong correlation between the percentage of the common nodes in previous time slots and the accuracy of clustering.

Different classification algorithms were used to analyze the MIT reality mining mobile dataset to learn from the recorded data and to be able to classify users based on the predictability. These algorithms include sequential minimal optimization (SMO), Naive Bayes, and Bayesian network classifiers. The Bayesian network classifier shows the best performance accuracy compared to other techniques.

The Bron-Kerbosch algorithm was used in a recursive manner to find the maximal cliques in the mobile networks.

Experimental results show that the discovered maximal cliques show the dynamic of the different nodes at different time slots and these maximal cliques can be used as well to find centrality.

We introduced several evolutionary centrality algorithms that can be applied in mobile social networks to analyze the dynamics of the networks and find the central nodes based on the behavior of the network over time. We conducted several experiments using different centrality measures. The proposed evolutionary approach to find centrality in mobile networks shows a significant enhancement in finding the central nodes using the evolutionary Eigenvector Centrality and evolutionary PageRank Centrality. Evolutionary clustering algorithms such as Evolutionary K-medoid clustering can also be used effectively to find central nodes.

The future work includes enhancing the current proposed evolutionary centrality technique by considering more than one time period from previous time slots and applying the proposed evolutionary algorithms on different networking datasets. Future work includes also applying the evolutionary approach to other clustering algorithms and exploring the effect of changing the smoothing factor dynamically based on the percentage of the common nodes at every time slot.

REFERENCES

1. Katy Börner, Soma Sanyal, and Alessandro Vespignani, "Network Science", Annual Review of Information Science & Technology, Volume 41, pp. 537-607, January 2007.
2. Ted Lewis, "Network Science: Theory and Practice", Wiley & Sons, 2008.
3. Ljupco Kocarev and Visarath In, "Network Science: A New Paradigm Shift", IEEE Network: The Magazine of Global Internetworking, v.24 n.6, pp.6-9, November/December 2010.
4. A.-L. Barabasi, "Linked: How Everything Is Connected to Everything Else and What It Means", Plume, 2002.
5. Thang N. Dinh and My T. Thai, "Community Detection in Scale-Free Networks: Approximation Algorithms for Maximizing Modularity", IEEE Journal on Selected Areas in Communications, Vol. 31, No. 6, pp. 997 - 1006, June 2013.
6. M. E. J. Newman, "Detecting community structure in networks", The European Physical Journal B - Condensed

- Matter and Complex Systems, Volume 38, Issue 2, pp 321-330, March 2004.
7. Margret Dunham, Data Mining: Introduction and Advanced Topics, Prentice Hall, 2002.
 8. Mehmed Kantardzic, Data Mining: Concepts, Models, Methods, and Algorithms. Wiley, IEEE Press, 2011.
 9. Jiawei Han, Micheline Kamber, Data Mining: Concepts and Techniques, Morgan Kaufmann, 2011.
 10. KS. Xu, M. Kliger, AO Hero, "Adaptive evolutionary clustering", Journal of Data Mining and Knowledge Discovery, Vol 28, Issue 2, pp 304-336, March 2014.
 11. D. Chakrabarti, R. Kumar, and A. Tomkins, "Evolutionary Clustering", Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 554-560, 2006.
 12. Y. Chi, X. Song, D. Zhou, K. Hino, and B. L. Tseng, "On evolutionary Spectral Clustering", ACM Transactions on Knowledge Discovery from Data, 3(4):17, 2009.
 13. Nathan Eagle, Alex (Sandy) Pentland, "Reality mining: sensing complex social systems", Journal Personal and Ubiquitous Computing, Volume 10 Issue 4, Pages 255 - 268, March 2006.

14. Y. Chi, X. Song, D. Zhou, K. Hino, and B. Tseng, "Evolutionary spectral clustering by incorporating temporal smoothness", KDD Proceedings of 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1-5, Aug 2007.
15. Sergios Theodoridis and Konstantinos Koutroumbas, "Pattern Recognition", Academic Press, 2009.
16. Kaufman, L., and Rousseeuw, P. J., Finding groups in data: An introduction to cluster analysis, New York: Wiley, 1990.
17. Hae-Sang Park and Chi-Hyuck Jun, "A simple and fast algorithm for K-medoids clustering, Expert Systems with Applications", Volume 36, Issue 2, Part 2, pp. 3336-3341, March 2009.
18. Eibe Frank, Mark A. Hall, and Ian H. Witten, The WEKA Workbench, Online Appendix for "Data Mining: Practical Machine Learning Tools and Techniques", Morgan Kaufmann, Fourth Edition, 2016.
19. Nathan Eagle, Alex (Sandy) Pentland, "Reality mining: sensing complex social systems", Journal Personal and Ubiquitous Computing Volume 10 Issue 4, pp. 255 - 268, March 2006.

20. Nicholas A. Christakis, James H. Fowler, "Social Network Sensors for Early Detection of Contagious Outbreaks", PLoS One, Vol. 5(9), pp. 1-8, 2010.
21. Manuel Garcia-Herranz, Esteban Moro, Manuel Cebrian, Nicholas A. Christakis, and James H. Fowler, "Using Friends as Sensors to Detect Global-Scale Contagious Outbreaks", PLoS One, Vol. 9(4), pp. 1-7, 2014.
22. C. AGGARWAL, J. HAN, J. WANG, and P. YU, "A framework for clustering evolving data streams", Proceedings of the 29th international conference on Very large data bases, Volume 29, pp. 81-92, 2003.
23. L. O'Callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-Data Algorithms for High-Quality Clustering", Proceedings of the 18th International Conference on Data Engineering (ICDE '02), pp. 685-694, 2002.
24. S. Guha, A. Meyerson, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering Data Streams: Theory and Practice", IEEE Transaction on Knowledge and Data Engineering, Vol. 15 (3), pp. 515-528, 2003.
25. S. Guha, N. Mishra, R. Motwani, and L. O'Callaghan, "Clustering Data Streams", Proceedings of the 41st

Annual Symposium on Foundations of Computer Science
(FOCS'00), pp. 359- 366, 2000.

26. Eppstein D., Strash D., "Listing All Maximal Cliques in Large Sparse Real-World Graphs", In: Pardalos P.M., Rebennack S. (eds) Experimental Algorithms. SEA 2011. Lecture Notes in Computer Science, vol 6630. Springer, Berlin, Heidelberg, pp 364-375, 2011.
27. Cazals, F., Karande, C., "A note on the problem of reporting maximal cliques", Theoretical Computer Science, 407(1-3), pp. 564-568, 2008.
28. F. Cazals, C. Karande, "An algorithm for reporting maximal c-cliques", Theoretical Computer Science, 349 (3), pp. 484-490, 2005.
29. David Eppstein, Maarten Löffler, Darren Strash, "Listing All Maximal Cliques in Large Sparse Real World Graphs", Journal of Experimental Algorithmics (JEA), Vol. 18, 2013.

30. Anne-Sophie Himmel, Hendrik Molter, Rolf Niedermeier, Manuel Sorge, "Adapting the Bron-Kerbosch Algorithm for Enumerating Maximal Cliques in Temporal Graphs", Social Network Analysis and Mining, pp. 7-35, 2017.
31. Xiaohui Zhao, Sai Guo, and Yanlei Wang, "The Node Influence Analysis in Social Networks Based on Structural Holes and Degree Centrality", IEEE International Conference on Computational Science and Engineering (CSE) and Embedded and Ubiquitous Computing (EUC), pp. 708 - 711, 2017.
32. Felipe Grando, Diego Noble, and Luis C. Lamb, "An Analysis of Centrality Measures for Complex and Social Networks", 2016 IEEE Global Communications Conference (GLOBECOM), pp. 1-6, 2016.
33. Amedapu Srinivas and R. Leela Velusamy, "Identification of influential nodes from social networks based on Enhanced Degree Centrality Measure", 2015 IEEE International Advance Computing Conference (IACC), pp. 1179 - 1184, 2015.

34. Zudha Aulia Rachman, Warih Maharani, and Adiwijaya, "The analysis and implementation of degree centrality in weighted graph in Social Network Analysis", 2013 International Conference of Information and Communication Technology (ICoICT), pp. 72 - 76, 2013.
35. Jie Zhang, Xuerui Ma, Weihao Liu, Yong Bai, "Inferring Community Members in Social Networks by Closeness Centrality Examination", 2012 Ninth Web Information Systems and Applications Conference, pp. 131-134, 2012.
36. Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine", Computer Networks, vol. 30, pp. 107-117, 1998.
37. <http://realitycommons.media.mit.edu/realitymining.html>

APPENDIX A

SUMMARY OF MIT REALITY MINING DATA FEATURES [37]

s(n).mac

The Bluetooth MAC address (unique hardware address) of the subject's phone.

s(n).my_startdate

The date the subject enrolled in the study.

s(n).my_affil

The subject's affiliation:

s(n).my_group

The subject's research group.

s(n).my_imei

The IMEI of the subject's phone:

s(n).my_neighborhood

The subject's neighborhood.

s(n).my_hours

The subject's reported hours at work.

s(n).my_regular

Does the subject report having a regular working schedule.

s(n).my_hangouts

The subject's reported hangouts

s(n).my_predictable

Does the subject report having a predictable schedule.

s(n).my_forget

Does the subject report forgetting his phone at home / work?

s(n).my_battery

How often does the subject report her battery runs out on the phones?

s(n).my_sick

How often does the subject report illnesss?

s(n).my_sickrecently

Has the subject reporting being sick recently?

s(n).my_travel

Does the subject report often traveling?

s(n).my_data

The subject's data plan

s(n).my_plan

The subject's mobile phone plan
s(n).my_provider
The subject's mobile phone provider
s(n).my_minutes
The number of minutes the subject buys each month.
s(n).my_texts
How often the subject reports send text messages.
s(n).my_intros
Whether the subject would like to receive introductions to others.
s(n).my_community
How connected does the subject feel with her community?
s(n).comm
Struct array with fields for each communication event.
(Note that calls to the subject's own phone number is typically associated with checking voicemail.)
s(n).charge
Date and time the phone is charging (1) or unplugged (0).
(convert using datestr)
s(n).active
Date and time the phone has been in use (1) and not in use (0)
s(n).logtimes
Times when the logs were being written (not particularly useful)
s(n).on
When the phone is turned on (1) or off (0)
s(n).locs
Time-stamped tower transitions. [date, areaID.cellID] (0 is no signal)
s(n).all_locs
The unique set of towers seen by the subject.
(areaID.cellID)
s(n).loc_ids
An indexed version of s(n).locs. Towers are replaced by a unique ID.
s(n).device_names
The names of the Bluetooth devices discovered on each scan.
s(n).device_macs
The MAC addresses of the Bluetooth devices discovered on each scan. (Converted to ints using hex2num)
s(n).device_date
The time / dates of each scan
s(n).device_list_names
A list of all the devices names seen by the phone.
s(n).device_list_macs

A list of all the devices Bluetooth MAC addresses seen by the phone (converted from hex to int)

s(n).device_types

The discovered Bluetooth device type (as determined by the standard Bluetooth protocol).

s(n).device_list_types

A list of the device types discovered by the phone

s(n).cellnames

An array of areaID.cellID and the string the user named the location.

s(n).apps

The time each application was started and the total number of times the app was used.

s(n).timeon

The total amount of time the phone has spent recording data (in days)

s(n).app_dates

The set of times when a user started an application

s(n).home_ids

The areaID.cellID of the tower we associate with the subject's home.

s(n).home_nights

The nights when we find the subject at home.

s(n).comm_local

The total amount of local (Boston-based) communication events

s(n).data_mat

Inferred locations at each hour of the day. 1 - home, 2 - work, 3 - elsewhere, 0 - no signal, NaN - phone is off

s(n).my_enddate

The last date in the dataset

s(n).comm_sms

Number of text messages send and received

s(n).comm_sms_date

A list of dates when a SMS was sent or received

s(n).comm_voice

Number of voice calls made and received

s(n).comm_voice_date

The dates of the voice calls.

s(n).comm_data

The number of data sessions initiated on the phone

s(n).comm_data_date

The times when the data sessions were started

s(n).places

The distribution of times the subject was at home, elsewhere, work and with no signal.

s(n).survey_start_n

The date the subject started the survey.
s(n).my_hashedNumber
The subject's hashed phone number.

CURRICULUM VITAE

Heba M. Elgazzar

EDUCATION

- *Ph.D. Computer Science and Engineering*
CECS Department, Speed School of Engineering
University of Louisville, Louisville, KY, USA
GPA: 3.933 / 4.0
Dissertation: Network Science Algorithms for Mobile Networks
Expected Graduation Date: Spring 2018
- *M.Sc., Computer Science*
CECS Department, Speed School of Engineering
University of Louisville, Louisville, KY, USA
GPA: 3.889 / 4.0
Thesis: Data mining approach to detect solar loops.
December 2007
- One year of Course Study for M.Sc. in Electrical Engineering
ECE Department, Faculty of Engineering
Zagazig University, Egypt
September 2000 – July 2001
- *B.Sc., Electrical Engineering*
ECE Department, Faculty of Engineering
Zagazig University, Egypt
May 1999

RESEARCH AND WORK EXPERIENCE

- *Instructor of Computer Science (Full time)* January 2017 – present
Department of Computer Science and Information Systems
School of Engineering and Information Science
Morehead State University, Morehead, KY, USA

Work includes teaching computer science courses at the graduate and undergraduate levels, curriculum development, conducting research with undergraduate students, advising students, and participating in service activities.

Research Areas: Network Science, Data Science, Data Mining, Social Networks, Network Analytics, Machine Learning, Image Mining, and Image Processing
- *Ph.D. student (Full time), Computer Engineering and Computer Science Department, Speed School of Engineering, University of Louisville.* August 2015 – December 2016

Research Areas: Network Science, Social Networks, Machine Learning, Network Analytics, and Data Mining.
- *Instructor of Computer Science (Full time)* August 2011-August 2015
Department of Mathematics, Computer Science, and Physics
College of Science and Technology
Morehead State University, Morehead, KY, USA

Work includes teaching computer science courses, curriculum development, conducting research with undergraduate students, advising students, and participating in service activities.

Research Areas: Network Science, Data Science, Data Mining, Social Networks, Network Analytics, Machine Learning, Image Mining, and Image Processing
- *Technology Coordinator and Instructor of Computer Applications (Volunteer – Part time)* November 2006-October 2007
Rowan County Public Library, Morehead, KY.
- *Graduate Research Assistant* January 2005-
• *Member of Knowledge Discovery and Web Mining Lab* August 2006
CECS Departments, Speed School of Engineering
University of Louisville, Louisville, KY, USA

Worked in a project related to Solar Loop Mining using Data Mining and Image Mining Algorithms (this research was supported by NSF and NASA).

TEACHING EXPERIENCE

- *Instructor of Computer Science* January 2017- present
School of Engineering and Information Science

Morehead State University, Morehead, KY
Courses:
 - Data Mining (Graduate Course)
 - Programming of Mobile Devices
 - First Year Seminar
 - Game Implementation Technique
 - Game Prototype Design and Implementation
 - Introduction to Engineering Technology

- *Instructor of Computer Science* August 2011-August 2015
College of Science and Technology

Morehead State University, Morehead, KY
Courses:
 - Introduction to Computer Science
 - Data Structures
 - Advanced Data Structures and Algorithms
 - Game Implementation Technique
 - Game Prototype Design and Implementation

HONORS AND AWARDS

- *The Outstanding Teaching Award*, Department of Mathematics and Computer Science, Morehead State University, May 2013.
- *Who's Who Among Students in American Universities and Colleges*, 2006 Edition.
- Invited member, *Golden Key International Honor Society*.

- Certificate of recognition in the Engineers' Day from the Computer Engineering and Computer Science Department, University of Louisville, 2005 and 2006.

PUBLICATIONS, POSTERS, AND PRESENTATIONS

- Heba Elgazzar and Adel Elmaghraby, "Network Science Algorithms for Mobile Network Analytics", Proceedings of the IEEE SoutheastCon 2017, pp. 1-7, Charlotte, NC, 2017.
- William Burriss and Heba Elgazzar, "Comparison between Different Classification Techniques", presented at the 2015 MSU Annual Celebration of Student Scholarship, Morehead State University, April 2015.
- Jorge Chang and Heba Elgazzar, "Image Recognition and Classification Using Multi-Histogram Approach", presented at the 2014 National Conferences on Undergraduate Research (NCUR 2014).
- Jorge Chang and Heba Elgazzar, "Quantum Computing Implementation of Shor's Algorithm", presented at the 2014 MSU Annual Celebration of Student Scholarship, Morehead State University, April 2014.
- Scott Blankenship and Heba Elgazzar, "Data mining on Medical Data", presented at the 2014 MSU Annual Celebration of Student Scholarship, Morehead State University, April 2014.
- Wesley Kelsey and Heba Elgazzar, "PeerSync: Synchronizing without Servers", presented at the 2014 MSU Annual Celebration of Student Scholarship, Morehead State University, April 2014.
- Jorge Chang and Heba Elgazzar, "Content-based Image Retrieval with Multi-histogram Approach", presented at the 99th Annual Meeting of the Kentucky Academy of Science, Computer and Information Sciences Section, Morehead, KY, November 9, 2013. (*Presented by Jorge Chang and he was the first place winner in the undergraduate research competition*).
- Heba Elgazzar, "Data mining Approach to Detect Solar Loops", M.Sc. Thesis, University of Louisville, December 2007.
- Durak, N.; Nasraoui, O.; Gomez, J.; Gonzalez, F.; Elgazzar, H.; Sellah, S.; Rojas, C.; Schmelz, J.; Roames, J.; Nasraoui, K.; "Mining Coronal Loops in Solar Images from the SOHO collection", Annual Meeting of the North American Fuzzy Information Processing Society, 2007. (NAFIPS '07), P. 514 – 519, 24-27 June 2007.
- Heba Elgazzar and Olfa Nasraoui, "Mining Solar Images to Support Astrophysics Research: Phase I", Poster in the 2006 Engineers' Day, Speed School of Engineering, University of Louisville.
- Heba Elgazzar and Olfa Nasraoui, "Needle in a Haystack: Mining NASA's Web Databases", Poster in the 2005 Engineers' Day, Speed School of Engineering,

University of Louisville, 2005 (Received Certificate of Recognition, CECS Dept., University of Louisville).

- Heba Elgazzar and Olfa Nasraoui, “Solar Loop Mining for the Coronal Heating Problem”, Presentation in the Knowledge Discovery and Web Mining Lab Seminar Series, University of Louisville, 2005.

GRANTS / FUNDING

- “Comparison between Different Classification Techniques”, Support for an Undergraduate Research Fellow, Morehead State University (Funded 2012 –2015).
- “Optical Character Recognition”, Support for an Undergraduate Research Fellow, Morehead State University (Funded 2014 –2015).
- “Content-Based Image Retrieval Using Machine Learning Techniques”, Support for an Undergraduate Research Fellow, Morehead State University (Funded 2012 –2014).

RECENT SUPERVISED UNDERGRADUATE STUDENTS

- William Burris, “Comparison between Different Classification Techniques”.
- Jorge Chang, “Content-based Image Retrieval”.
- Steven Harris, “Game Engine Development”.
- Garret Gullet, “Optical Character Recognition”.
- Jorge Chang, “Quantum Computing Implementation”.
- Scott Blankenship, “Mining Medical Data”.
- Wesley Kelsey, “Synchronizing without Servers”.
- Jordan Holt, “Web Application Development”.

PROFESSIONAL MEMBERSHIPS

- Institute of Electrical and Electronics Engineers (IEEE).
- IEEE Computer Society.
- IEEE Young Professionals.
- IEEE Women in Engineering.
- IEEE Biometrics Council.

- Association for Computing Machinery (ACM).
- Kentucky Academy of Science (KAS).
- Egyptian Syndicate of Engineers.

SERVICE AT MOREHEAD STATE UNIVERSITY

- Faculty Advisor and Sponsor of the MSU Association for Computing Machinery (ACM) Student Chapter, January 2017-present.
- Member of the Computer Science Committee (August 2011 – August 2015).
- Advisor for CS students' senior capstone projects (January 2012 – August 2015).
- Directed exploration activities for Mathematics, Physics, and Advanced Technology Exploration (MPATE) Day (2011, 2012, 2013, and 2014).
- Served as a judge in the MSU Annual Celebration of Student Scholarship, Morehead State University (April 2013 and April 2014).