UNIVERSITY OF CALIFORNIA,
IRVINE

Mobile Data Analysis For Smart City Applications

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Networked Systems

by

Blerim Cici

Dissertation Committee:
Associate Professor Athina Markopoulou, Chair
Professor Nalini Venkatasubramanian
Doctor Nikolaos Laoutaris

2016

DEDICATION

To my family and friends,
without whom this work
would not have been possible.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

ACKNOWLEDGMENTS

I would like to thank my co-workers, professors, family and friends that have made this dissertation possible.

I feel extremely lucky for doing my Ph.D. at the Networked Systems program of UC Irvine, and for having Professor Athina Markopoulou as my adviser. I own deep gratitude to Professor Markopoulou for her guidance and support; without her patience and encouragement this work would not have been possible. Also, I own deep gratitude to Dr. Nikolaos Laoutaris for giving me the opportunity to work in one of the most prestigious research labs in the area of Computer Networks, Telefonica Research Labs, and for his guidance and support during my internship there. Thanks to Professor Carter T. Butts and Dr. Minas Gjoka for their immeasurable help in defining and implementing the Urban Ecology project, which is an important part of this thesis. Thanks to Professor Alexander Ihler and Emmanouil Alimpertis for their help in defining and executing the cell-to-cell activity prediction project. Thanks to Dr. Enrique-Frias Martinez for his feedback and help in the Ridesharing projects. Also, Thanks to Dr. Fabio Soldo for his guidance and support during my internship in Google. Thanks to my lab-mates and collaborators Professor Hulya Seferoglu, Dr. Anh Le, Dr. Lorenzo Keller, Professor Christina Fragouli, Professor George Xylomenos, Dr. Pegah Sattari, Dr. Maciej Kurant, Dr. Francesco Malandrino, Dr. Abinesh Ramakrishnan, Dr. Chun Meng, Dr. Omer Nebil Yaveroglu, Anastasia Shuba, Balint Tillman, and Luca Baldesi for their helpful discussions, suggestions and feedback. Thanks to Professors Nalini Venkatasubramanian, Amelia Regan for serving on my thesis committee and my candidacy committee.

A thank you is in order for all the institutions that have financially supported me through my studies. Thanks to Networked Systems program for supporting a big part of my studies. Thanks to Telefonica Research Labs and Google for supporting me during my internships. Thanks to the National Science Foundation for supporting me through the NSF CDI award 1028394.

I would like to thank the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) for granting me permission to disclose copyrighted materials presented on my publications.

Finally, I would like to thank my family and friends for their unconditional support and encouragement, and especially my parents, Vangjush and Marjana, my sister Aurora, and my girlfriend, Dr. Anna Papio Toda. They have been key to my success.

CURRICULUM VITAE

Blerim Cici

EDUCATION

Doctor of Philosophy in Networked Systems **2016**
University of California, Irvine *Irvine, California, USA*

Master of Science in Networked Systems **2012**
University of California, Irvine *Irvine, California, USA*

Bachelor of Science in Informatics **2009**
Athens University of Economics and Business *Athens, Greece*

RESEARCH EXPERIENCE

Graduate Research Assistant **2013–2016**
University of California, Irvine *Irvine, California, USA*

Intern **2012–2013**
Telefonica Research Labs *Barcelona, Spain*

Graduate Research Assistant **2009–2012**
University of California, Irvine *Irvine, California, USA*

TEACHING EXPERIENCE

Teaching Assistant **2012–2013**
University of California, Irvine *Irvine, California, USA*

ABSTRACT OF THE DISSERTATION

Mobile Data Analysis For Smart City Applications

By

Blerim Cici

Doctor of Philosophy in Networked Systems

University of California, Irvine, 2016

Associate Professor Athina Markopoulou, Chair

Thanks to the pervasiveness of smartphones and their applications there is an abundance of data generated from mobile devices. These data are either actively contributed by the users (*e.g.*, Foursquare check-ins), or can be passively inferred from the mobile phone activity (*e.g.*, user's location during a phone call). Since, smartphones are constantly with their users, mobile data reflect the underlying human activity and they provide rich information about spatio-temporal and social patterns. This can be used to enable a variety of new services.

This dissertation is focused on how to mine mobile phone data to improve a variety of Smart City applications. In particular, we focus on Call Description Records (CDRs) that are generated every time a user makes a phone call. Also, users' phone calls reveal social information (*e.g.*, whom they call). The spatio-temporal information in mobile data reflects how users move in the city (*i.e.*, users' trajectories) and in which areas they spend their time during the day. In this dissertation we use their rich and unique combination of insights into human dynamics in order to: (1) understand and improve transportation in a city via ridesharing, (2) characterize various areas of the city, as well as how these areas interact with each other (Urban Ecology), and (3) predict communication between different areas of the city in order to improve provisioning in cellular infrastructure.

First, we use CDRs to assess the potential of ridesharing. Our offline analysis based on large CDRs and other data sets from four different cities indicates that ridesharing has a great potential considering spatio-temporal and social constraints the users might have when sharing a ride. Moreover, we design and implement an online ridesharing system (ORS), with emphasis on scalability.

Second, we use CDRs to infer features of urban ecology (*i.e.*, social and economic activities, and social interaction). We present a novel approach that consists of time series decomposition of aggregate cell phone activity per unit area using spectral methods, and clustering of areal units with similar activity patterns. We validate our methodology using external ground truth data that we collected from municipal and online sources.

Finally, we use CDRs to predict cell-to-cell mobile traffic. Traffic prediction is crucial for provisioning and virtualization of cellular architectures. We build a traffic predictor using state-of-the art machine learning techniques. Our predictor is based on key insights that we got after examining the data and it achieves accuracy of 85% (while the baseline achieves 80%). Also, by giving higher weight to false positives, which is important for network operators, we can achieve a recall of 94%.

Chapter 1

Introduction

1.1 Motivation

1.1.1 Human Patterns Reflected in Cell Phone Data

Thanks to the pervasiveness of smartphones and their applications there is an abundance of data contributed from mobile devices. These data are either actively generated by the users (*e.g.*, Foursquare or Facebook check-ins), or can be passively inferred from the mobile phone activity (*e.g.*, user's location during a phone call). Since, smartphones are constantly with their users, mobile data reflect the underlying human activity and they provide rich information about spatio-temporal and social patterns. This can be used to enable a variety of new services.

Call Description Records (CDRs) [1] are the most common mobile data available. They are generated every time a mobile subscriber makes a phone call and they come at no extra cost to the operator since they are maintained for billing purposes. Moreover, they are available

[1] Also known as Call Data Records, or Call Detail Records.

for a large portion of the population, and they can be used to study human mobility, social networks, and urban ecology.

1.1.2 Overview of CDRs

Cell phone networks are built using a set of Base Transceiver Stations (BTS) that are in charge of communicating with cell phone devices. The area covered by a BTS tower is called a cell. The size of a cell varies from a few hundred square meters in an urban environment to up to 3 square kilometers in rural areas. At any given moment, one or more BTSs can give coverage to a cell phone. Whenever an individual makes a phone call, the call is routed through a BTS in the area of coverage. The BTS is assigned depending on the network traffic and on the geographic position of the individual.

Call Description Records (CDRs) are generated when a mobile phone initiates or receives a phone call or uses a service (*e.g.*, SMS, MMS, etc.). Information regarding the time/date and the location of the nearest cell tower are then recorded. More specifically, CDR entries consist of the following main fields: (1) the originating number (2) the destination number (3) the time and date of the call (4) the duration of the call (5) the latitude and longitude of the cell tower used by one, or both, phones numbers (note that cell phone companies save CDR records only for their customers). The following example shows part of the information contained in CDR records:

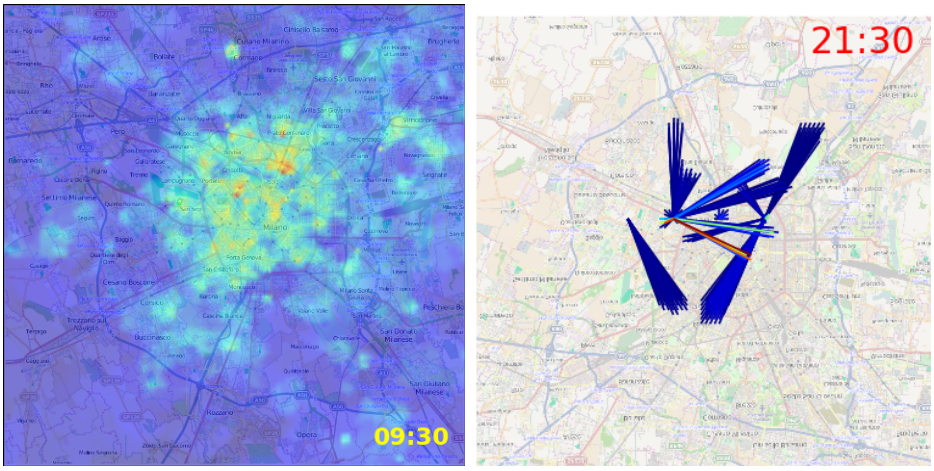
```
#caller,#callee,start time,tower coordinates,
-------------------------------------------------
u1,  v1,2009-12-01 00:44:13,(40.421377,-3.698631)
u2,  v2,2009-09-15 23:33:19,(40.441566,-3.697189)
u3,  v3,2009-12-10 14:33:29,(40.414634,-3.709735)
u4,  v4,2009-12-10 19:53:09,(40.419974,-3.630850)
```

These records are logged for pricing purposes, so they come at no extra cost to the cellular operator. Note that no information about the exact position of a user is known, since cell phone data provide coarse location accuracy — a few hundred meters from the cell tower for city center, and up to 3 km in rural areas. For privacy reasons, no contract or demographic data are made available to researchers with the CDRs, and the originating and destination phone numbers were anonymized.

One can analyze CDRs at different levels of granularity. In this thesis, we refer as (1) "Individual CDRs" to CDR records that provide per-user information; and (2) "Aggregate CDRs" to indicate the fact that the CDRs have been aggregated over a number of users in a spatial and/or time unit.

Figure 1.1: Example of spatio-temporal information extracted from individual CDRs. The figure shows the locations of a particular user reported in CDR records. One can infer two important locations (possibly home and work) and the trajectory of this user (indeed coinciding with a highway in the city).

Individual CDRs: Individual CDRs provide the spatio-temporal information and the phone calls of individual users. Therefore, they constitute samples of individuals' trajectories, as well as samples of their social network. Such, rich samples of can shed light into how people move around the city (*i.e.*, their daily home-to-work trajectories) and how they interact with other mobile users (see Fig. 1.1). Although the user (caller and callee) ids are typically anonymized, the risk for privacy breach is significant for the telecommunication companies



(a) Aggregated Activity Heat-map (b) Intra-city Communication

Figure 1.2: These two figures demonstrate the volume of mobile activity that Aggregate CDRs contain. Fig. 1.2(a) shows a heat-map of the cell phone activity, *i.e.*, the total volume per areal unit in a 10min interval. Fig. 1.2(b) shows an example of communication volume between pairs of cells (areas) in the city.

to make them publicly or share with with collaborators.

Aggregate CDRs: CDRs can be aggregated spatially and/or temporally; they are typically aggregated by dividing the geographical area where they have been generate, *e.g.*, a city, into a grid; for example a 100x100 grid will divide a city into ten thousand grid-squares where each one of them corresponds to a few city blocks (up to a few kilometers depending the size of the city and the number of grid-squares). Aggregate CDRs pose low risk to individuals' privacy and they are often made publicly available [10]. In addition, aggregate CDRs are of interest on their own right as they reveal interesting patterns at different granularities, such as city blocks, time periods etc (See Fig. 1.2).

1.2 Contributions

This thesis investigates how to mine mobile phone data to improve a variety of Smart City applications. The spatio-temporal information in the data reflects how users move in the

city (*i.e.*, users' trajectories) and in which areas of the city they spend their time during the day, while their phone calls shows whom do they call (from a specific location at a specific time). The data sets used in this thesis are described in their corresponding chapters. Next, we outline the contributions of this thesis.

1.2.1 Mining Aggregated CDRs

Because of their rich and unique combination of insights into human activities in a city, mobile phone data can be used to: (1) characterize various areas of the city, as well as how these areas interact with each other (*i.e.*, Urban Ecology), and (2) predict communication between different areas of the city (*e.g.*, in order to improve provisioning in cellular infrastructure).

Urban Ecology

The goal of this part of the thesis is to infer features of urban ecology (*i.e.*, social and economic activities, and social interaction) from spatio-temporal cell phone activity data. We present a novel approach that consists of (i) time series decomposition of the aggregate cell phone activity per unit area using spectral methods, (ii) clustering of areal units with similar activity patterns, and (ii) external validation using a ground truth data set we collected from municipal and online sources. A key to our approach is the spectral decomposition of the original cell phone activity series into seasonal communication series (SCS) and residual communication series (RCS). The former captures regular patterns of socio-economic activity within an area and can be used to segment a city into distinct clusters. RCS across areas enables the detection of regions that are subject to mutual social influence and of regions that are in direct communication contact. The RCS and SCS thus provide distinct probes into the structure and dynamics of the urban environment, both of which can be obtained from the same underlying data. We illustrate the effectiveness of our methodology by applying it

to aggregate Call Description Records (CDRs) from the city of Milan. This is described in Section 2.3 of the thesis.

Activity Prediction

In this part of the thesis, we analyze data from a large mobile phone provider in Europe, pertaining to time series of aggregate communication volume $A_{i,j}(t) > 0$ between cells i and j, for all pairs of cells in a city over a month. We develop a methodology for predicting the future (in particular whether two cells will talk to each other $A_{i,j}(t) > 0$) based on past activity. The data set is sparse, with 80% of the values being zero, which makes prediction challenging. We formulate the problem as binary classification and, using decision trees and random forests, we are able to achieve 85% accuracy. By giving higher weight to false positives, which cost more to network operators, than false negatives, we improved recall from 40% to 94%. We briefly outline potential applications of this prediction capability to improve network planning, green small cells, and understanding urban ecology, all of which can inform policies and urban planning. This work is described in Section 2.4 of the thesis.

1.2.2 Mining individual CDRs

We use individual CDRs to understand and improve transportation in a city via ridesharing.

Assessing the Potential of Ridesharing using Mobile Data

This part of the thesis assesses the potential of ride-sharing for reducing traffic in a city, based on mobility data extracted (1) from CDRs, for the cities of Madrid and Barcelona (BCN), and (2) from OSNs, such as Twitter and Foursquare (FSQ), collected for the cities of New York (NY) and Los Angeles (LA). First, we analyze these data sets to understand mobility

patterns, home and work locations, and social ties between users. Then, we develop an efficient algorithm for matching users with similar mobility patterns, considering a range of constraints, including social distance. The solution provides an upper bound to the potential decrease in the number of cars in a city that can be achieved by ride-sharing. Our results indicate that this decrease can be as high as 31%, when users are willing to ride with friends of friends. And up to 51% if they are willing to share a ride with anyone within their spatio-temporal constraints. This contribution is described in Section 3.2 of the thesis.

Designing a Scalable Online Ridesharing System

Ridesharing systems have the potential to match travelers with similar itineraries and time schedules, and to bring significant benefits to individual users and to the city as a whole. In this part of the thesis, we build on our previous off line analysis and we design and evaluate *ORS*- an Online Ridesharing System, where drivers and passengers send their requests for a ride in advance, possibly on a short notice. The system consists of two components: the constraint satisfier and the matching module. The constraint satisfier takes as input the itineraries and spatio-temporal constraints of drivers and passengers and provides feasible (driver, passenger) pairs. We achieve scalability by designing a constraint satisfier using a road networks data structure, specifically optimized for our spatio-temporal queries. We show that our specialized module is more scalable than generic databases: query time increases 4.65x slower with the number of users. We use the feasible pairs found by the *satisfier* to define a bipartite graph between possible drivers and passengers, with edge weights representing the length of the shared trip of a pair. The matching module takes as input the weighted bipartite graph and returns the maximum weighted matching (MWM), which captures the objective of real-world ridesharing systems (such as `Lyft Carpool`). We propose an efficient algorithm to solve the MWM problem, which is 51% better than greedy heuristics used by many real systems. Furthermore, the system is designed to handle requests that

arrive on-line, via efficient queries of feasible pairs and incremental updates of the matching solution. We evaluate the entire ORS system using real mobile datasets to extract driver trajectories and passenger locations in urban environments. We show that ORS can provide a ridesharing recommendation to individual users with a sub-second query response time, even at high workloads. We also evaluate the sensitivity of *ORS* performance to various parameters, which provides insights for the design of practical ridesharing systems. This contribution is described in Section 3.3 of the thesis.

Chapter 2

Mining Aggregate CDRs for Urban Ecology and Activity Prediction

2.1 Introduction

The modern urban environment is a complex ecosystem, characterized by distinct geographical regions sharing common patterns of socio-economic activity, infrastructure, social cohesion, *etc.*, [20, 79]. Further, some of these regions are strongly interacting (via mobility, communication, *etc.*,), while others are relatively isolated from one another [45, 51]. We refer to this intra-urban structure as "urban ecology." Historically, the detection of urban ecology has been difficult as it usually requires extensive local knowledge of the area in question and investigation using time-consuming and expensive techniques (*e.g.*, informant interviews, ethnographic observation, *etc.*,). This poses challenges for urban governance — *e.g.*, urban planning, infrastructure management, administration, and law enforcement—particularly in an era of increasingly rapid urban growth and change (*e.g.*, due to shifting patterns of immigration and economic activity). Likewise for the private sector, effective siting of businesses

requires extensive knowledge of the urban landscape; in a global economy, obtaining such knowledge via years of experience "on the ground" in a given location may be expensive or impractical.

It is estimated that within the next forty years, two-thirds of the world's population will be living in expanding urban centers, and the level of urbanization is expected to increase in all major areas of the developing world [7]. Given this, there is clearly a need for methods that will cheaply yield up-to-date information on urban environments, without extensive on-the-ground investigation. The research area of Smart City and Urban Computing provides such promising methods. Smart City refers to the use information and communication technology in order to understand and coordinate the environment, systems, people and things in the city; Smart City infrastructures and systems as well as intelligent urban computing will help the city administrators better understand and react to the metropolitan needs, challenges, as well as the operation of urban infrastructural systems.

A promising tool for Smart Cities is the use of aggregate geo-located data on communication activity, a resource that is increasingly available given the near-universal penetration of mobile devices within urban populations. Because communication behavior is a fundamental aspect of virtually all social and economic interaction, even aggregated communication data can provide rich information on the types and volumes of activities occurring within a given geographical area, and on the degree of interaction (or lack thereof) between occupants of different areas. Such data is inexpensive, can be collected and distributed in a privacy-preserving manner (*e.g.*, in the case of aggregate activity), and can be monitored to track developments within the urban landscape.

In this chapter we describe two Smart City tools that are based on Aggregated CDRs. First, we show how aggregate mobile communication data can be used to provide information on urban ecology; we provide an approach to this problem that draws on the notion of seasonal decomposition from the field of classical time series analysis. Second, we develop a machine

Name	Period	Source
Milan Activity	Nov.4-Dec.1 2013	Telecom Italia [10]
Milan Cell-to-Cell	Nov.4-Dec.1 2013	Telecom Italia [10]
Universities, Businesses, Parks, Population per area, Sport Centers, Bus stops	Jan.1-Dec.31 2013	City of Milan [3]

Table 2.1: Data Sets

learning techniques for cell-to-cell mobile traffic prediction based on past cellular records; this prediction will help understand city-wide human activity patterns as manifested in cellular activity, and it can enable network provisioning and control.

2.2 Data

The first two data sets were made publicly available by Telecom Italia Mobile as a part of the *Big Data Challenge* [10] competition. They consist of telecommunications activity records in the city of Milan. In this chapter, we focused on a 4-week period of November 2013. The decisions about spatial and temporal granularity in the dataset, were already made by the dataset provider (Telecom Italia) when they made the data sets available for the *Big Data Challenge*, and were out of our control.

In particular, the city of Milan, an area of 550 km^2, was divided into a 100×100 square grid. Each grid-square has the same dimensions: a side length of 0.235 km and an area of 0.055 km^2. This is the areal unit we use throughout the chapter, and we refer to it as a "cell". The temporal unit is the 10-minute interval.

In the Milan Activity dataset, the activity is aggregated within each cell for each 10-minute interval. Each activity record consists of the following entries: cell ID, time-stamp of 10-minute time slot, country code, incoming SMS activity, outgoing SMS activity, incoming call activity, and outgoing call activity. According to the data set release information [10],

each activity value corresponds to the level of interaction of all the users in the cell with the mobile phone network, *e.g.*, the higher the number of outgoing calls made by the users, the higher the outgoing Calls activity. Values of incoming/outgoing Call and SMS activity are normalized and have the same scale. Therefore, we sum up the latter four values,[1] over all country codes in order to come up with a single value which describes the total activity volume in a cell during each 10-minute time slot. Fig. 2.1 shows an example of the original time series for two cells over a one week-period.

The Milan Cell-to-Cell dataset contains information regarding the "directional interaction strength" (as per terminology in [10]) between two cells, based on the calls exchanged between users in them. Each activity record consists of the following fields: ID of cell i where call was initiated from, ID of unit j where the call was made to, time slot, and value of directional strength from i to j. We can obtain the total directional strength from cell i to cell j, by summing up over all time slots in the 4-week period.

The third data set is used a ground truth for the methodology that we develop in the first section of this chapter. We obtained it by crawling the Municipality of Milan's Open Data website [3].

2.3 Urban Ecology

2.3.1 Introduction

In this section, we show how to use aggregate CDRs to infer elements of urban ecology. Our intuition is that human activity results in mobile activity, and therefore mobile activity could be used as a "signature" to infer human activity. Our approach draws on the notion of

[1] This aggregation helps avoid data sparsity.

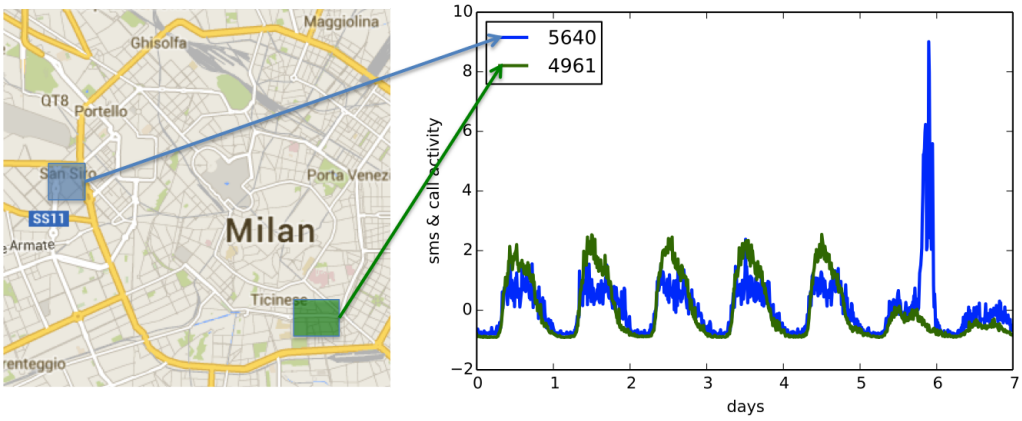


Figure 2.1: Cell phone activity series (normalized) for two grid cells of Milan, for the week of 11/4/2013–11/10/2013. Cell 5640 is located close to the San Siro stadium, while cell 4961 is located in a university region. Differences in seasonal patterns (weekday/weekend) reflect stable differences between the university and stadium environments. The spike on day 6 corresponds to 11/9/2013, when Internazionale, one of the two major football teams of Milan, played against Livorno in San Siro; this event is captured as a local perturbation in the stadium area time series. Here, we exploit both types of information.

seasonal decomposition from the field of classical time series analysis. By decomposing communication data across time, frequency, and space, we create distinct time series that provide information on routine activities and on deviations from those routines. Subsequent analysis of the resulting multivariate time series allows for the identification of socio-economically distinct regions within the urban environment, identification of socially interacting regions, and other goals of interest to the analyst. As we show, these analyses can be performed efficiently and in an unsupervised or semi-supervised manner, facilitating their use in settings for which the analyst has only limited resources for additional data collection.

Key idea: It is well known that time series can be decomposed into three components (the *classical decomposition* [22]): a *trend*, representing systematic, non-periodic change over the time scale of observation; a *seasonal* component, representing systematic periodic variation in the phenomenon of interest (possibly with multiple characteristic frequencies); and a *residual* component, representing variation due to idiosyncratic factors and exogenous shocks. Many social processes either generate gradual, systematic change (*e.g.*, population

growth) or are strongly periodic (*e.g.*, on hourly, daily, weekly, or annual time scales); the trend and seasonal components of the time series "selects out" such processes, allowing them to be either measured or removed from analysis, if desired. Many other social processes are characterized by short-term responses to exogenous perturbations, and (when integrated over all perturbations) have little or no systematic component. Information about these processes is contained in the residual series, which can thus be used to study them without contamination from their systematic counterparts.

Data: The data that we use in this study consists of time series of aggregate cell phone traffic sent or received by persons within small areal units in the city of Milan (see Fig. 2.1), for approximately one-month period. In particular, we use aggregate call-description-records (CDRs) and aggregate SMS activity per area unit, made available for the *Big Data Challenge* [10] competition. In addition, we collected ground truth data from the municipality of Milan and online sources, containing elements such as universities, residential areas, sport centers, parks, *etc.*

Methodology and Results Overview: We apply our approach to the cell phone activity series, using the decomposed series to characterize distinct aspects of urban ecology. We proceed as follows:

First, we begin by decomposing the original cell phone activity series for each areal unit into seasonal and residual components. The decomposition is done using FFT, with the seasonal component corresponding to high-amplitude frequencies and the residual component corresponding to the deseasonalized series in the time domain. The seasonal communication series (SCS) are due to typical patterns of socio-economic activity within an area; *e.g.*, a university generates higher traffic during weekdays and lower traffic during weekends and holidays than a residential neighborhood. The residual communication series (RCS), on the other hand, can represent irregular traffic (*e.g.*, an area may have higher traffic than usual due to a protest or sporting event, or lower traffic than usual due to a strike) and/or due to

the influence of one area on another (*e.g.*, due to mobility and/or social interaction).

Second, we perform hierarchical clustering of different areas based on the different time series and we validate the results using the ground truth data. We show that our SCS clustering scheme successfully segments areas dominated by distinct types of socio-economic activity, and allows for discovery of regions whose activity patterns differ markedly from the rest of the city. The results compare favorably with state-of-the-art approaches such as [82], since SCS can incorporate regular patterns occurring on any time scale; by contrast, state-of-the-art methods estimate regular patterns using average weekday and weekend days (evaluated using binned averages), and cannot therefore exploit activity patterns that occur across multiple days. In addition to using the SCS to identify regular patterns, we show that its counterpart, the RCS, enables the detection of regions that are subject to mutual social influence or in direct communication contact; this was not previously possible using mere activity data. More specifically, we show that the structure of lagged spatial correlations in RCS across areas allows for the detection of regions that are subject to mutual social influence (i.e., disruptions in one area propagate to the other), and of regions that are in direct communicative contact. We validate the latter by showing that RCS correlations between areas are significantly related to the volume of inter-area cell phone traffic, and that this relationship is substantially stronger than for SCS.

In summary, the RCS and SCS provide distinct probes into the structure and dynamics of the urban environment, both of which can be obtained from the same underlying communication data.

2.3.2 Related Work

This section summarizes the most relevant related work in the intersection between urban dynamics and human activity data. Data generated by human activity can be divided into

two broad categories: (1) self-reported data and (2) behavioral traces. In self-reported data, users decide to report their semantically annotated location via check-ins e.g. in Foursquare a user selects a venue from a list of venues that are detected nearby his location. In behavioral traces, users are passively monitored and do not actively select the information that is being revealed. Cell phone activity patterns are an example of behavioral traces, in which some aspects of the users' behavior are *fully* revealed, but without any *semantic information* i.e. the user location is not annotated to indicate venue or category type. This work uses cell phone activity patterns.

Behavioral traces. Cell phone activity patterns, also commonly known as Call Description Records (CDR), capture important aspects of human activity in a city [76], and they have been used to study human mobility, social networks, and urban ecology. Since the focus of this chapter is on the latter, we mainly review related work in that area, and we only briefly mention other work in the broad area of cell phone activity analysis.

Toole et al. [86] used aggregated CDRs in order to infer land usage in Boston. They used a supervised learning technique: they built various features from activity series and used them to classify areas of the city into five different categories (residential, commercial, industrial, parks, and other). Their ground truth was a data set of zoning regulations from the municipality of Boston. However, the classifier's accuracy was worse than classifying every area to belong in the dominant category, due to the high percentage of residential areas. In contrast, we use clustering, a form of unsupervised learning, our ground truth data set contains the facilities in each area, and we study the city of Milan.

Soto et al. [82] also followed an unsupervised learning approach to characterize areas in the city of Madrid. They clustered areas of the city based on their activity signature, i.e. the activity pattern for a typical weekday and a typical weekend, where "typical" is defined as average activity in a period of 3 months. They produce five different clusters: industrial and offices, business and commercial, nightlife, leisure and touristic, and residential. We follow

a different clustering approach that facilitates selecting solutions for specific purposes (e.g., segmentation versus anomaly detection), and employ a more general time series decomposition that can be used in settings with regular patterns that do not fit into the typical weekday/typical weekend day typology. In Section 2.3.6 we compare our spectral approach to that of [82] and show that our approach allows us to detect additional information that is lost under averaging. Additionally, our work differs from [82] in that we make use of residual fluctuations in the time series, which can be used to detect interactions between areas. We note that our clustering and residual analysis techniques could be applied to the averaging scheme used by [82], and many aspects of our approach are hence complementary to this work.

Similar decompositions of activity series have been applied in other settings as well. Calabrese et al. [25] applied eigendecomposition to extract features from Wi-Fi time-series, and then used those features, produced from the top 4 eigenvectors, to cluster access points with similar traffic. We apply time series decomposition on a different type of series and we also take advantage of the residual communication to highlight important aspects of the data.

Finally, CDRs have been used for human mobility analysis, [48], [44], [24] as well as for studying social interactions [42]. These studies have answered various questions, including: what is the potential of ride-sharing for reducing traffic in a city [34]; which are the poorest areas of a city [80], etc.

Self-reported data. Work on urban dynamics and self-reported locations is primarily focused on Foursquare check-ins [38, 67, 68]. The authors of [68] use the categories of Foursquare check-ins in order to cluster similar areas of the city, i.e. two areas are similar if their normalized check-ins are also similar (e.g. both have 70% restaurant and 30% work check-ins). More specifically, each area is represented by a vector of category check-ins aggregated over a time period. Spectral clustering with a cosine similarity measure is used to create clusters of similar areas. However, the paper provides no external validation of

the clustering results. In contrast, our evaluation results make use of a separately obtained ground truth data from the official city records and our method does not require semantically annotated location information.

The authors of [38] use the same clustering approach as [68], with some modifications in the distance function, and they validate they results through interviews with city residents. However, the distance function in [38] requires per-user and per-venue check-in information whereas our method is suitable with spatially aggregated data over all users in the same area.

[90] considers that the function of an area is a combination of two aspects: places of interest (POI) categories and human mobility data (e.g. number of people arriving or leaving) in a region. They cluster similar areas using a topic model-based method that combines both aspects. They evaluate their method by using well-known locations, and interviews with residents of the city. However [90] aggregates the human mobility data in typical weekday/weekend fashion, an approach that we show looses information. Additionally, their method requires as input POI categories which, unlike ours, assumes existing knowledge about the city.

2.3.3 Ground Truth Data

We collect additional data sets at the Municipality of Milan's Open Data website [3], in order to use them as ground truth, for external validation of clustering, in Section 2.3.6. For each cell defined in the main data sets, we gathered the following *category* information: population, %green area, #sport centers, #of universities, #businesses, and #bus stops. The blue shaded area in Fig. 2.2 marks the neighborhoods (or "local identity units") in the city of Milan, for which category information is available.

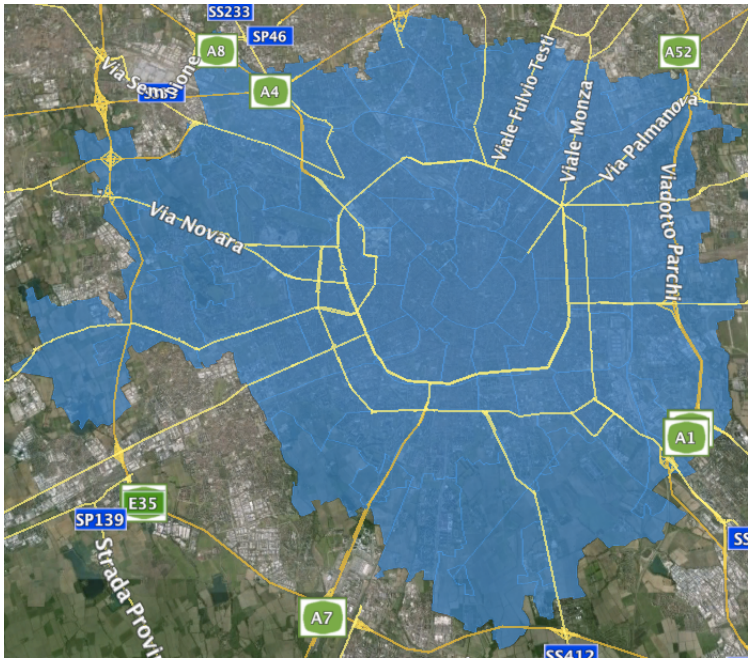


Figure 2.2: Local identity units of Milan. The blue shaded area shows the part of the city for which we have ground truth information. Ground truth information consist of information regarding facilities in each area, as well as census data.

Gathering the category data and putting them in a ready-to-use format was non-trivial. First, we performed basic data cleaning and post-processing of the collected data. This involved removing duplicate entries in each category, transformation of coordinates from the Italian Gauss-Boaga projection to standard latitude-longitude, and mapping of addresses to latitude-longitude using Google Maps API. Second, we assigned every ground truth element in each category to the corresponding cell. Information that appeared as a single latitude-longitude coordinate was easy to assign to a single cell; those included businesses, universities, bus stops and sport centers. However, assigning categories such as green areas (which appear as geometric shapes with multiple points) and demographic information (which is reported on top of the local identity units) to the grid cells was more challenging. In the case of the green areas, we calculate the overlapping area between a specific cell and a given green area as a percentage of the cell.[2] Fig. 2.3 shows the overlap between cells and local identity

[2]For example, if the park was large and the whole cell was inside it then the cell was considered to be 100% green space. For demographic data, we calculate the overlapping area between a cell and a local identity unit, we subtract the green-space area, and we assign part of the population of the local identity unit to the

Figure 2.3: Cells inside the local identity unit of Pagano. Some cells have full overlap, while others have only partial overlap. The Population of Pagano will be spread uniformly to the overlapping cells proportionally to the overlapping area (between a cell and Pagano). Also, the population of a cell may be reduced even further if the cell contains green areas. A cell can receive population from multiple local identity units.

areas.

2.3.4 Activity in Time and Frequency Domains

Let $S = [1, n]$ be the set of all grid cells, where $n = 10^4$. Also, let $T = \{t_1, t_2, ..., t_m\}$ be the set of all time units (10-minute time slots as defined by the data provider) in our 4-week period, where $m = 4032$. Finally, for each cell $i \in S$, we denote the original activity series as $O_i(T) = \{o_i(t_1), o_i(t_2), ..., o_i(t_m)\}$.

The original time series is expected to have strong periodicity, since it depends on human activity. Therefore, we first map our time series into the frequency domain to identify the

cell, proportionally to the size of the remaining overlapping area as a percentage of the local identity unit.

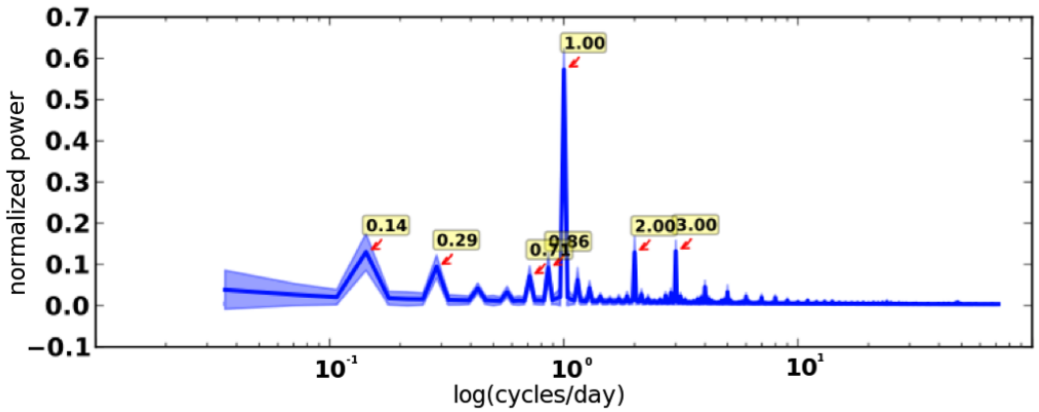


Figure 2.4: Power spectrum of activity aggregated over all grid cells (blue line indicates mean, shaded area ±1 std dev); marks indicate high-amplitude frequencies, e.g. daily (1 *cycles/day*) and weekly (0.14 *cycles/day*.

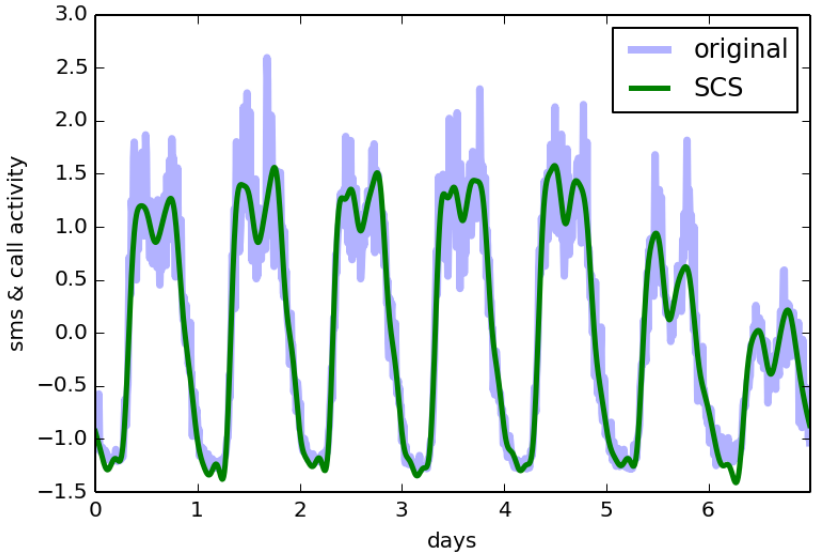
dominant seasonal components. We apply the fast Fourier transform (FFT) to convert $O_i(T)$, for all $i \in S$, from the time domain to the frequency domain. FFT is suitable for our purpose because it is a non-parametric method that extracts periodicity, it is useful for series with no obvious trend and provides a spectrogram that is easily interpretable [29]. Fig. 2.4 shows the power spectrum of all series in the frequency domain. We observe several frequencies with high power (e.g., weekly, daily, and 12-hour cycles). These high-power frequencies dominate the seasonal component of the communication series, and indicate endemic social processes taking place within the city.

For each grid cell $i \in S$, we decompose the original time-series $O_i(T)$, into *seasonal* and *residual* components through the following steps.

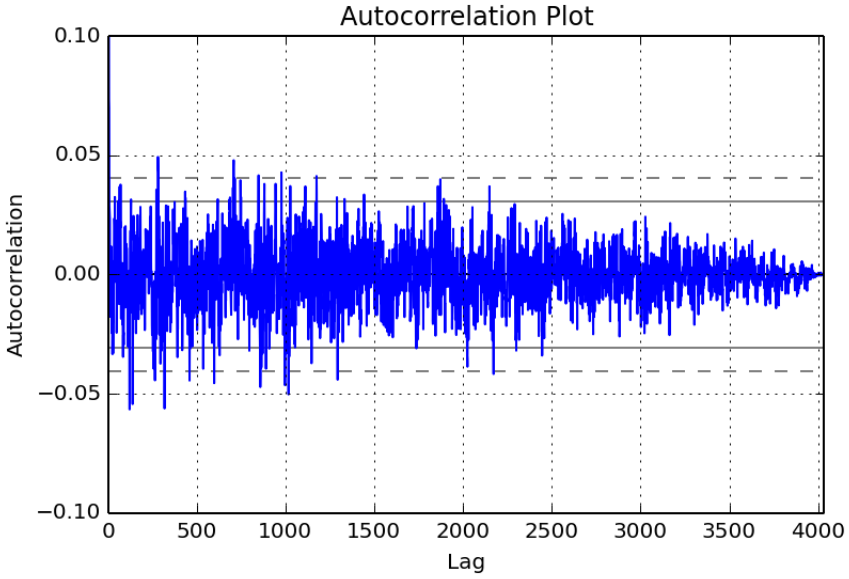
1. We select its k highest-power frequencies.

2. We regenerate the seasonal communication series $SCS_i(T)$ using only the top k frequencies of the cell, where $SCS_i(T) = \{scs_i(t_1), ..., scs_i(t_m)\}$.

3. We obtain the residual communication series $RCS_i(T)$ by subtracting the basic series from the original series:
   $RCS_i(T) = \{o_i(t_1) - scs_i(t_1), ..., o_i(t_m) - scs_i(t_m)\}$.



(a) Original series and derived SCS; SCS contains only systematic information, and is hence smoother.



(b) RCS autocorrelation. Low autocorrelation values verify that systematic components have been removed. Note that the y-axis is zoomed in the interval (-0.1, 0.1).

Figure 2.5: Decomposition of original activity series for grid cell 5071.

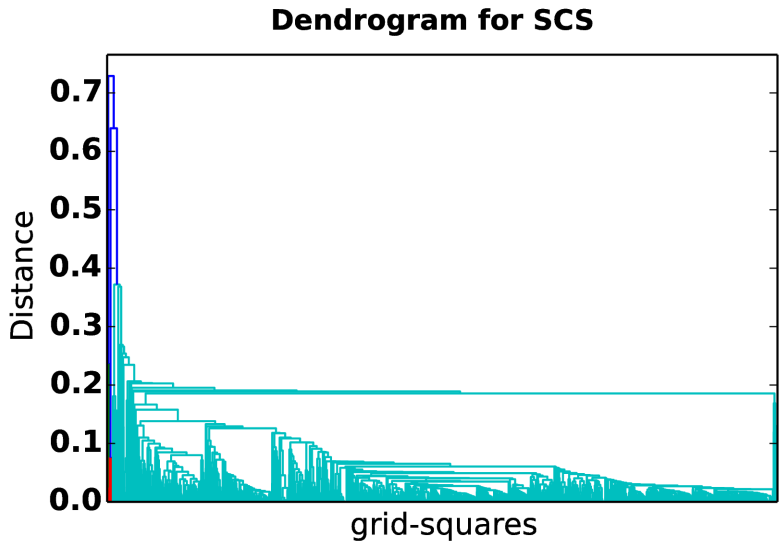
The data analysis in the remainder of the chapter uses $k = 30$; this was selected by finding the smallest k such that the RCS autocorrelation function does not differ significantly from that of a white noise sequence, as shown in Fig 2.5(b). Please note that SCS captures regular

activity (which we use in Section 6.1 for clustering of areas in the city), while RCS carries information about similarity of residual activity between two cells (which we use in Sec. 2.3.6 to study cross-cell interactions.)

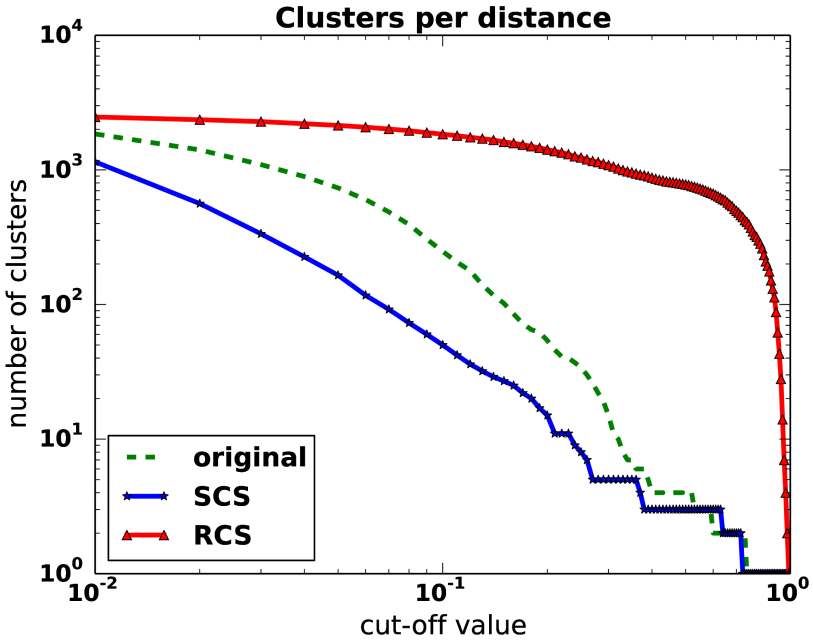
Insights from decomposition and neighborhoods: Here we discuss observations on the correlation of a cell and its neighbors w.r.t. the different time series, namely, original, SCS and RCS. These are important to guide the clustering based on these time series, discussed in the next section.

Fig 2.7(a) evaluates the correlation between neighboring and non-neighboring cells for the original, SCS, and RCS series. In RCS, the correlation between neighbors is much stronger than the correlation between non-neighbors; this is a clear indication that when something occurs in a cell, it often spreads to its neighbors. On the other hand, the correlation between non-neighbors is close to zero, which indicates that perturbations of activity are spatially restricted; this is even more clear when we look at Fig. 2.7(b) that shows how the correlation among neighbors decreases as the size of the neighborhood increases. Also, in Fig. 2.7(c), we see that what happens in a cell at time slot t_0 will affect its neighbors at a later time slot, with the effect dampening over time. This is compatible with an underlying diffusion process.

Fig 2.7(a) also shows that in SCS the correlation between both neighbors and non-neighbors increased when compared to the original series. Moreover, the difference of the correlation between neighbors and non-neighbors decreased. This shows that SCS is dominated by the day-to-day activity patterns, and is stripped from perturbations that are spatially constrained.

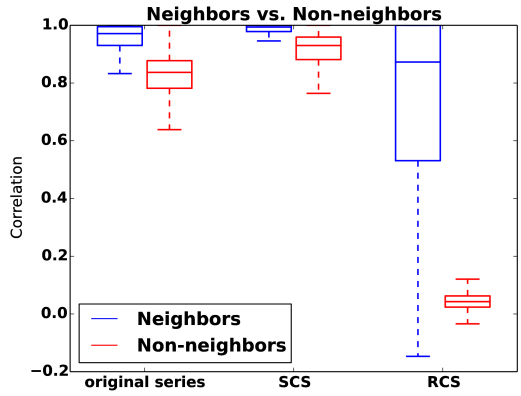


(a) Dendrogram

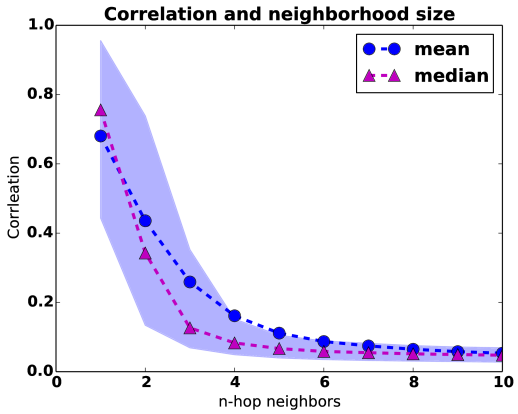


(b) Number of clusters per cut-off distance.

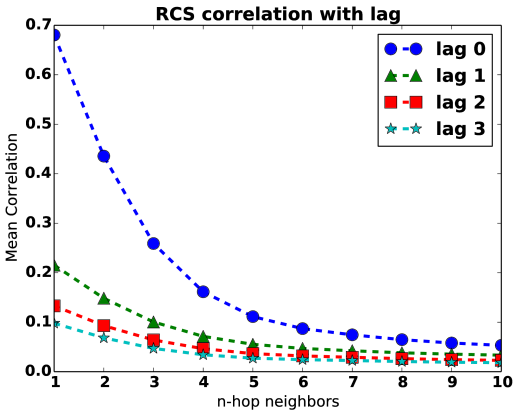
Figure 2.6: Cut-off distances and clusters. (a) shows the dendrogram for clustering the SCS. At distance 0.74 all cells have been merged into one cluster. By looking at the dendrogram we see that most cells in Milan have quite similar traffic, *i.e.* highly correlated activity series, with a few special cells that are different from the rest. Also, we summarize clustering with SCS, original, and RCS in (b), which shows the number of clusters per cut-off distance.



(a) Correlation between neighbors vs. non-neighbors for original, SCS, and RCS.



(b) Correlation for n-hop neighbors for RCS.



(c) Correlation and lag.

Figure 2.7: Correlation and neighborhood. In (a) we observe that in all three cases, the correlation between neighbors is stronger than non-neighbors. However, the difference in the correlation between neighbors and non-neighbors varies. In (b) we observe that as the size of the neighborhood increases the mean correlation decreases – the filled area of the graph represent the region between 25^{th}-percentile and the 75^{th}-percentile. And, in (c) we observe that even when we look at lagged correlation for the RCS, we observe higher correlation, on average, between neighboring cells and a decline in the mean correlation as distance increases. Correlation also decreases when time lag increases.

2.3.5 Clustering

Our goal is to use the result of the decomposition to segment the city into distinct areas, where the members of same area would have similar activity patterns. We hypothesize that if two cells have similar communication patterns then they have similar local ecologies, whereas

if their communications patterns differ then they have different local socio-economic activities; *e.g.* one would expect a university and a stadium to have different activity patterns, since people visit them during different hours.

To achieve our goal, we cluster cells via agglomerative hierarchical clustering. We employ hierarchical clustering because of its generality (requiring no particular assumptions regarding the underlying distance measure) and because it yields a family of solutions (generally expressed as a dendrogram) that contains more information than a single clustering solution. As we show, this information can be exploited to perform both segmentation and anomaly/outlier detection from a single dendrogram. The distance function used in the rest of the chapter is based on the Pearson correlation between activity series. More specifically, for two cells $i, j \in S$, their distance in a given activity series A is:

$$dist(A_i, A_j) = 1 - correlation(A_i, A_j)$$

This distance function takes values in the range [0,2]; when two cells are fully correlated they will have a distance of 0, when they are completely uncorrelated a distance of 1, and when they are inversely correlated a distance of 2. Also, we used the average linkage criterion to build the dendrogram since it had the highest cophenetic coefficient in comparison with other linkage types – the cophenetic coefficient is a measure of how faithfully a dendrogram preserves the pairwise distances.

Fig. 2.6 shows the high-level clustering results. Fig. 2.6(a) shows the dendrogram for the clustering via SCS; by looking at it we see that for small cut-offs, *e.g.* values between 0.02 and 0.08, we get a segmentation of the city into multiple areas, while for large cut-offs, *e.g.* values higher than 0.28, we get a very large cluster and a few small ones – hence, high cut-offs can be used for segmentation and low cut-offs can be used to detect areas with anomalous activity, without requiring additional computation.

Fig. 2.6(b) evaluates the number of clusters generated in the original, SCS, and RCS series for the full range of cut-off distances from 0-1. We observe that the results of clustering with RCS differ significantly from that with SCS and original. More specifically, clustering with RCS yields a large number of clusters (1000s) until cut-off distance $\sim$ 0.90. On the other hand, clustering with SCS yields a low number of clusters for cut-off distance as small as 0.10. We speculate that the reason for this result is due to the degree of correlation between neighbors and non-neighbors (also see figures 2.7(a) and 2.7(b)). Intuitively, SCS provides information on routine activities in the city. Thus, where the analyst's goal is to cluster urban areas based on regular patterns of activity, the SCS should be employed.

2.3.6 Results

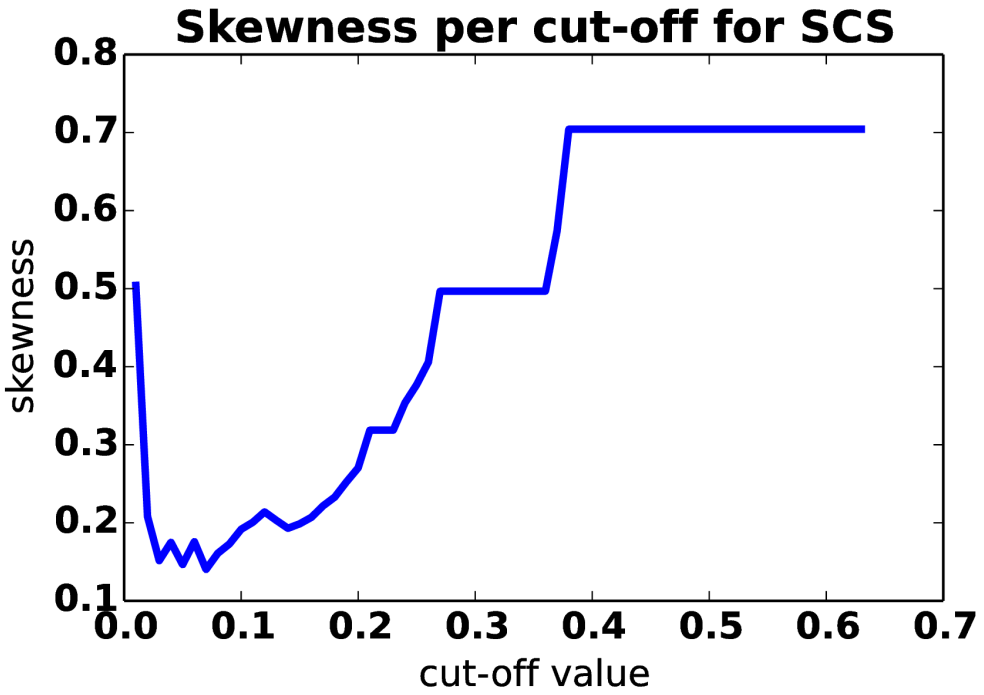


Figure 2.8: Skewness plot for SCS clustering. We use Pearson's second skewness coefficient: (mean-median)/(standard deviation). The minimum skewness is at cut-off = 0.07.

The results presented in this section are limited to the grid cells that overlap with the ground truth; this corresponds to the approximately 3K cells represented in the colored area

of Fig. 2.2.

Clustering based on the SCS

SCS hierarchical clustering requires a choice of where to cut the dendrogram. Examination of the number of clusters by distance, Fig. 2.6(b), shows that this alone does not produce a natural cutting point for the SCS; thus, we also factor in skewness of cluster sizes, defined as (mean-median)/(standard deviation). Fig. 2.8 shows the calculated skewness for the full range of cut-off values in the SCS clustering. The rationale for employing skewness to guide distance selection is as follows. One expects that a segmentation of the urban environment into distinct functional areas, *e.g.*, university areas, residential areas, *etc.*, will produce clusters of relatively comparable size, with correspondingly low skewness. On the other hand, if the distribution is very skewed, then there is a small number of large clusters which contain the vast majority of cells, and various small clusters with unique activity patterns that are quite different from the rest of the city, *e.g.* stadiums. By choosing low versus high-skewness cut-points, one can then choose to break the city into a few large areas of similar activities, or (respectively) detect small, anomalous areas in the city against an "average" background pattern. Both options provide distinct information, and it is not necessary to employ only one; here, we show both cases.

Low-skewness segmentation: In this case we seek to divide the city into comparably sized clusters, and hence choose a cut-point that yields a size distribution with low skewness. Per Fig. 2.8, we obtain this via a distance threshold of 0.07, which is the minimum skewness value in the SCS clustering. As Fig. 2.10 shows, the clusters of Fig. 2.9(a) successfully segment the city by features of the urban environment. For instance, clusters c1 and c5 have a high density of universities, while cluster c3 has a high density of green space; density is the ratio of the number of ground truth elements over cluster size. Thus, we conclude that the clusters indeed reflect regions with distinct socio-economic and environmental characteristics

Category	Entropy for hierarchical SCS	Entropy for [82]
Universities	0.96	0.97
Businesses	0.82	1.33
Green (%)	0.94	1.27
Population	0.97	1.34

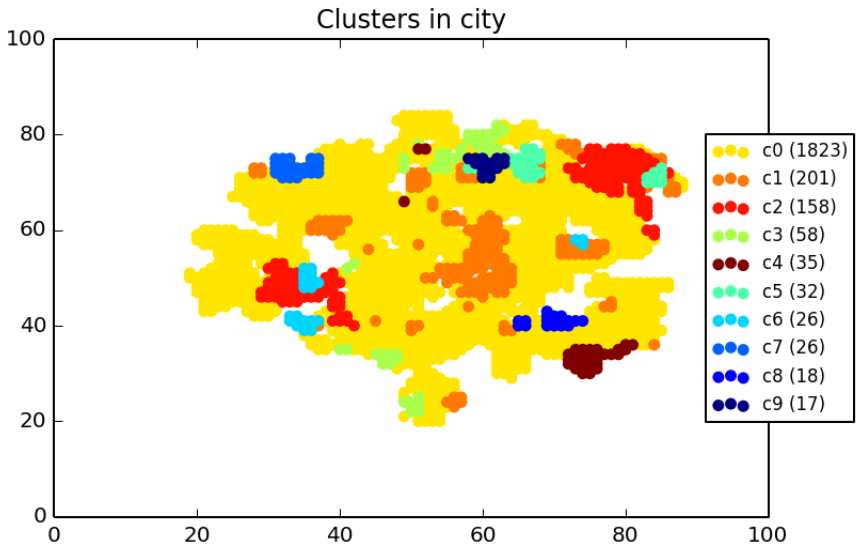
Table 2.2: Segmentation performance via Entropy (lower value is better). Hierarchical SCS clustering produces more functionally distinct clusters for all categories.

as reflected by their differences in SCS.

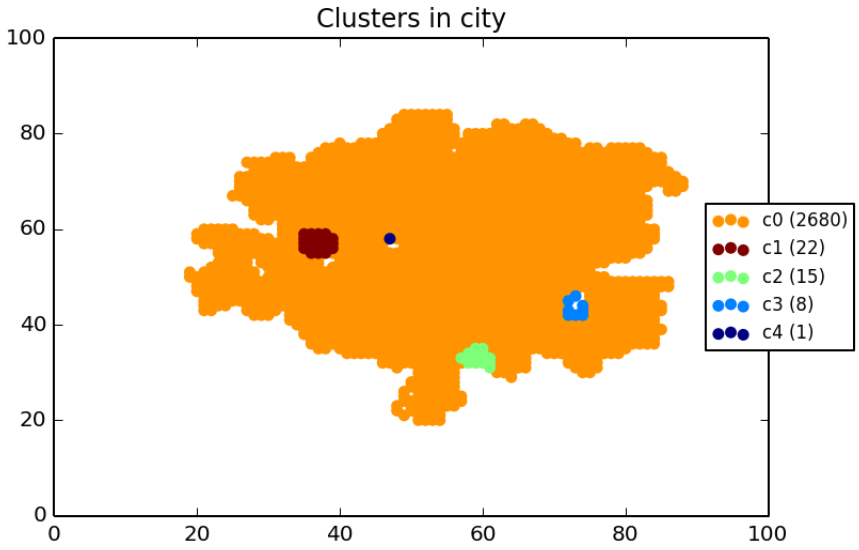
High-skewness anomaly detection: In this case we seek to identify one large cluster reflecting the range of "typical" activity patterns and several small areas of anomalous activity; we thus select a cutoff leading to a skewed distribution e.g. Fig. 2.9(b) with cut-off 0.30 which contains five clusters. In Fig. 2.11 we see the normal activity in the city in cluster c0, while the other clusters represent strongly anomalous traffic. For instance, cluster c1 corresponds to the San Siro stadium, while cluster c3 contains the Otomercato, a wholesale market for fruit and vegetable.

SCS vs. original series: In Fig. 2.12 we show a comparison between the clustering with SCS, and the clustering with the original in regard to coverage for the full range of cut-off distances $0 - 1$. We confirm that clustering with SCS yields better results. This verifies our expectation that removing idiosyncratic variation from the signal clarifies the regular activity within a cell.

SCS vs. Typical Weekday/Weekend : Prior work [82] handles clustering of time series by aggregating cell phone activity in a typical weekday/weekend over all users in an area. Our method fits data better and requires less assumptions compared to that approach. We compare the performance of our method with that of a typical weekday/weekend using the Entropy of the density distribution for a given category, which is a common external clustering evaluation measure [92]. Table 2.2 shows the obtained values of Entropy for our method and the clustering via K-means and typical weekday/weekend from [82]. We observe



(a) Segmenting Milan (cut-off 0.07)



(b) Detecting anomalous areas (cut-off 0.30)

Figure 2.9: Clusters generated by low skewness (left) and high skewness (right) segmentation. When the number of clusters exceeds 10, we show only the top 10 largest clusters.

that our method performs better for all ground truth categories. The intuition behind that is that our way of summarizing the cell phone activity is superior in the sense that the seasonal component of FFT (our SCS) holds more information than the typical weekday/weekend approach, as shown in Fig. 2.13. To quantitatively illustrate that point, we calculate the

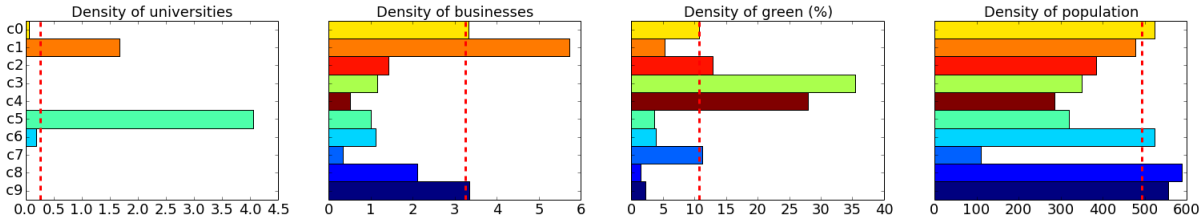
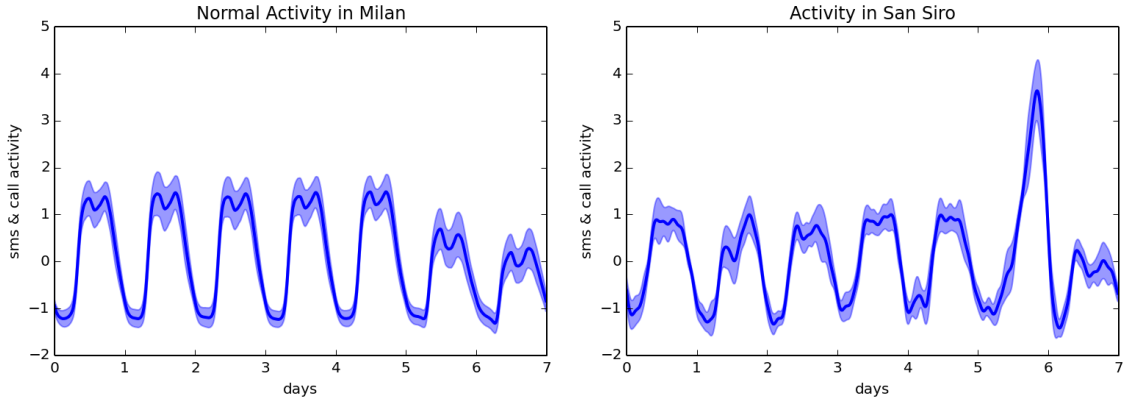
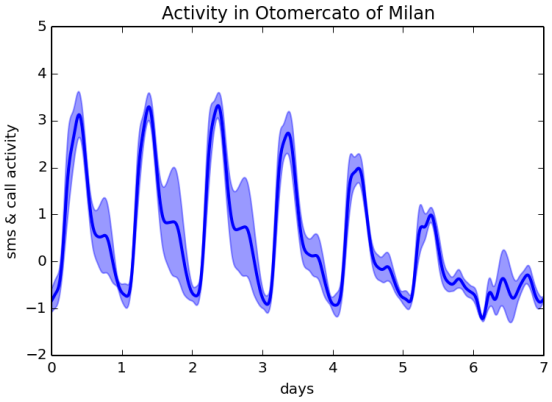


Figure 2.10: Densities per category for top 10 clusters of Fig. 2.9(a); red dashed line indicates the Milan average. SCS clustering separates regions with distinct urban environments (e.g., commercial vs. green space).



(a) Normal area (c0)

(b) San Siro (c1)

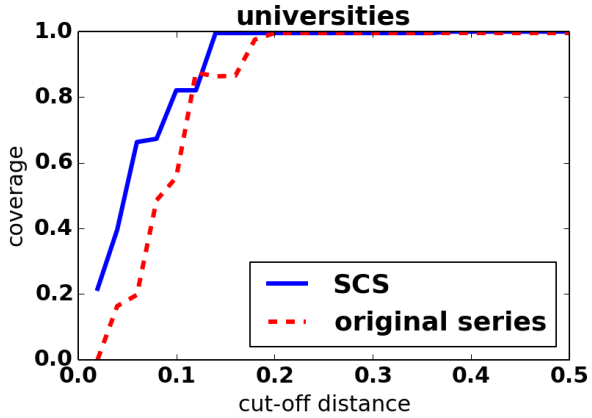


(c) Otomercato (c4)

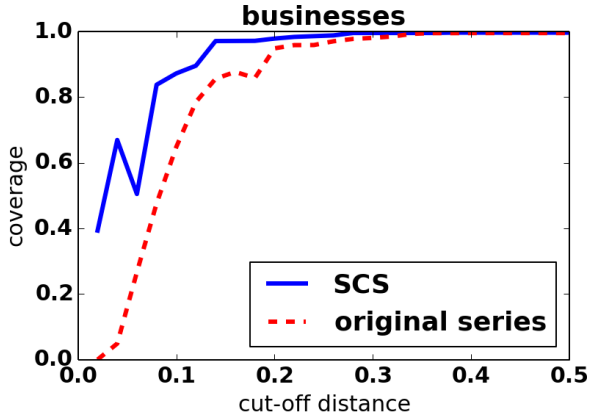
Figure 2.11: Average activity series for the clusters of Fig. 2.16(b); different clusters have very distinct seasonal patterns.

normalized sum of squared errors for each cell i, for both methods as follows:

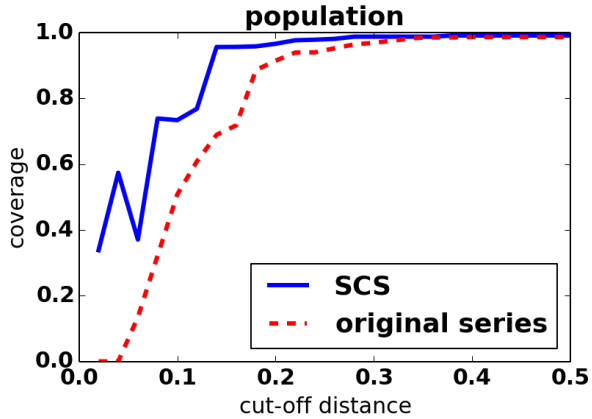
$$e_i = \frac{\sum_{t=1}^{t=T} (O_i(t) - B_i(t))^2}{T}$$



(a) Universities



(b) Businesses



(c) Population

Figure 2.12: Coverage for SCS clusters vs. original series clustering. Coverage is defined as the percentage of the ground-truth elements "covered" by the clusters with higher than mean concentration of the element type. SCS clustering shows stronger segmentation (higher coverage) for all cut-off values.

In our method $B_i(t) = SCS_i(t)$, whereas in the method of [82] $B_i(t)$ is obtained by using the typical weekday five times and the typical weekend twice to create a typical week, and repeating the typical week four times to create a time series of length T, which corresponds to a month. Fig. 2.14 shows the cumulative distribution of the errors over all cells. SCS has smaller errors which indicates that it holds more information.

Our method requires less assumptions regarding the nature of the data. It works well in a different culture, e.g. some countries, such as Egypt, designate Friday as a weekend day, while

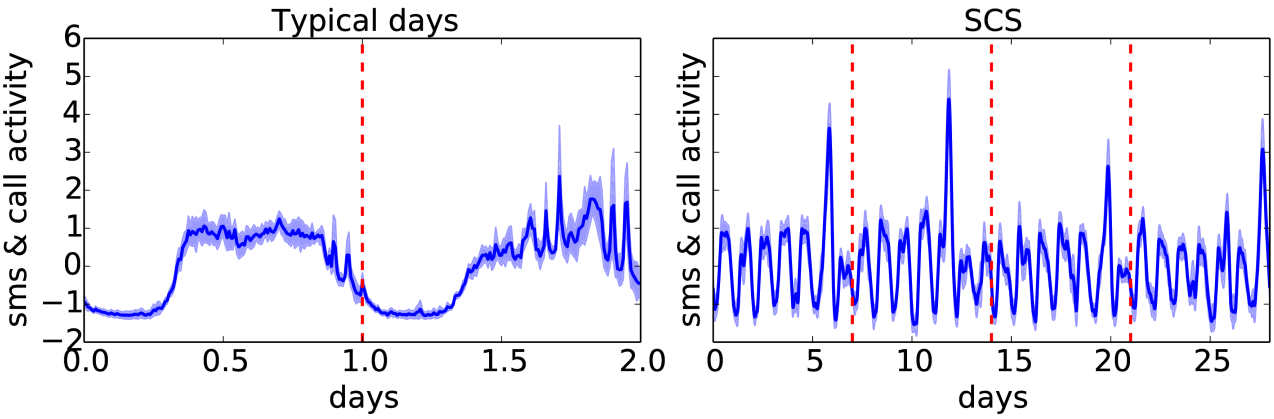


Figure 2.13: The left figure shows how the typical weekday/weekend approach summarizes the cell phone activity series for the San Siro area. Notice that the high peaks are lost when traffic is aggregated. The right figure shows the SCS traffic in the San Siro area for one month (our method).

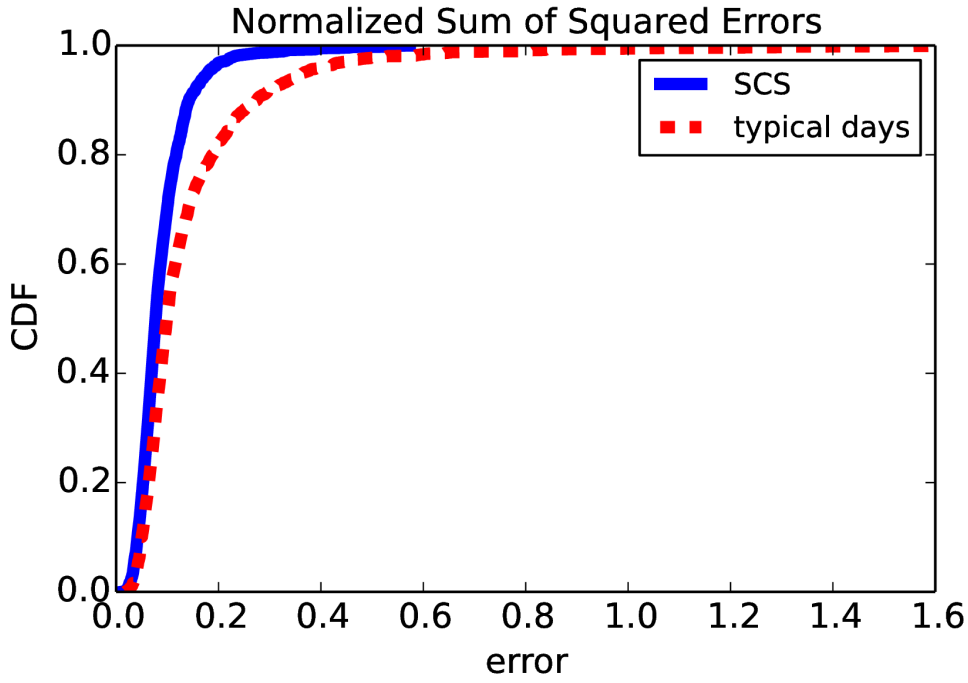


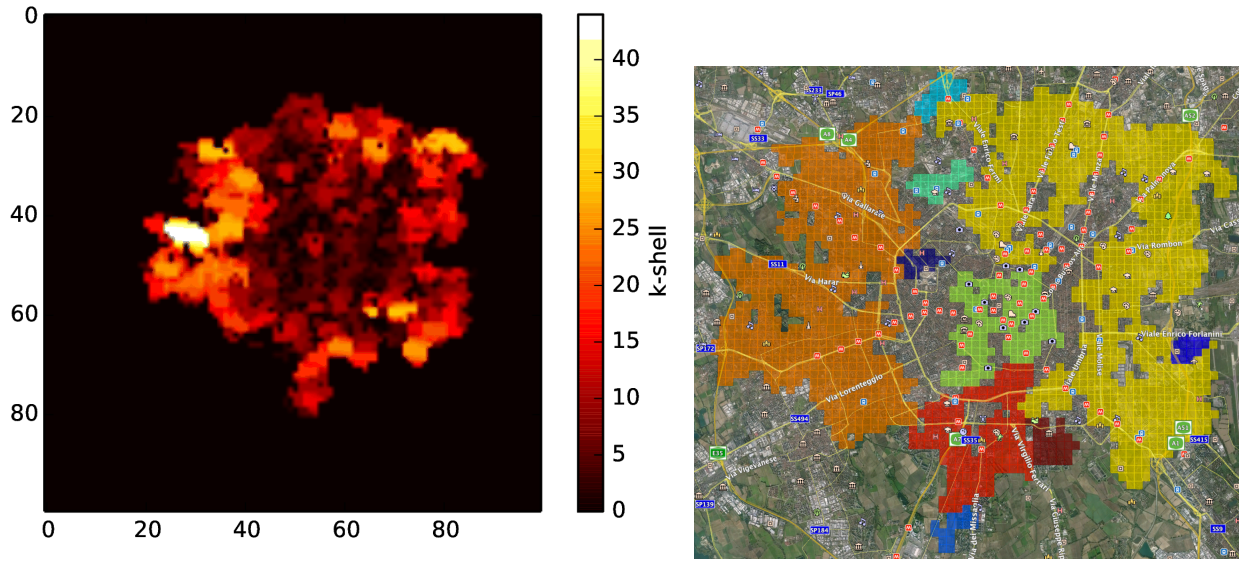
Figure 2.14: The figure shows how much information is lost from the original series, when we use SCS and when we use typical days. Despite the fact that SCS is created using the top-30 frequencies, while the typical days series is created using a vector of 288 values – 144 values for weekdays, and 144 for weekends – the SCS can reconstruct the original series much better.

others have Thursday-Friday weekends. To apply the typical weekday and weekend division requires knowledge of the culture, while our approach does not require that. Additionally,

the typical weekday approach does not capture mid-week variations that might appear in certain areas.

What we learn from RCS

RCS reflects the response to perturbations rather than regular behavior. Therefore, we treat them differently: we use cross-correlation between the RCS time series of cells, and we investigate how the RCS of one cell affects its neighbors and how it correlates with cell-to-cell communications.



(a) Heat-map showing the k-shell score of each grid cell. We observe that cells in the center of the city have lower k-shell scores in comparison with the periphery. This shows that, for this particular directed graph, cells in the periphery are more socially well-connected than cells in the center.

(b) Largest Strongest Connected Components

Figure 2.15: k-shell heat-map, and 10 largest strongly connected components for the directed graph of RCS cross-correlations with $lag = 1$.

Using RCS to study cross-cell interactions

A more powerful use of the RCS is to examine interactions between cells. Specifically, examination of lagged cross–correlations in the RCS for two cells shows how cells affect each other: i.e., for two cross-correlated cells, a perturbation in one cell at time t will be associated with a change in the other cell at a later point in time. We denote the cross-correlation "distance" for cells B_i, B_j at a given lag by:

$$dist'(Bi, Bj, lag) = 1 - correlation(Bi, Bj, lag)$$

Note that, unlike a true distance, $dist'$ is not symmetric, i.e.
$dist'(Bi, Bj, lag) \neq dist'(Bj, Bi, lag)$. We thus build a directed graph where the nodes are the cells and cell i is adjacent to cell j at a given lag iff

$$dist'(Bi, Bj, lag) < thresh$$

where $thresh$ is an analyst-selected threshold that filters out weak time-lagged correlations. We set $lag = 1$ which corresponds to a 10-minute difference between two cells; it is the most fine-grained unit we can get from our data. We found experimentally that a threshold 5 standard deviations away from the mean yields good results. Given the cross-correlation digraph, we may examine the strongly connected components of the graph to find areas that are subject to mutual social influence (e.g., there are paths by which events in any cell can affect communication activity in any other cell at a later time).

What we learn from the strong connected components of the graph: In Fig. 2.15(b) we show the top 10 strongest connected components of the graph, as described in the previous paragraph. We observe a different, but interesting, segmentation of the city from that obtained by SCS, with a clear structure becoming apparent. The center of the city is a connected

component (green color), completely separate from the rest. This means that perturbations occurring inside a cell on the center, will most likely propagate to other cells in the center. Another example is the dark blue connected component on the center-right side which corresponds to the Milan Linate airport. More generally, the large clusters identified in Fig. 2.15(b) reflect *socially connected regions* of the city, with events in any given cell tending to reverberate within its regional cluster (but not to propagate beyond).

What we learn from the k-shell decomposition: k-shell decomposition is a well-known technique in graph theory for identifying regions of high local cohesion [74] and has been used as a visualization tool for studying networks such as the Internet [26]. It involves identifying maximal sets of nodes with at least k neighbors who are also in the set (k-cores) and then identifying the highest-number to which each core belongs. In this article, we apply the k-shell decomposition on the RCS-based digraph, identifying the cells that are more/less cohesively connected to their neighbors; spatial regions with high values of k are strongly interactive (in the sense that perturbations in one location can propagate to other locations in the region through multiple, redundant correlation paths). In Fig. 2.15 we observe that cells in the center of the city have lower k-shell scores in comparison with the periphery. This shows that, within Milan, there are several spatially peripheral regions with high local connectivity, while cells near the city and along major arterials tend to be either isolated (with respect to propagation of shocks) or connected via locally tree-like structures.

Correlation of SCS and RCS with Milan Cell-to-Cell Communication

QAP test results	SCS	RCS
Correlation	**0.05**	**0.27**
Min random	-0.018	-0.005
Mean random	0	0
Max random	0.011	0.004

Table 2.3: Correlation of SCS and RCS inter-cell correlations with Milan Cell-to-Cell data set, with QAP test replications ($n = 100$). Correlations are significant at $p < 0.01$.

Finally, we compared the Milan Cell-to-Cell call volume data set with the SCS and RCS cell-to-cell cross-correlations, assessing the resulting relationship using the Quadratic Assignment Procedure (QAP)[57, 23]. QAP is a technique for testing an observed bimatrix statistic (here, matrix correlation) against a null hypothesis of no association, while controlling for the underlying structure of the matrices being compared; the technique is a form of matrix permutation test, in which the distribution of bimatrix statistics obtained under row-column permutation of the input matrices is used to form a null distribution.

We transformed the Milan Cell-to-Cell data set into a NxN matrix, where N is the number of cells, and we denote it as MM. An entry $MM_{i,j}$ corresponds to the symmetrized communication strength between cells i, j in the Cell-to-Cell data set. Similarly, we created two additional NxN matrices: 1) matrix RM, with $RM_{i,j} = RCS\_Correlation(i, j)$, corresponding to the residual correlation for cells i, j, and 2) matrix BM, with $BM_{i,j} = SCS\_Correlation(i, j)$, corresponding to the SCS correlation for cells i, j.

As we can see from Table 2.3, there is a significant correlation between the Cell-to-Cell data set both for SCS and RCS (p <0.01 in both cases), but for the residual series it is almost six times stronger. Thus, we see that the cross-correlations between cells are associated with direct contact between persons in the respective areas, and this is a substantially stronger effect than the baseline similarity in calling pattern within each cell. This further validates our above intuition that the RCS cross-correlations provide information on social interaction across areas within the city.

2.3.7 Summary

In this work, we studied the decomposition of cell phone activity series, via FFT, into two series: 1) the seasonal communication series (SCS) produced from high-amplitude frequencies, and 2) the residual communication series (RCS) produced after subtracting SCS from

the original series. As shown, the SCS can be used to characterize typical patterns of socio-economic activity within an area, while the RCS can be used to capture both irregularities due to novel events and the influence of one area on another. For the first part, we perform an external evaluation of the produced clusters using a ground truth data set that we gathered from the municipality of Milan. Our SCS clustering, produces clusters of areas with similar characteristics as shown in ground truth data. RCS allows to identify regions such that disruptions in one area propagate to the other, and regions that are in direct communicative contact. The RCS and SCS thus provide distinct probes into the structure and dynamics of the urban environment, both of which can be obtained from the same underlying data.

Our techniques are applicable to other geo-social activity data sets,e.g. Twitter and Foursquare, and can be used to reveal patterns of how areas related to each other; in future work we plan to apply our techniques to cell phone activity data from other cities, as well as other type of geo-social activity data. These findings will provide the network operator with information that can improve planning, operations and anomaly detection.

2.4 Cell-to-Cell Prediction

2.4.1 Introduction

Cellular penetration has increased dramatically over the past decades and the number of unique mobile subscribers is estimated around 3.4 billion users [83]. At the same time there is an even greater growth in demand for wireless access bandwidth worldwide, due to the fast adoption of smartphones. The traffic volume generated by mobile phones will increase approximately by 8 times in 2020 (30.6 exabytes/month) compared to 2015 (3.7 exabytes/month), according to traffic trends forecasts [36].

To address this demand, mobile phone providers and the 3GPP are currently working on improvements to the current 4G standards as well as on future 5G networks [75]. More specifically, a mixture of macro-cells and small cells (i.e. heterogeneous nets) is currently being considered for increasing 4G capacity. Small cells are feasible by utilizing femtocells [19], i.e. low power base stations with limited range, typically designed for use in a home or business, covering the spectrum holes of the larger cells. This shift (towards smaller cells and denser networks) is closely connected with a shift towards virtualization of computational resources, that follows software defined networking (SDN) and self-organizing networks (SON) principles[3].

However, a dense infrastructure is complicated and costly to maintain. The energy consumption of base stations is one of the largest costs for mobile phone providers [41]. Hence, they try to make their infrastructure more energy efficient, e.g. by switching femtocells on or off or by lowering the transmission power. The aforementioned technologies will incorporate the necessary logic for smart decisions and network configuration based on network events, to automate resource allocation. For instance, [58] describes the architecture of SDN - SON where traffic prediction algorithms will be utilized in the control plane for the assignment of virtualized radio resources. Thus, being able to predict cellular traffic patterns city-wide, can inform and enable network provisioning and control.

Accurate prediction of mobile phone traffic in a city is necessary for enabling urban planning and a number of smart city applications. In this thesis, we develop a building block in that direction: machine learning techniques for cell-to-cell mobile traffic prediction based on past cellular records,.

We analyzed a data set provided by Telecom Italia as part of the Big Data Challenge [10] competition, and more specifically the Milan Cell-to-Cell data from Tab. 2.1, which describes

[3]SON is an example of this trend [62, 17, 60]. SON is a software module responsible for planning, configuring, and managing the cellular infrastructure. For example, SON could use cognitive radio techniques to exploit under-utilized spectrum in the unlicensed bands, during high load hours [75].

the intra-city activity in Milan: time series $A_{i,j}(t)$ describe the communication volume between two areas of the city, i and j, for $t = 1..N$. We formulate the traffic prediction problem as a classification problem. Based on past activity our goal is to predict whether two cells will talk to each other during at time t, i.e., $A_{i,j}(t) > 0$. First, we visualize important aspects of our data using SVD to better understand the data. We use the insights gained from the data analysis for feature selection; for example, we found that neighbors tend to talk more to each other and are more correlated. Second, we used decision tree classifiers and random forest in order to do prediction. We were able to achieve accuracy 85%, which outperforms the naive max-class predictor (80%) that predicts the most frequent class. A key insight and challenge was the sparsity of the dataset: most cell pairs have zero communication activity with each other. This leads to high skewness of our classes and low recall rate (lower than 40%). Since, F_p (false positive) and F_n (false negative) errors don't have the same cost for providers, we show how to improve recall up to 94%, by giving higher weight on F_p.

2.4.2 Related Work

Related work can be roughly classified in three categories: (a) traffic volume prediction from a mobile telephony cell tower, (b) link prediction in telecommunication or or social networks and (c) analysis and assessment of network operators' data sets which reveals the spatio-temporal characteristics and the dynamics of the cellular network infrastructure.

In traffic volume prediction, the goal is to forecast the volume (voice or data) generated by a specific base station (i.e. cell tower) for a future time window, given historic traffic traces. Methodologies include, but not limited to, moving average [41], Holt-Winters's exponential smoothing [41], [85], [65], hybrid prediction models [41], temporal compressive sensing [58] and Kalman filtering [41]. For instance, work in [58] utilizes entropy for assessing the predictability of the traffic and quantify what time window (temporal dimension) and how

many adjacent cells (spatial dimension) would actually help. Furthermore, [41] proposes a framework for optimizing power consumption by switching off a portion of the base stations in low network traffic condition. Interestingly, [41] distinguishes the base stations between the typical traffic profiles and the opportunistic profiles (e.g. stadiums where the traffic is present only in weekends), which was a key observation made also by our prior work [32].

However, the traffic volume prediction problem differs significantly in 4G and 5G due to the small cells deployment [27]. A femtocell covers a much smaller area with less users, therefore, bursty traffic is more likely rather than a periodic volume activity which is usually generated by a macro-cell. More interestingly, burstiness and sparsity of the traffic were observed in our data set analysis. Thus, [27] proposes a solution which combines Gaussian processes (GPs) and kernel based methods for the periodic component and tolerance intervals for the bursty component. The data used are a combination of synthetic and real data sets. In contrast, our work studies a real world data set and we predict *if* a cell i communicate with a cell j, i.e. a binary classification problem considering directed communication, which has not been studied by any of the previous works.

Link prediction in networks (social networks, IP subnetworks, mobile phones etc) is also related to our problem. The goal is to predict if two nodes of the network (e.g. two persons in an social network or two mobiles) will form a link and communicate at time t. For instance, work in [81] tries to predict a network attack, given historic data and by considering properties that network attackers and regular users share. The authors use the recommendation systems framework and utilize SVD for principal component analysis, an idea explored in this thesis as well. Work in [59] considers the problem of link prediction in time t for a data set containing phone calls between users. It investigates several factors such as the class imbalance problem, the sparsity of the links, time and statistical features , the strong neighborhoods and other topological features of the phone calls graph. Then, it assesses several supervised learning approaches and proposes a novel flow-based predictor.

Analysis of network operators' data sets, such as Call Detail Records (CDRs), have also been studied [73], [72]. The casual influence from a base station to neighboring base stations load is studied in [73] to assess Granger Causality for traffic prediction. In [73], the time granularity for traffic aggregation is studied showing higher cross correlation between pairs of base stations for time interval of one hour vs 10min intervals. Our prior work in [32] also looked at the data set from the city of Milan and used the aggregate activity per cell as a signature of human activity in that cell, in order to cluster similar areas of the city together for urban ecology. In contrast, this thesis (i) studies not a single cell time series but cell-to-cell communication series and (ii) its goal is to predict future based on past activity.

In summary, this thesis focuses on traffic prediction between two different areas of the city/cellular network and not a call prediction between two independent users, and has the following main differences compared to prior work. First, we consider aggregated CDRs in the spatial dimension (i.e., total volume of calls between all users in the two cells), while [59] considers phone calls between individual users and do not not consider the problem from the perspective of cellular providers: false negatives can be really costly. Last but not least, with all the concerns regarding privacy [91], aggregated CDRs are more likely to be available from the providers.

2.4.3 Formulation

Let S be the set of cells, and let T be the time axis as a set of timestamps $T = \{t_1, t_2, ..., t_m\}$. We denote as $A_{i,j}(t)$ the volume of mobile activity from cell i to cell j, where $t \in T$, and $i, j \in S$.

We formulate the traffic prediction problem as a *binary* classification problem. Given activity series $A_{i,j}(t)$, which shows the activity from i to j at time t, we build a set of features using the past ($< t$) records, and our goal is to predict if $A_{i,j}(t) > 0$. In other words, our goal is

to predict if cells i and j communicate at time t (class 1), or not (class 0).

Finally, we partition time T into two subsets: (i) a training set called T_{train} and (ii) a testing set called T_{test}. This is done via a random 70/30 split of the data, where 70% of the data is used for training and the remaining 30% for testing.

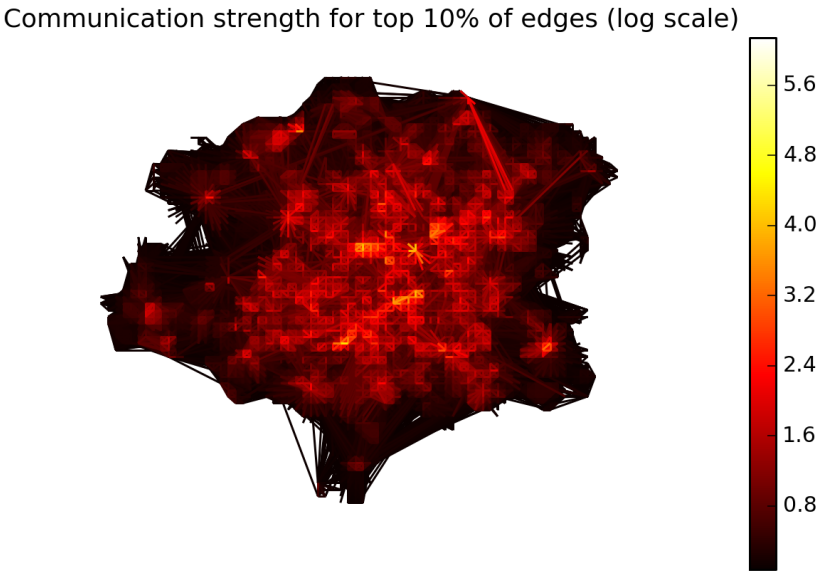


Figure 2.16: The figure shows the communication strength between cells. The communication strength have been calculated by aggregating the interaction during the 1^{st} of Nov. 2013. We can observe that there is strong communications between neighboring cells. For clarity we show only the top 10% of the edges.

2.4.4 Data and Key Observations

In this section, we focused on the Milan Cell-to-Cell data from Tab. 2.1. The data set contains information regarding the *directional interaction strength* (as per terminology in [32]) between two cells, based on the calls exchanged between them. Each activity record consists of the following fields: ID of cell i where call was initiated from, ID of cell j where the call was made to, time slot, and value of directional strength from i to j. Fig. 2.16 visualizes the 10%

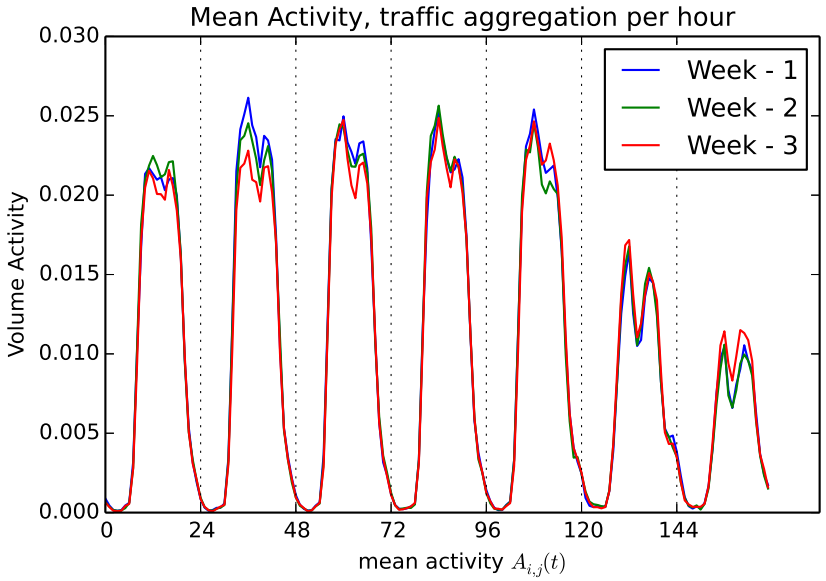


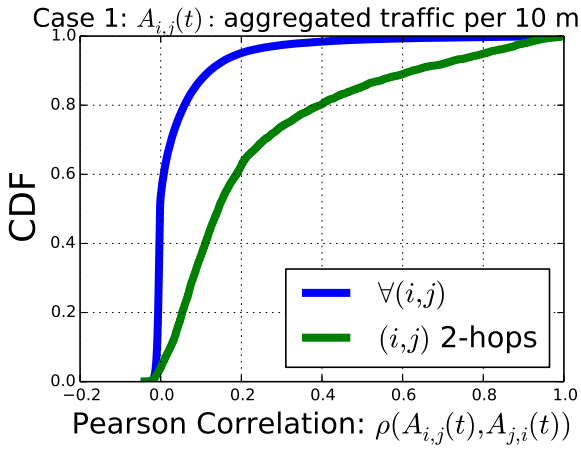
Figure 2.17: Average activity $A_{i,j}(t)$. We see that when mobile phone activity, when averaged across all cell-to-cell traffic, is predictable and follows expected daily and weekly patterns. Also, these patterns are similar across various weeks.

strongest connections.[4]

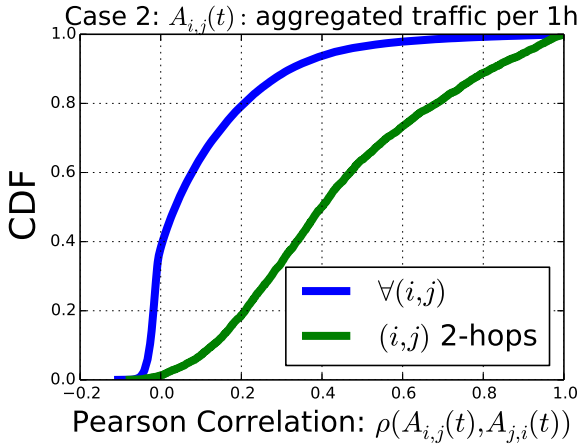
Aggregation of Traffic per Hour

The initial data consist of 10-minute traffic reports. However, we aggregated traffic per 1 hour because (1) traffic in such short intervals is very dynamic and fluctuates heavily, (2) allocation of resources in the cellular infrastructure (e.g. by SON) is not an easy task and planning of resource allocation in 10-min. intervals may lead to unstable networks or excessive overhead, (3) 95% of the activity is zero in such short time intervals, making traffic very sparse and (4) work in [73] faced the same dilemma regarding the time granularity for traffic aggregation (10-min. vs 1 hour) and concluded in 1-hour aggregation since it had higher cross correlation between pairs of base stations.

[4]We focus on the prediction of voice traffic since dropped or bad quality voice calls are noticed immediately as a"poor service" by the customers. However, the methodology should apply to any data set of cell-to-cell communication activity.



(a) 10-minute time interval



(b) 1-hour time interval

Figure 2.18: The above figures show the distribution of activity correlations for all pairs of cells ($\forall(i,j)$), as well as pairs that are within a maximum distance of 2-hops; the correlation for a pair of cells, i and j, is calculate by this formula: $\rho\left(A_{i,j}(t), A_{j,i}(t)\right)$, where ρ denotes the Pearson correlation. We observe that there is much higher correlation when i and j are neighbors. Moreover, this picture shows that by aggregating cell phone activity into hourly time reports then traffic becomes more structured ($A_{i,j}(t)$ and $A_{j,i}(t)$ are more correlated).

Traffic aggregation leads to more predictable series. As you can you see from Fig. 2.18(a), in the case of 10-minute time intervals, traffic between pairs of cells seems to be random and uncorrelated; 60% of the random pairs have zero correlation and more than 95% of them have a correlation lower than 0.2. However, when we aggregate traffic into one-hour time intervals, the time series become more similar and correlated (See Fig. 2.18(b)).

Challenges and Key Insights

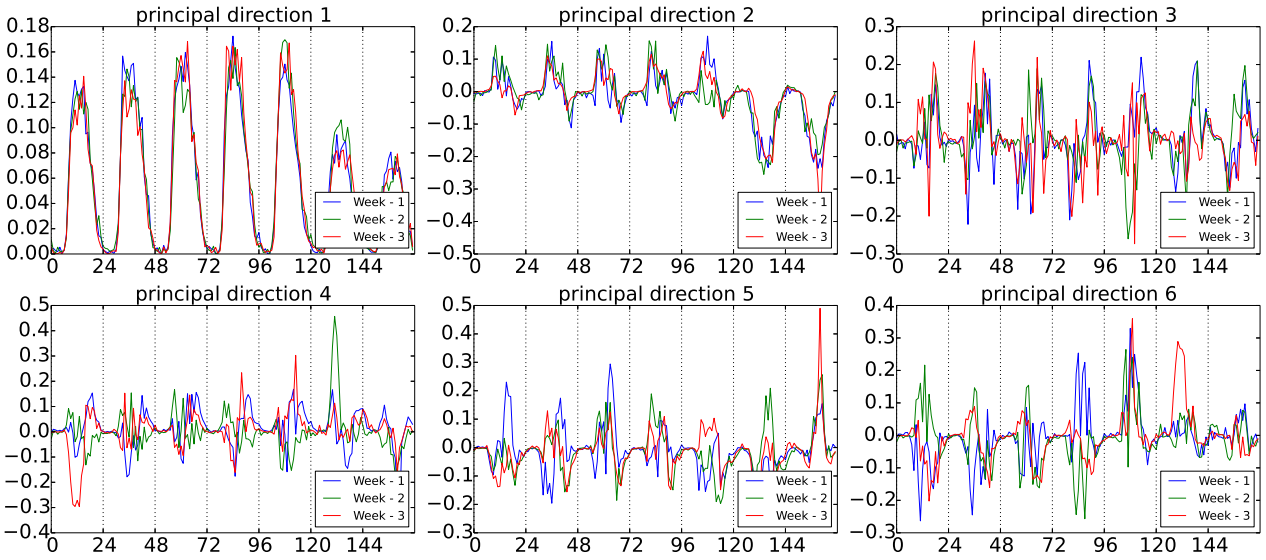
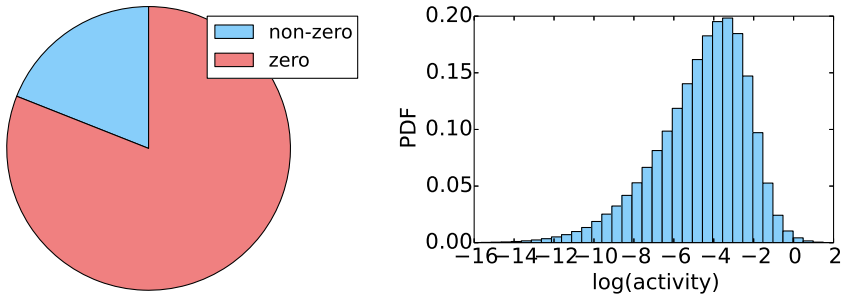


Figure 2.19: SVD of $A_{(i,j)(t)}$ for 1week for aggregated data. Top-6 Principal Components.



(a) sparsity (b) non-zero distribution

Figure 2.20: Distribution of communication values. These figures describe a zero-inflated and skewed distribution.

In this section we present some of the intuitions that we obtained from our exploratory analysis, and we highlight the challenges for the traffic prediction.

(1) *Zero–Inflated Distribution:* One of the main challenges with cell-to-cell communication data is their sparsity. The distribution of the cell phone activity is a zero–inflated distribution – almost 80% the activity is zero (See Fig. 2.20) – while the remaining non-zero activity follows a skewed distribution. This makes prediction very hard since we cannot fit traditional

time series models. Also, on aggregate mobile activity exhibits well-understood seasonal patterns and is easy to predict (See Fig. 2.17), but cell-to-cell traffic is dynamic, it fluctuates and prediction is hard.

(2) *Strong communication between Neighboring Cells:* Traffic between neighboring cells is much stronger (see Fig. 2.16), in comparison to the rest of the city. Also, traffic between neighboring cells is more structured, e.g. $A_{i,j}(t)$ and $A_{j,i}(t)$ are more likely to be correlated when cells i and j are within a 2-hop distance (See Fig. 2.18(b)).

(3) *We observe seasonal patterns in data:* In order to get a better understanding of the data, we decompose the activity series into their first six principal components, which we achieve via singular value decomposition (SVD). Fig. 2.19 demonstrates the components of the traffic from three different weeks. We observe that the first two principal components are structured and tend to be similar across weeks, but the remaining principal components look more spiky and dissimilar across weeks. For example, the second principal component shows the areas that exchange traffic *only* on weekdays and that do not communicate on weekend (observe the negative direction in weekends). This happens for example in universities or in business areas. In addition, there are spikes in the traffic around noon (5-th principal component) which models another "direction" of cellular traffic. This is also demonstrated by Fig. 2.21, where we use the principal components for a week (training week) as a model for another (testing week). The parameter k (number of principal components used) denotes the complexity of the model. We observe that as the complexity increases the model is a better fit for the training week – the mean squared error (MSE) decreases. However, MSE for the testing week MSE decreases until $k = 2$, and the it start increasing. This shows that we cannot expect much gain in prediction from the seasonal patterns of our data.

(4) *Unexpected events affect communication:* Finally, traffic can be affected by unpredictable event. For example, in Fig 2.19, for the second principal component we observed a significant difference at the traffic level for the Friday ($t = 96 \cdots 120$) of the 1^{st} and the 2^{nd} week. This

principal component encapsulates the traffic during the week days as we discussed. The traffic in the 2nd week is significant lower *only* for Friday. This day was the 15-th of November of 2013. We were intrigued from this difference and we searched for potential causes. After a short search in Google, we found that the 15-th of November was the first day of the big social protests and strikes in Italy in 2013 [9]. Apart from the fact that mobile traffic can be affected by unexpected events, this also shows that cell phone activity series can enable many other types of Smart City application, i.e. they can be used to reveal abnormal activities in a city.

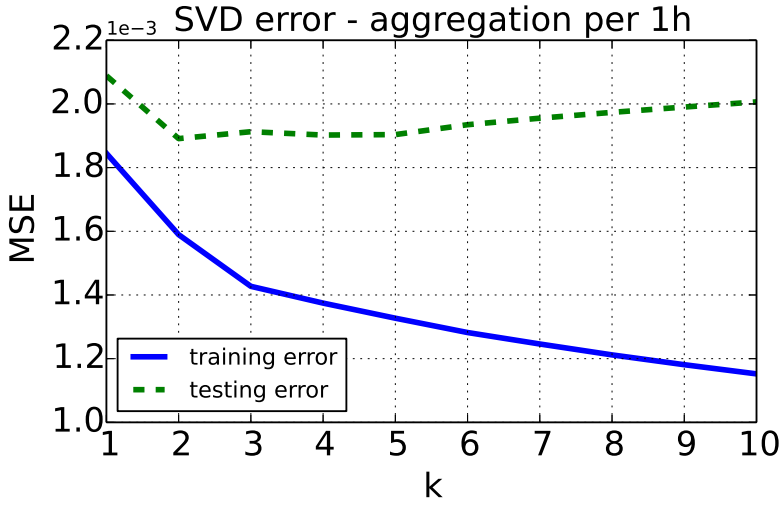


Figure 2.21: Error for $A_{ij}(t)$ reconstruction by k principal components of the data utilizing SVD (aggregation per hour). For this case, the utilization of the first principal components can improve the error on the testing data.

2.4.5 Methodology

Since we are dealing with a zero-inflated distribution – and our data are skewed towards zero – it is difficult to apply classical time series prediction methodologies, such as ARIMA models. Instead we will treat the prediction task as a classification problem; we will generate a set of features for each prediction we want to make, and we will apply standard but powerful classifiers. More specifically we will use a Random Forest Classifier. Next, we describe the

features we used for the prediction. This decision was inspired from key insights (1) and (4) from the previous section.

Feature Selection

For each communication series from cell i to cell j we use the following features:

Static features: We denote as static features, those features that are constant across weeks. These are:

- Geographic distance between cell i and cell j. This was inspired by our earlier analysis that show that neighboring cells tend to talk more.
- Hour of the day. Human activity and communication is heavily influenced by the hour of the day.
- Day of the week. Human activity and communication may change depending on the day of the week (e.g. Monday vs. Saturday).

The geographic distance feature was inspired from key insights (2) of the previous section, while the other two static features were inspired by key insight (3).

Dynamic features: These are features that change from one week – or even day – to the other.

- Past traffic (3 previous hours) of $A_{i,j}$. For example, for target value $A_{i,j}(t)$, the features are $A_{i,j}(t-1), A_{i,j}(t-2), A_{i,j}(t-3)$.
- Past traffic (3 previous hours) of reverse series, i.e. from cell j to cell i. E.g. for target value $A_{i,j}(t)$, the features are $A_{j,i}(t-1), A_{j,i}(t-2), A_{j,i}(t-3)$.

- Average traffic of neighbors (3 previous hours). For target value $A_{i,j}(t)$:
  - $E[A_{i,k}(t-1)], E[A_{i,k}(t-2)], E[A_{i,k}(t-3)]$, where k is a neighbor of j.
  - $E[A_{k,j}(t-1)], E[A_{k,j}(t-2)], E[A_{k,j}(t-3)]$, where k is a neighbor of i.
- Standard deviation of neighboring traffic (3 previous hours). For target value $A_{i,j}(t)$:
  - $\sqrt{Var[A_{i,k}(t-1)]}, \sqrt{Var[A_{i,k}(t-2)]},$
    $\sqrt{Var[A_{i,k}(t-3)]}$, where $k \in$ neighborhood(j).
  - $\sqrt{Var[A_{k,j}(t-1)]}, \sqrt{Var[A_{k,j}(t-2)]},$
    $\sqrt{Var[A_{k,j}(t-3)]}$, where $k \in$ neighborhood(i).

The latest set of features (average traffic of neighbors, and standard deviation of neighboring traffic) were inspired by our analysis that showed that there is higher correlation among 2-hop neighbors (key insight (2) from the previous section).

2.4.6 Results

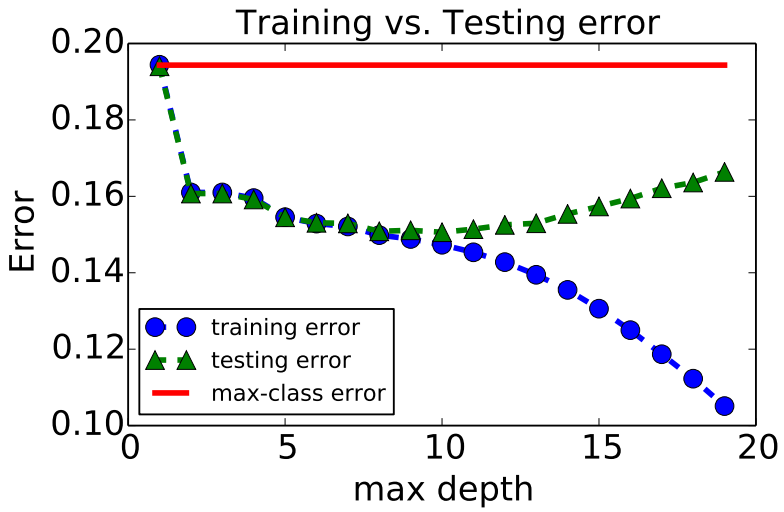
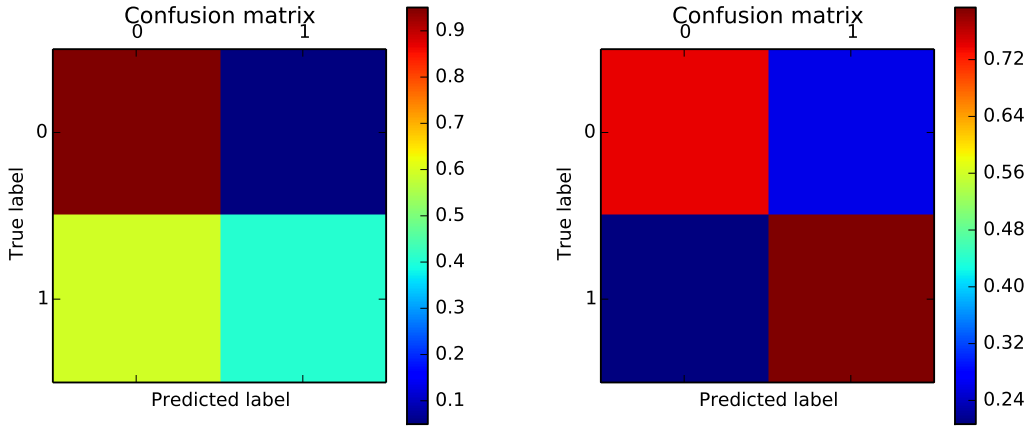


Figure 2.22: Error on training vs. testing data for a decision tree.

We elected to use a tree classifier, since tree classifiers are powerful tools that can learn complex functions. We tuned the classifier's parameters and made sure that we don't overfit

Scores	Max-class *predictor*	Decision Tree	Random Forest
Accuracy	80%	84%	85%
Precision	-	72%	68%
Recall	-	37%	40 %
F1-score	-	48%	51%

Table 2.4: Scores for max-class predictor, decision tree classifier and random forest.



(a) Classes equally important

(b) Class 1 more important

Figure 2.23: Normalized Confusion Matrix. Each square of the matrix has been normalized based on the true class, e.g. the square at (0,0) and (0,1) have been divided with the size of class 0, and the square at (1,0) and (1,1) have been divided by the size of class 1.In Fig. 2.23(a) when both classes have the same importance, class 0 is accurately predicted 96% of the times, but class 1 is predicted correctly only 41% of the times. In Fig. 2.23(b), where class 1 is considered more important than class 0, then the accuracy for class 0 dropped down to 74%, but accuracy for class 1 increased to 79% of the times.

by applying standard complexity control techniques (see Fig. 2.22).

After tuning the classifier we apply it on our testing data and we analyze the initial results (see Tab. 2.4). Since we are dealing with a highly skewed distribution (class 0 dominates our data set) we will compare against the majority predictor – a naive predictor that always predicts the most frequent class. Based on the accuracy, we see that our model outperforms the majority class predictor. However, when we dive into more details, namely we look at

precision[5], recall[6], and F1-score – the harmonic mean of precision and recall – we see that due to the highly skewed distribution the recall is very low (37%). This is also confirmed by the confusion matrix in Fig. 2.23(a).

We applied a Random Forest of 200 ensembles (larger numbers did not show improvement). Because the ensemble averaging will avoid overfitting, we also increased the maximum depth of each tree to 30. The random forests classifier improved the results, e.g. recall increased to 40%, and the F1-sore reached 51%.

Improving Recall

Up to now we have made the assumption that F_p and F_n have the same cost. However, this may not be the case. Providers would prefer having a high recall than high accuracy or precision (F_n will have a higher cost).

The same as when detecting patients with cancer, F_n has a much higher cost. In this case a naive classifier that predicts always zero – a patient doesn't have cancer – will be very accurate, due to the skewness of the two classes, but that's not necessary the best classifier.

Therefore, in this last section we will investigate how to improve recall, even if that means sacrificing accuracy. This is achieved by increasing the weight of positive samples. A higher weight on positive samples forces the decision tree to pay more attention to class 1 than class 0. Tab. 2.5 show the improvement of recall given different weights, e.g. for weight of 3 – positive samples are 3 times more important than negative ones – recall rises to 79% (from 37%), and for a weight of 10 – positive samples are 10 times more important than negative ones – recall is 94%. This change is also reflected in the confusion matrix (see Fig. 2.23(b)).

[5] $\frac{T_p}{T_p+F_p}$, where T_p is the true positive rate, and F_p the false positive rate.
[6] $\frac{T_p}{T_p+F_n}$, where T_p is the true positive rate, and F_n the false negative rate.

Scores	weight = 3	weight = 4	weight = 10
Accuracy	79%	75%	61%
Precision	47%	42%	33%
Recall	**71%**	**79%**	**94%**
F1-score	57%	55%	49%

Table 2.5: Results for decision tree with weighted samples. We can improve the recall by giving higher weight to positive samples, in expense of precision.

2.4.7 Summary

In this work, we applied machine learning techniques to predict cell-to-cell activity, based solely on past cellular activity records. We were able to achieve 85% accuracy and 94% recall, for the voice call data set provided by Telecom Italia for the city of Milan.

In future work, we will further improve the prediction by exploiting information outside the cellular activity data set, such as similarities between cells based on the socio-economic activity occurring in the surrounding areas. We could also extend the problem formulation (for example, instead of binary traffic prediction, we could predict multiple classes of traffic, such as no traffic, low traffic and high traffic between cells) or apply the methodology to other data sets (e.g., other cities or data instead of voice activity). Finally, we will investigate the use of this prediction methodology as a building block for network planning and control, urban ecology and smart city applications.

Chapter 3

Mining Individual CDRs for Ridesharing Recommendations

3.1 Introduction

The car has been for some time one of most heavily used ground transportation vehicles, and in the US, it is the dominant one. According to American Community Survey Reports currently there are more than 130 million commuters, and almost 80 of them drive alone when commuting [64]. This has many negative consequences: pollution, traffic, high car expenses, and loss of productivity. The dependency on cars is so high that 8.1% of the US workers – who did not work at home – had commutes of 60 minutes or longer [63].

Ridesharing is a promising approach for reducing the number of vehicles on the streets in order to address both individual and city-wide issues. Ridesharing refers to a mode of transportation in which individuals share a vehicle for a trip and split travel costs. It combines the flexibility and speed of private cars with the reduced cost of shared public transportation. Ride-sharing systems started in the US during WW-II. Early informal systems required pre-

defined rendezvous and prior familiarity among commuters, making them cumbersome and limiting the number of neighbors a person could ride-share with. More recently, web-based solutions, such as `Zimride.com` have allowed drivers and passengers to advertise route interests, thereby increasing the chances of finding a match. Most such web-based solutions target ridesharing among employees of large corporations. This is an ideal ridesharing use case since: (i) bootstrapping the system is easier by targeting a well defined set of users, *i.e.*, all employees of the same company, (ii) the user paths and time schedules are relatively aligned, since many users work in the same location and need to arrive there at the same time, and (iii) "stranger-danger" issues [18] are less pronounced among people working in the same company.

Thanks to smartphone devices (with their high penetration, positioning capabilities and ubiquitous cellular connectivity) flexible and dynamic scheduling of trips is now a reality. On demand *taxi-like companies*, such as Uber and Lyft, are enjoying great success and are a flagship example of sharing economy. Similar technology that is used to support on-demand taxi services can be used to support opportunistic ridesharing. This can benefit the entire population of a city/area in terms of reducing the cost for individual users and the congestion and pollution in the city.

Smartphone-based ridesharing technology gains momentum but still needs to deal with several issues including safety (traveling with strangers), liability (*e.g.*, accidents), as well as the bootstrapping problem (the more users a ride-sharing service has, the more the ride-sharing opportunities). However, even if the above problems were completely solved, the opportunities for ride-sharing would still depend on the underlying human mobility patterns and the layout of a city, which ultimately determine the route overlap.

In the first section of this chapter, we assess the potential of ridesharing (*i.e.*, how many cars can be removed from the streets of the city) and we find that ridesharing has indeed a great potential. Then, we develop an online ride sharing (ORS) system for matching users

that could share a ride, and we evaluate its performance.

3.2 Analysis of Potential

3.2.1 Introduction

In this section, we seek to understand what is the potential decrease in the number of cars in a city if people with similar mobility patterns are willing to use ride-sharing in their daily home/work commute. This is clearly an upper bound to the actual benefit of any practical system but it can be used to guide the deployment and policies regarding ride-sharing in a city. We assess this potential in four major cities using mobile and social data; we obtained two CDR data sets from a major cell provider (Madrid and BCN, in Spain, Europe), and we also collected data from Twitter (geo-tagged tweets) and FSQ (NY and LA, in US). A similar question has been asked before in [88], where the authors assumed a uniform distribution of home/work locations and concluded that ride-sharing has negligible potential. In contrast, we find that ride-sharing can provide significant benefits, depending on the spatial, temporal and social constraints for matching users. In particular, we take the following steps.

First, we infer home/work location of individual users, by adapting state-of-the-art techniques [54] to our CDRs and geo-tagged tweets. Also, we infer social ties among the users; we use phone calls in the CDR data and explicitly stated friendship in the Twitter data. These ties are later used for social filtering, to address concerns about riding with strangers.

Second, given a set of users with known home/work locations, we develop a framework for matching users that could share a ride. Our goal is to minimize the total numbers of cars and provide rides to as many users as possible. We consider several constraints including: spatial (ride-sharing with neighbors, *i.e.*, someone within a certain distance from

their home/work location), temporal (ride-sharing within a time window from the desired departure/arrival time) and social (ride-sharing with friends or friend-or-friends) constraints. We also consider two versions of the problem: `End-Points RS`, ride-sharing between home and work locations, and `En-Route RS`, allowing the possibility to pick up passengers along this route. Our formulation is rooted at the Capacitated Facility Location Problem with Unsplittable Demand. Since this is an NP-hard problem [56], and we want match more than 272K drivers and passengers, we develop efficient heuristic algorithms, namely `End-Points Matching` and `En-Route Matching` to solve the two aforementioned problems.

Third, we use our framework to assess the inherent potential of ride-sharing to exploit the overlap in people's commute in a city. We find that there is significant potential for reducing traffic via ride–sharing, the exact magnitude of which depends on the constraints assumed for matching, as well as on the characteristics of the cities and the type of data set (CDR vs Twitter). For example, our study shows that traffic in Madrid can be reduced by 59% if users are willing to share a ride with people who live and work within 1 km; if they can only accept a pick-up and drop-off delay up to 10 minutes, this potential benefit drops to 24%; if drivers also pick up passengers along the way, this number increases to 53%. If users are willing to ride only with people they know ("friends" in the CDR and OSN data sets), the potential of ride-sharing becomes negligible; if they are willing to ride with friends of friends, the potential reduction is up to 31%. Albeit upper bounds to the actual benefit, these positive results encourage the deployment and policies in favor of ride-sharing.

3.2.2 Related Work

Traditionally, carpooling studies focused in characterizing the behavior of carpoolers, identifying the individuals who are most likely to carpool and explaining what are the main factors that affect their decision [84]. Instead, in this thesis we focus on assessing its potential for

traffic reduction in a city. A similar study has been done before in [88], which assumed a uniform distribution of home/work locations in a city, and concluded that ride-sharing has little potential for traffic reduction. In contrast, we infer home/work locations from CDR and Twitter data and we find that they are far from uniform.

Some ride–sharing systems have been built over GPS [52, 87] data. He et al. [52] presents a route–mining algorithm that extracts frequent routes and provides ride-sharing recommendations based on these routes; they use the GPS traces of 178 individuals. Trasarti et al.[87] use GPS data to build mobility profiles for 2107 individuals, and match users with similar profiles; they also apply their algorithms to a GSM-like data set, which they synthesize by reducing the size of their GPS data. Bicocchi et al. [21] extract common routes from mobile traces and use them for ride-sharing recommendations. To the best of our knowledge, our work is the first attempt to study the potential of ride–sharing using CDR and OSN data. Although, our data have coarser granularity in terms of user trajectories (since we observe a user's location only when she makes a call or posts a geo-tagged tweet), they have information about orders of magnitude more users than previous carpooling studies and thus are better positioned to answer the question about the city-wide benefits of ride-sharing.

Compared to commercial ride-sharing systems, such as `Avego, Lyft, Uber`: our work is partly based on publicly available (*e.g.* geo-tagged tweets) as opposed to proprietary data; it has a larger number of users for the cities studied; it takes into account social ties for matching drivers and passengers; and it assesses offline the city-wide benefit of ride sharing, as opposed to online matching of passengers with a small set of dedicated drivers.

Our methodology on inferring home/work locations for individuals builds upon a recent work by Isaacman et al. [54, 55], on inferring important places from CDR. Social aspects of CDRs, *i.e.* the call graph, has been studied in [30], [42]. In this chapter, we combine both aspects, namely inferred locations and social ties, to restrict ride sharing accordingly. We do the same with the Twitter data too.

Other related studies focus on characterizing crowd mobility and urban environments using information from Twitter or FSQ. Wakamiya et al. [89] and Fujisaka et al. [49] have used geo-tagged Twitter data to study crowd mobility, and Frias-Martinez et al. [47] to characterize land use. FSQ has been used by Noulas et al. [69], [67] for modeling crowd activity. To the best of our knowledge, Twitter and FSQ data have not been used for carpooling.

The most closely related work is our preliminary study [33]. Compared to [33], in this chapter we make the following additional contributions: (1) we collect data from Twitter (geo-tagged tweets for NY and LA) in addition to CDRs, (2) we use CDRs from BCN, (3) we compare among the four cities, (4) we restrict ride sharing opportunities based on social ties, and (5) we estimate users' departure times from the data, instead of assuming a distribution.

3.2.3 Inferring Home/Work Locations from Data

The first step in assessing the benefits of ride-sharing is to infer where people live and work. To achieve this, we build on a state-of-the-art methodology that has been proven to infer important locations in people's lives with adequate accuracy [54]. We apply their methodology with some modifications in order to make it applicable to our scenario.

Data Sets

Tab. 3.1 summarizes our data sets, and the following subsections describe the data collection process.

Cell Phone Data:

We obtained CDRs from a major Telecommunication provider in Europe for two cities,

Data Set Name	Period	Number of Records	Total Users	Home/Work Users
CDR–Madrid	Sep 2013 - Dec 2013	820M	4.70M	272,479
CDR–BCN	Sep 2013 - Dec 2013	465M	2.98M	133,740
Twitter–NY	Nov 2012 - Feb 2013	5.70M	225K	71,977
Twitter–LA	Nov 2012 - Feb 2013	3.23M	155K	43,575
FSQ–NY	Nov 2012 - Feb 2013	362K	31.3K	–
FSQ–LA	Nov 2012 - Feb 2013	134K	13.6K	–
FSQ–US	Dec 2009 - Aug 2011	1.47M	40.1K	–

Table 3.1: Description of our data sets. The right column shows the number of users with inferred Home/Work locations, which is a subset of all the users. The Foursquare data sets were used to tune and validate the home/work inference methodology, for the Twitter data sets.

Madrid and Barcelona (BCN). The CDRs are from the period of September 2009 – December 2009, excluding the last two weeks of December, which are holidays. See Tab. 3.1 for more details.

Twitter and Foursquare (FSQ):

Many users access Twitter from mobile apps and some of them choose to reveal their current location (typically as GPS coordinates) in their tweets, thus making Twitter an important source of human mobility information. We used the Twitter's *Streaming API* [6] in order to obtain individuals' mobility traces in large geographic areas. We collected geo-tagged tweets from the metropolitan areas New York and Los Angeles for a period of four months – from November 2012 until February 2013. This was possible thanks to Twitter's Public Stream Service where you can specify the geographic area that you are interested in. See Tab. 3.1 for more details.

Geo-tagged tweets contain location information, but they lack location semantics, which are crucial for inferring individuals' home/work locations and commuting routes. We collected this information from FSQ – a large location-based OSN with more than 30M users. FSQ does not provide an API for data collection but its users can post their check-ins in Twitter and other OSNs. We obtained FSQ check-ins from our Twitter data set. In addition, we

(a) Headquarters of Telefonica in Madrid

(b) Home Area. Coordinates: 40.503736, -3.635469

Figure 3.1: Example of residential and working areas

exploited another FSQ data set that we obtained with the help of the authors of [77]. The latest data set was obtained by crawling publicly available tweets of check-ins in the US, and spans the period: December 2009 - August 2011. See Tab. 3.1 for more details.

Home/Work inference methodology:

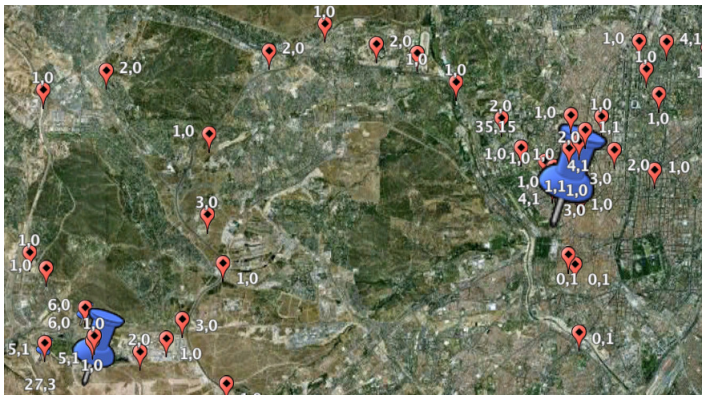
We apply the methodology of Isaacman et al. [54] for inferring important places for cell phone subscribers from (1) CDR data and (2) ground truth for a subset of subscribers. First the recorded cell towers of a user are clustered to produce the list of places that the user visits. Then, regression analysis is applied to the ground truth users (clusters and their true important locations) to determine the features of the clusters that represent important places. The used features are: (1) the number of days that the user appeared on the cluster; (2) the duration of user appearances on the cluster; and (3) the rank of the cluster based on number of days appeared. Once important locations have been inferred, and the algorithm chooses which of these are home and which are work locations. According to their results, the best features that characterize home and work are: (4) the number of phone calls between 7PM - 7AM, i.e. *Home Hour Events*, and (5) number of phone calls between 1PM - 5PM, i.e. *Work Hour Events*.

For our CDRs, first, we filter out users for whom we have too little data: *i.e.* users with less than 1 call per day on average, or less than 2 clusters with 3 days of appearance and 2 weeks of duration – the specific filtering parameters are consistent with [54]. Then, we tune the methodology of [54] to our needs. More specifically, we build two classifiers, one for home and one for work, and we train them using the 5 features described above and the ground truth, which is described in the following section. Once the training on the ground truth is done, we apply the classifiers to the rest of the users. Finally, after classification, we keep only the users who have only one inferred home location, and a different inferred work location, since we are interested only in commuters. Applying the home/work inference methodology to our CDR data, we are able to infer the home/work locations of more than 272K individual users in Madrid, and more than 133K users in BCN (See Tab. 3.1). Finally, we apply the same methodology in our Twitter data – FSQ data serves as ground truth – and we infer home/work locations for 71K users NY, and 43K users in LA.

Obtaining Ground Truth:

In [54], a set of 37 volunteers reported their most important locations, including home and work. This was used to tune their methodology, *i.e.* in the regression analysis, before applying it to the rest of their users – around 170K.

Ground Truth for CDR Data: For the CDR data, we obtained our ground truth for a select subset of users based on a previous study [82], which characterizes areas in Madrid. In particular, we exploited strictly residential and strictly industrial areas (see Fig. 3.1 for example), which offer a clear distinction between home and work. To this end, we selected 160 users that appeared for many days in only one such residential area during 7PM - 7AM ("home hours"), and only one such industrial area during 1PM - 5PM ("work hours"). Then, the location inside the residential area is pointed as the user's Home, while the location inside the industrial area is pointed as the user's work. For each one of the 160 users, we visually inspected their recorded locations through Google Earth. Fig. 3.2 shows a selected ground



(a) A "ground truth" user

(b) Zooming in at home

(c) Zooming in at work

Figure 3.2: A ground truth example. The red paddles show the cell towers, while the blue pushpins the clusters. The numbers next to each mark indicate the number of weekdays and weekends she appeared in that location. Also, the size of each mark is proportional to the days of appearance.

truth user.

Ground Truth for Twitter Data: We used the Foursquare data to build the ground truth for the geo-tagged Twitter data sets by selecting users who appear more than a week in a location tagged as Home(Private), and the same duration in a location containing one of the tags: Professional, Office, or Work. For each one of these users we define their home to be the location tagged as home with highest number of days of appearance, and as work the location tagged as work with most days of appearance. We also manually inspect their Twitter account and, when possible, their LinkedIn accounts. In the FSQ-US data set, we found 481 such users, and in the FSQ-NY and FSQ-LA data sets we found 98.

Validation:

Percentile	25^{th}	50^{th}	75^{th}	95^{th}
Our Home Error	0.0	0.01	0.49	13.62
Home Error in [54]	0.85	1.45	2.06	6.21
Our Work Error	0.1	0.03	1.52	16.09
Work Error [54]	1.0	1.34	3.7	34.17

Table 3.2: Comparing the home/work identification error to [54].

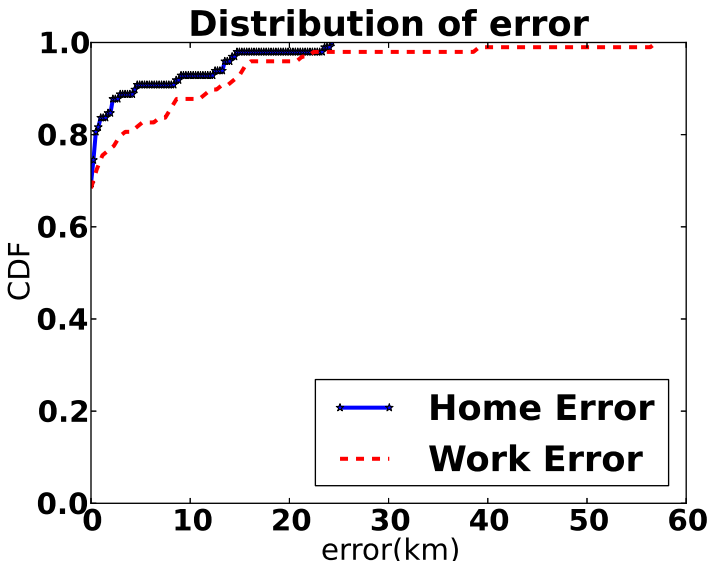


Figure 3.3: CDF of error for the home/work inference methodology. Inferred home/work locations from Twitter are compared against the declared locations in Foursquare.

Fig. 3.3 shows the accuracy of the home/work inference methodology for our Twitter data set. We use the FSQ-US data set to train the classifiers. Then, for the ground truth users that appear both in the Twitter data set and the FSQ data set, we infer their home/work locations using the geo-tagged tweets, and then we compare the inferred home/work locations to the ones in FSQ. In Tab. 3.2 we compare the accuracy of the home/work identification methodology with the reported accuracy in [54]. We see that in the case of the 75^{th} percentile the home error has decreased by 76% , and the work error has decreased by 59%. For a few cases, our error is higher. We attribute our overall higher accuracy to the more precise location information in the Twitter-Foursquare data sets. Finally, Fig. 3.4 shows a visual comparison between our results and the characterization of the Madrid's areas from [82], and indicates a strong agreement between our results and the related work.

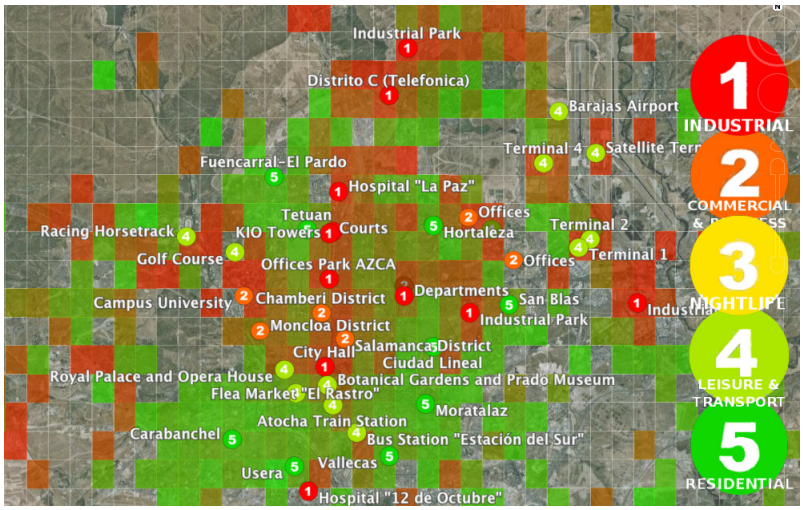


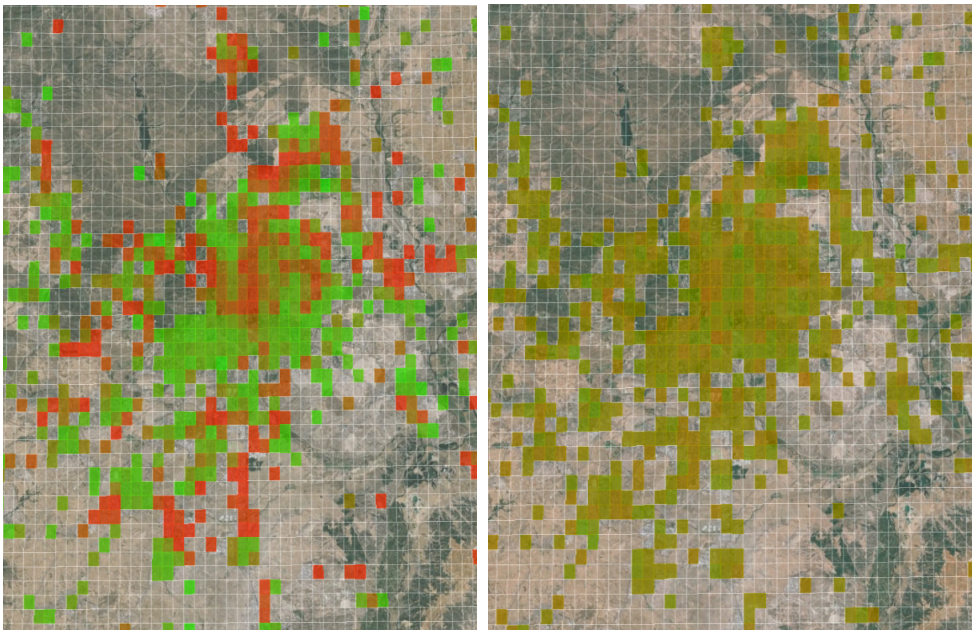
Figure 3.4: Characterizing Madrid based on the inferred home/work locations, and comparison to the characterization of [82]. We break the city into a grid, and color each square with a combination of green and red. Green squares have relatively more home than work locations; while red squares have relatively more work than home locations. We observe that the squares that we colored red contain contain more circles, indicating industrial and commercial zones, than residential zones. Also, squares colored green contain more residential than industrial zones.

Differences from the Uniform Distribution:

We find that home/work distribution is far from uniform, which was assumed in [88], in the following aspects:

Segregation of home and work areas: According to Fig. 3.5, Madrid contains segregated home and work (*e.g.* industrial) areas. In work areas, there is a relatively large number of working places, while in home areas there is a relatively large number of home locations Fig. 3.5(a). To illustrate the difference, we show how the city would look if the home/work distribution were uniform, Fig. 3.5(b).

Non-uniform density: The density of home and work locations in various areas is quite different from uniform, as shown in Fig. 3.6; 30% of most popular home areas – areas with most home locations – contain 75% of the homes; if home/work distribution was uniform then the top 30% of home areas would contain only 30% of the homes.



(a) Inferred (b) Uniform

Figure 3.5: Inferred vs. uniformly distributed home/work locations. Fig 3.5(a) shows a city with segregated home and work areas, while Fig. 3.5(b) shows a city where all areas are the same.

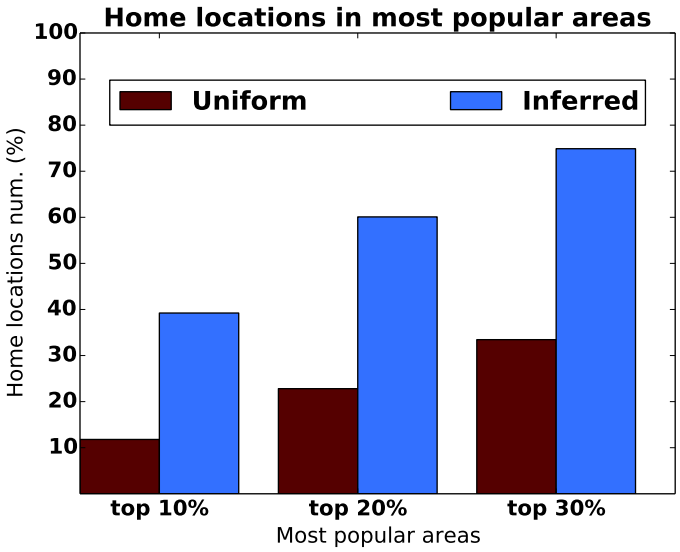
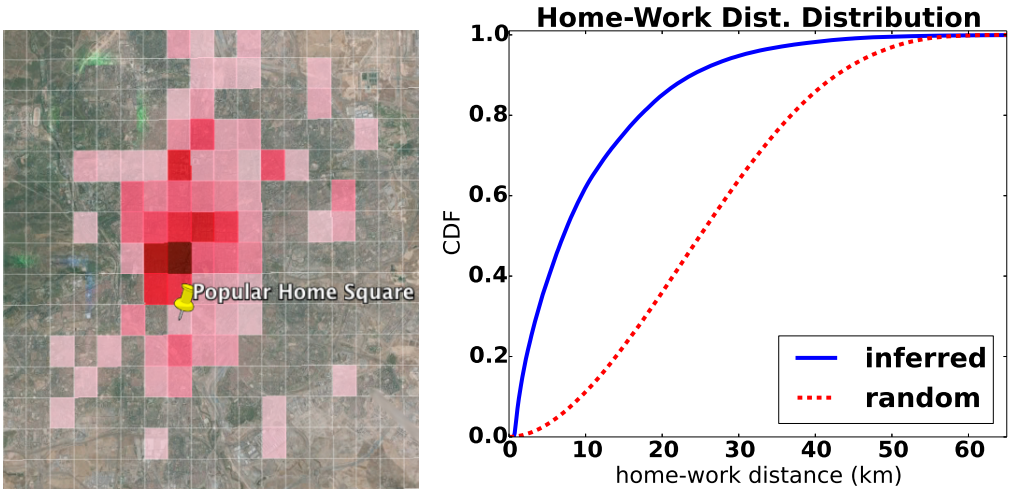


Figure 3.6: Number of home locations for the 10%, 20%, and 30% most popular areas. In the inferred distribution, areas vary in popularity (highly populated and sparsely populated ones), while in the uniform they are equally popular.



(a) Most popular home area (b) CDF of home-work distance

Figure 3.7: Distance between home and work locations. 3.7(a) shows the square grid with most homes (yellow paddle), and where are the corresponding work locations; stronger the colors indicate higher concentration of work locations. Users tend to work close to home.

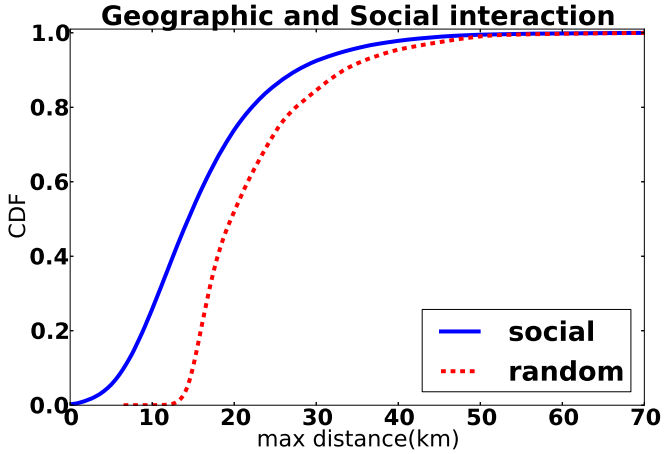


Figure 3.8: Distances between users who have social ties – inferred from the calls – vs. distance between random strangers. The distance between two users u and v is the maximum of their home and work distance. This figure indicates correlation between social and geographic proximity.

Relatively short home-work distances: As seen in Fig. 3.7, users tend to work close to where they live. For the grid square with the highest number of users who have their home there, as shown in Fig. 3.7(a), the corresponding work locations tend to be close by. Also, according to Fig. 7(b), the home-work distances are shorter compared to what they would be if home

and work were randomly distributed.

Geographic distances and social ties: In a later section, we will consider social ties among users, inferred from calls (CDRs), or declared relations (Twitter). In Fig. 3.8, we compare the average distance of each user u to her friends, vs. her geographic distance to randomly selected strangers (i.e., users who are not neighbors of u in the social graph). According to Fig. 3.8, the geographic distance between users who have social ties are shorter, on average, in comparison to strangers.

Departure Times:

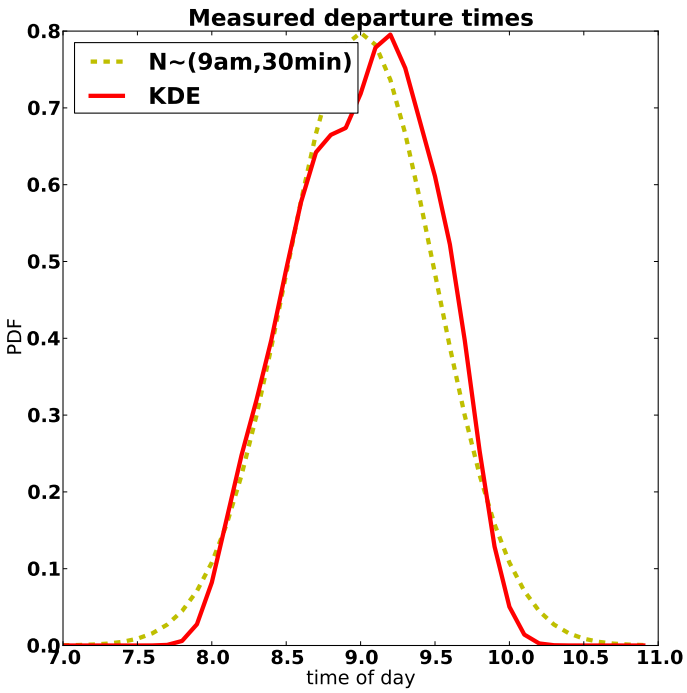


Figure 3.9: Distribution of home-departure Times. A normal distribution with mean at 9 am, and standard deviation 30 minutes, is a close approximation to the inferred departure times from our data. The continuous line is what we get via Kernel Density Estimation from our data.

We estimate departure times of individual users from consecutive home/work calls. More specifically, we use pairs of calls where one is a home call, the other a work call, and the

time difference between the calls is less than 2·trip_time, where trip_time is the time distance between home and work, as obtained from a popular Online Map service.

For each user, we find her departure time from home by taking the median of the calls, that: (1) were made between 8 am and 10 am from home, and (2) were followed by a work call no more than 2·trip_time later. Similarly, we find her work departure time, by taking the median of the calls, that : (1) were made from work between 4pm and 6pm, and (2) were followed by a home call no more than 2·trip_time later.

The distribution of home departure times for all individuals who had such calls is shown in Fig. 3.9 – each individual is required to have at least three such calls; there were 484 such users in our data set. The departure time from work follows a similar distribution, which is omitted due to lack of space.

3.2.4 End-Points Ridesharing

In this section, we formulate the problem of `End-Points RS`, *i.e.* ride-sharing among people that live and work close to each other. We develop a practical algorithm, we apply it to the users with inferred home/work locations, and we compute the number of cars that can be reduced under different scenarios.

Formulation

Let V denote a set of potential drivers and $c(v)$ the capacity, in terms of available seats, of the car of driver $v \in V$ and $p(v)$ a penalty paid if driver v is selected for driving her car and picking up passengers. Let $h(v, u)$ denote the geographic distance between the home locations of drivers v and u and $w(v, u)$ the corresponding distance between their work locations. Let δ denote the maximum acceptable distance between a driver's home/work and

the home/work of passengers that she can pick up in her car, *i.e.*, v can have u as passenger only if: $max(h(u,v), w(u,v)) \leq \delta$

Let $d(v,u)$ denote a virtual distance between v and u, defined as follows:

$$d(v,u) = \begin{cases} h(v,u) + w(v,u), \\ \text{if} \quad \max(h(v,u), w(v,u)) \leq \delta \\ \\ \infty, \quad \text{otherwise} \end{cases}$$

Our objective is to select a subset of drivers $S \subseteq V$, and find an assignment $a : V \to S$, that minimizes $P(S) + D(S)$, the sum of penalty and distance costs, while satisfying the capacity constraints of cars. The two costs are defined as follows:

$$P(S) = \sum_{v \in S} p(v) \quad \text{and} \quad D(S) = \sum_{v \in V} d(a(v), v)$$

where $a(v) \in S$ is the driver in S that is assigned to pick up passenger v (can be himself if v is selected as a driver). By setting $p(v) > 2\delta \cdot c(v)$ we make sure that an optimal solution will not increase the number of cars in order to decrease the (pickup) distance cost between a driver and its passengers [1]. The above problem is an NP-hard *Capacitated Facility Location Problem with Unsplittable Demand* in metric distance: the set of potential drivers corresponds to the set of locations; the set of chosen drivers corresponds to opened facilities; car capacity corresponds to facility capacity; distance $d(v,u)$ corresponds to the cost of assigning a location v to the facility u. Efficient approximation algorithms are known for this type of facility location problem [56].

[1] For all (u,v) pairs withing constraints, $d(v,u) \leq 2\delta$, therefore in worst case a full car v can increase the total cost by $2\delta \cdot c(v)$. We set the penalty for every car, to be higher that the worst case scenario.

The above formulation includes spatial constraints only. Next, we refine our formulation to include time. In the previous section (see Fig. 3.9), we showed that departures from home and work can be approximated by a normal distribution, centered at 9 am and 5 pm respectively, with standard deviation σ. We introduce the delay tolerance τ that captures the maximum amount of time that an individual can deviate from her normal schedule in order to share a ride. More specifically, if $LH(u)$ denotes the time a person u leaves home to go to work, and $LW(u)$ expresses the time she leaves work in order to return to home. Then, two people u and v, can share a ride only if:

$$max(|LH(u) - LH(v)|, |LW(u) - LW(v)|) \leq \tau$$

The introduction of the temporal constrains will change the virtual distance between v and u to the following :

$$d(v,u) = \begin{cases} h(v,u) + w(v,u), \\ \text{if} \quad \max(h(v,u), w(v,u)) \leq \delta \\ \text{AND} \quad |LH(u) - LH(v)| \leq \tau \\ \text{AND} \quad |LW(u) - LW(v)| \leq \tau \\ \\ \infty, \quad \text{otherwise} \end{cases}$$

A Practical Algorithm

In this section, we modify the existing approximation algorithm [56] for the facility location problem described above and design a heuristic that can cope with the size of our matching needs, the biggest of which has 272K users.

The algorithm in [56] starts with a random solution and improves it iteratively via local search. At each iteration, there are $O(n^2)$ candidate solutions, where n corresponds to the

number of potential drivers. For each one of them, it finds the assignment (passengers to drivers) that minimizes the cost; this is done in polynomial time by solving an appropriately defined instance of the *transportation problem*. The algorithm terminates when local search cannot find a better solution.

We modify the algorithm in three ways. First, since the quality of the solution depends mostly on the number of drivers, we try to keep that number as low as possible. Therefore, we use the b-matching [28] algorithm to generate the initial solution, instead of generating it randomly. The input to the b-matching algorithm consists of the set of potential drivers V, a function $o(v)$ that defines the set options for a potential driver v *i.e.* $o(v) = \{u|d(u,v) < \inf\}$, and a global ordering of the potential drivers, O. The global ordering will be based on the number of options; the fewer the options, the higher the position in O. By using b-matching with a global order we are guaranteed to find a solution in $O(n)$ time [28]. For each match generated by b-matching, we assign the potential driver with the most occupied seats to drive; we make sure that every user in V appears in only one car. This solution has much lower cost than the random one by paying $O(nlog(n))$ for sorting the users to generate the global preference list and $O(n)$ for the matching.

Second, solving a transportation problem with 272K users is computationally expensive. Therefore, we need to modify the local search steps of the approximation algorithm. Given an initial solution we leave the users commuting in cars of four as they are and search for better assignments only for the rest. This reduces the size of the transportation problem and speeds up the process of generating the assignment.

Third, reducing the size of the transportation problem is not enough; we also need to reduce the neighborhood of candidate solutions. Given an initial set of drivers, S, we create a fixed size neighborhood, where each solution S' is created by doing random changes in S. The reason why we do that is because considering all potential solutions that differ from S only by one, means that we have to examine $O(n^2)$ candidate solutions; that makes each iteration

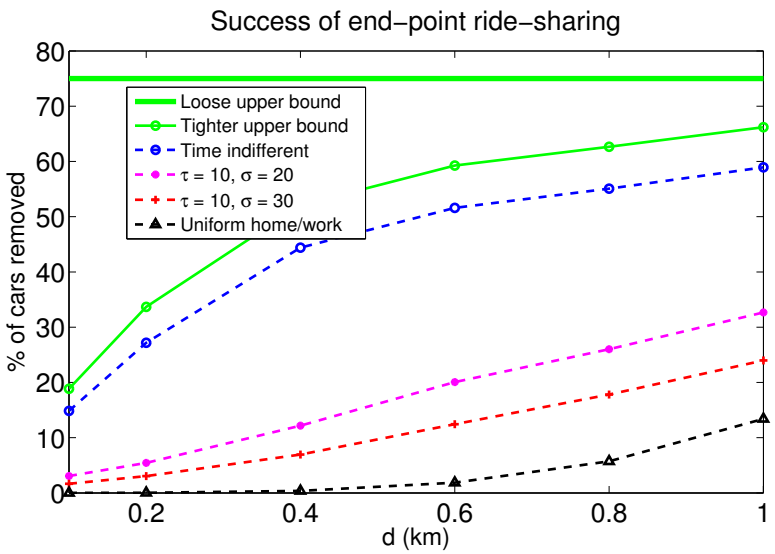


Figure 3.10: Benefits of `End-Points RS`.

very expensive. Therefore, the fixed size solution helps us speed up the time we spend in each improvement step.

Without the above modifications it would be impossible to solve the problem in real time. Solving an instance of the transportation problem for 270K users required a couple of hours for $\delta = 0.6$ km, and even more when $\delta = 0.8$ or $\delta = 1.0$ km. Therefore, solving $O(n^2)$ such problems for a single iteration becomes too expensive. Moreover, in our experiments, we observed that the improvement steps would add little value to the solution offered by the b-matching.

Results

We now calculate the effectiveness of `End-Points RS` based on our data sets. For ease of exposition, we will focus on the Madrid metropolitan area (we cover the remaining cities in a later section). We reduce the size of our data set by randomly selecting only 60% of the users. We do that to capture the fact that only 60% of the population has a car in the area of Madrid [2]. We also show results for the case that half of the car owners use their

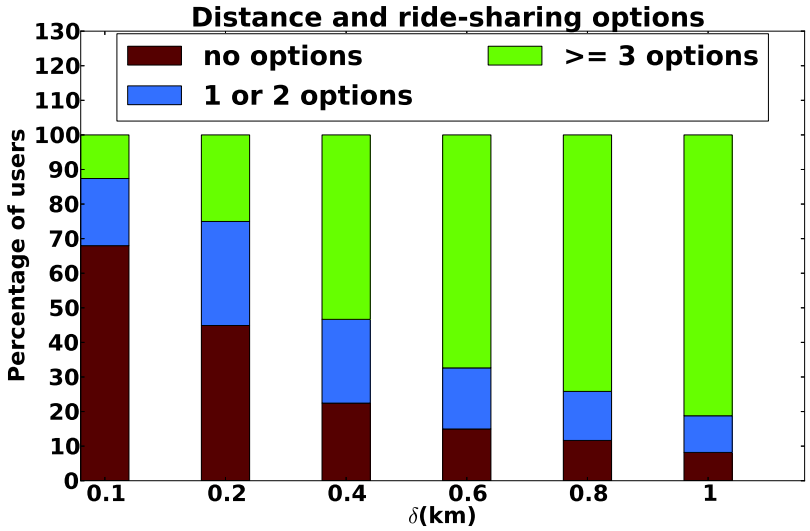


Figure 3.11: How δ affects the ride-sharing options

car at their daily commute (the results are quantitatively similar). For the remaining of the section, we will refer to users who can share rides with a specific user v, as *options of* v. We compute the reduction of cars, as % of the initial number:

$$\text{success} = \frac{\#(\text{init. cars}) - \#(\text{ride-sharing cars})}{\#(\text{init. cars})} \cdot 100$$

using the following algorithms:

Loose upper bound: Given our definition of success, we cannot do better than 75%, when all cars carry 4 people.

Tighter upper bound: Assuming that all users with at least one ride-sharing option commute in cars of 4.

Time-indifferent matching ($\tau = \infty$): This is the practical algorithm described in the previous section.

Time-aware matching: This is the version of the algorithm that considers timing constraints under the assumption of normally distributed departure times.

Uniform home/work: Ride-sharing assuming that home/work locations are distributed uniformly.

Fig. 3.10 presents what happens when the users are willing to tolerate a detour of δ km and deviate τ minutes from their departure times, in order to share the same car with another individual. The results show that with a modest delay tolerance of 10 minutes and a detour distance of 1.0 km (a couple of city blocks) more than 20% of the cars can be saved. The success ration improves when δ or τ increase. The diminishing improvement with increasing δ can be explained by the number users' options, given the distance δ. In Fig. 3.11, the red color represents the users with no options, the blue color the users with 1 or 2 options, and the green color the users with 3 or more options. We see that the success of ride-sharing is proportional to the number of users with 3 or more options.

Fig. 3.10 also shows that the potential of `End-Points RS` is quite small in the case of uniformly distributed home/work locations; note that no time constraints were applied in this case. If we apply time constrains too, then the success of `End-Points RS` is even smaller, e.g. for $\delta = 1$ km, $\tau = 10$ min, and $\sigma = 30$ min, its potential becomes 0.2%

3.2.5 En-Route Ridesharing

The effectiveness of ride-sharing can be greatly improved by picking up additional passengers en-route; a driver that lives in a sparsely populated area might not have any neighbors to fill her seats, but once she enters the city she can pick several passengers that are on her way. In order to quantify the benefits of en-route ride-sharing we obtain routes from a popular Online Map service for the 272K users with inferred home/work locations, and we extend the algorithm of the previous section. Again, we will focus on Madrid.

En-Route Algorithm

We use an iterative algorithm with the following steps:

1. Run the basic `End-Points RS` algorithm.
2. Exclude from the solution cars that get fully packed (a car of 4). Then order cars in decreasing order of passengers and start "routing" them across the urban environment (*e.g.* Madrid) using data from the Online Map service.
3. When the currently routed car v meets a yet un-routed car v', then v is allowed to steal passengers from v' as long as it has more passengers than v' (a rich-get-richer strategy). Whenever a routed car gets fully packed it is removed from further consideration. Whenever a car with a single passenger is encountered the number of cars is reduced by one.
4. The algorithm finishes when no change occurs.

These steps are repeated until there is no possible improvement. The rich-get-richer rule leads to convergence, since it forces cars to either become full or to stay home (the driver becomes a passenger in another car). The algorithm converged in every single execution.

Results

Fig. 3.12 shows the performance of `En-Route RS`. To make the comparison with `End-Points RS` easier we summarize our results in Tab. 3.3. One can see the significant improvement obtained through `En-Route RS`, which in several cases comes within 10% of the optimal performance.

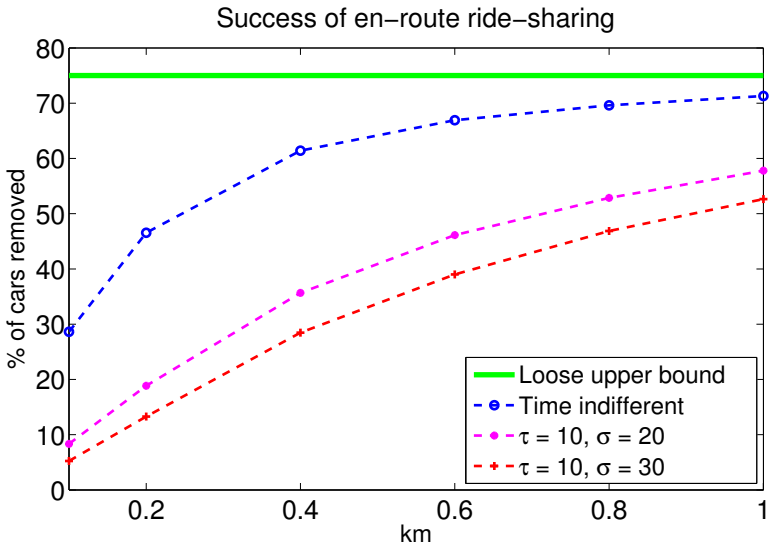


Figure 3.12: Benefits of `En-Route RS`.

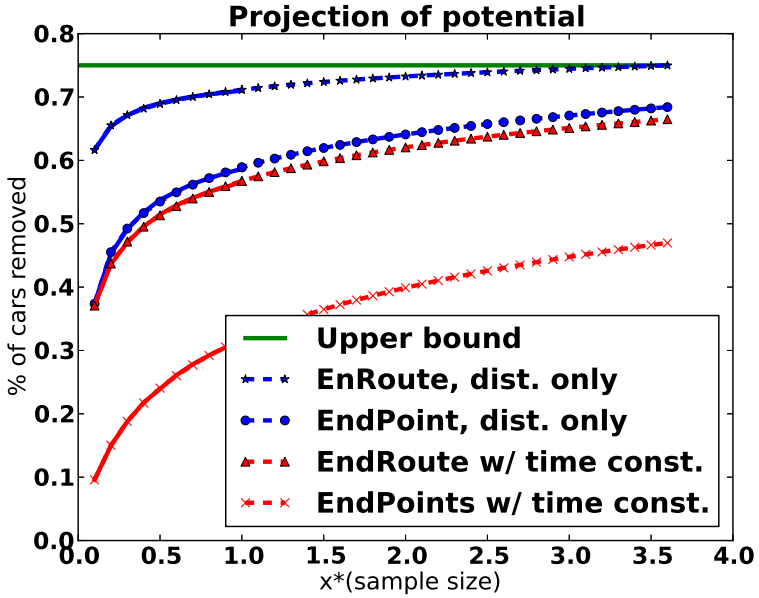


Figure 3.13: Extrapolation to commuters' size. "Sample" refers to the 272K users with inferred home/work locations in Madrid. The solid lines correspond to values generated from our data set, while the dashed lines correspond to values generated through extrapolation.

Projection to the entire commute population: All previous results have been produced based on the 272K users with inferred home/work location in Madrid. This, however, represents only roughly 8% of the total population of the city. To get a feeling of the ride-sharing potential based on the entire population, for which we do not have location information,

Sample (%)	δ (km)	τ (min)	σ (min)	End-Points RS (%)	En-Route RS (%)
30	1.0	–	–	54	65
30	1.0	10	30	17	47
60	1.0	–	–	59	70
60	1.0	10	30	24	53
100	1.0	–	–	62	71
100	1.0	10	30	30	56
360	1.0	–	–	70	75
360	1.0	10	30	44	65

Table 3.3: Effect of population size on the performance of End-Points RS and En-Route RS in Madrid. "Sample" refers to the 272K users with inferred home/work locations in Madrid. 100% means using all of them. 30% and 60% means using a random subsets, while 360% means projecting the potential to the entire commuters' population of Madrid.

Graph	Nodes #	Edges #	Mean degree	Median degree
call graph Madrid	4M	21M	6.0	1
twitter graph NY	132K	725K	10.95	5

Table 3.4: Graph sizes

we extrapolate to a larger number of users. We repeat the calculation of ride-sharing with different subsets of our total 272K users, and fit numerically these data points to a logarithmic function (in order to capture the diminishing effect). Then we extrapolate the potential of ride-sharing for larger population sizes. The results are summarized in Tab. 3.3 and shown in Fig. 3.13, where we see that the population size has a progressively diminishing results on the ride-sharing potential. In the remainder of the section we will report results for both our 8% sample, and extrapolation to the commuters' population.

3.2.6 Social Filtering - Riding with Friends of Friends

In this section, we present how social filtering affects the potential of ride-sharing. Instead of assuming that anybody is willing to share a ride with anybody else, we introduce social constraints in selecting ride-sharing partners. The social constraints are represented by

city	filter	End-Points RS (%)	En-Route RS (%)	En-Route RS extrapolation (%)
Madrid	no filter	30	56	65
Madrid	1-hop	0.26	1.1	–
Madrid	2-hop	3.7	19	31
NY	no filter	20	44	68
NY	1-hop	0.18	1.2	–
NY	2-hop	2.1	8.2	26

Table 3.5: Social Filtering. The potential or End-Points RS and En-Route RS for $\delta = 1.0$ km (distance constr.), τ=10, $\sigma = 30$ (time constr.). The third and the forth column show the potential for sample size, while the last column shows the potential of ride-sharing extrapolated to the commuters' population.

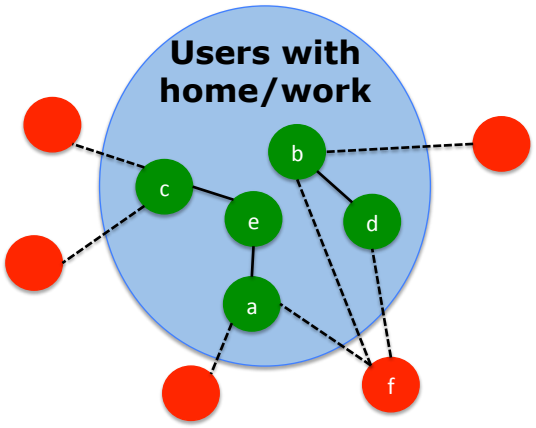
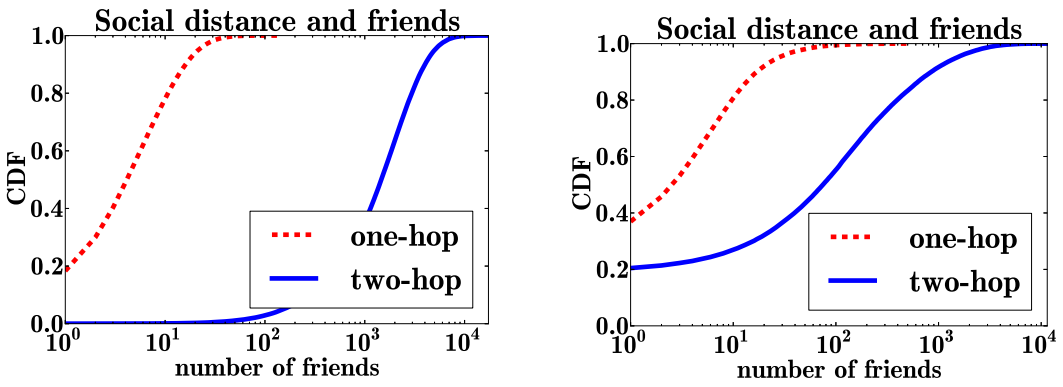


Figure 3.14: How social filtering works. Green nodes represent users with inferred home/work locations. Red nodes represent their neighbors (without inferred home/work locations). We only consider ride-sharing among the green nodes. In one-hop filtering, a can share a ride only with e. In two-hop filtering, a can share a ride with e,c,b, and d.

graphs, e.g. as shown in Fig. 15: the nodes correspond to users, and the edges correspond to social ties. A user considers sharing a ride with a one-hop neighbor, or with a two-hop neighbor (a friend of a friend).

Given that we have two different types of data sets – CDR and geo-tagged tweets – we need to use two different definitions of edges. In the case of CDR data [40] [70], choosing a threshold condition for an edge between two users involves a trade-off between the strength of the tie and the number of edges. When choosing a threshold one needs to take into account the needs of the application [31]. In this study, we create an edge in the social graph between



(a) call friends in Madrid

(b) declared friends in NY

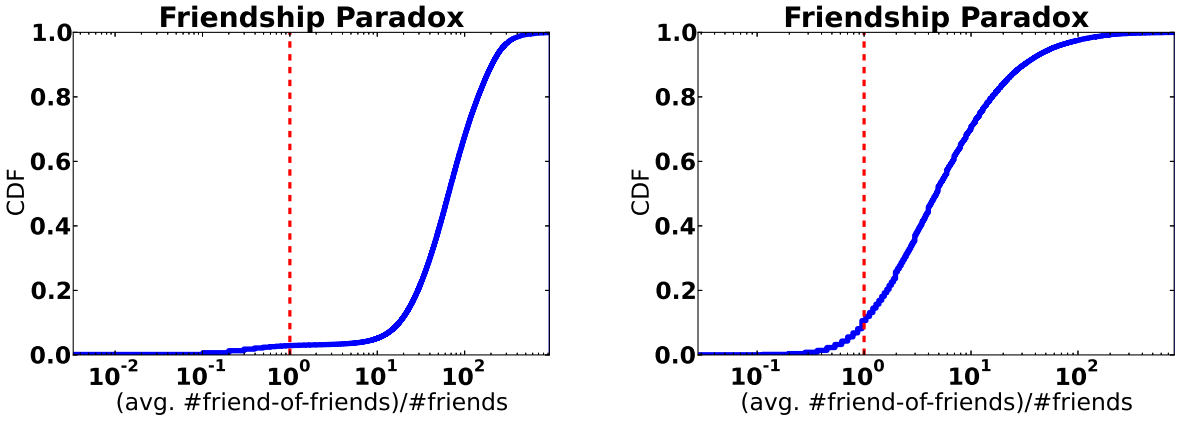
Figure 3.15: Number of friends for the users with home and work address.

two users when there is at least one call between them. We experimented with various definitions, and we found that – due to the small number of users with inferred home/work locations – higher thresholds would result in extremely sparse, thus useless,[2] graphs.

In the case of Twitter, we crawl the friends and the followers of the users for with inferred home/work locations, and we create an edge in the social graphs if there is a bidirectional edge on Twitter. See Tab. 3.4 for graph details. Moreover, to be sure that the friend nodes in our Twitter graph represent real people we considered only users who had at least one geo-tagged tweet. Finally, in both CDR and Twitter cases, we filtered out nodes with more than 1000 friends, in order to exclude popular phone services, or celebrities, respectively.

Now, we examine how social filtering affects the potential of ride-sharing. Fig. 3.14 illustrates the social filtering process. Lets start with Madrid. As we can see from Tab. 3.5 the potential of ride-sharing is quite low when users are willing to share a ride only with their one-hop friends. This is expected, since the graph shows only a small portion of a user's friends, and the users for whom we have home/work addresses are only a small subset of all users.

[2]Using a reciprocal call, as a threshold, would result in a graph with 2.4M nodes, and 3.7M edges. In that case, 92% of the users had zero one-hop neighbors with whom they could share a ride. As a result, the ride-sharing potential was 2% (5.1% with extrapolation) for `En-Route RS` with 2-hop social filter, and $\delta = 1.0$ km (dist. constr.), τ=10, $\sigma = 30$ (time constr.)



(a) Call graph

(b) Twitter graph

Figure 3.16: The CDF of the ratio between average number of friends-of-friends over number of friends. Friendship paradox holds when this ratio is greater than one (over 90% of the users both in figures).

From Fig. 3.15(a), we can see that 80% of the nodes in the call graph have no more than 10 one-hop friends, whose home/work addresses have been identified. However, if users are willing to share rides with friends of friends, then from Tab. 3.5 we can see that, even with a sparse social graph, there can be considerable gain from `En-Route RS`. This can be explained from Fig. 3.15(a), in which we can see the much higher number of two-hop than one-hop friends. In all data sets, there is a considerable improvement; e.g., in Madrid, ride-sharing has a potential of 19% (or 31% if extrapolated to the entire population of Madrid).

In general, the number of nodes and edges in the social graph is crucial for any ride sharing application that wants to exploit social filtering. Moreover, the difference between the large increase in the ride-sharing potential when using friend-of-friends can be attributed to the friendship paradox ("on average your friends have more friends that you do", [53, 43] that also holds in our data sets as illustrated in Fig. 3.16.

scenarios	`End-Points RS` ratio. (%)	`En-Route RS` ratio. (%)
τ=10, σ = 30	3.3	1.8
Social constr.	68	14

Table 3.6: Madrid vs. BCN. This table shows the difference in ride-sharing potential between BCN and Madrid, for both `End-Points RS` and `En-Route RS`, in two different scenarios: (1) δ = 1.0 km, τ=10, and σ= 30, and (2) δ = 1.0 km, τ=10, σ= 30, and two-hop friends. The ratio is computed as : $((BCN - Madrid)/Madrid) * 100$

scenarios	`End-Points RS` ratio. (%)	`En-Route RS` ratio. (%)
τ=10, σ = 30	-33	-9
Social constr.	-50	-46

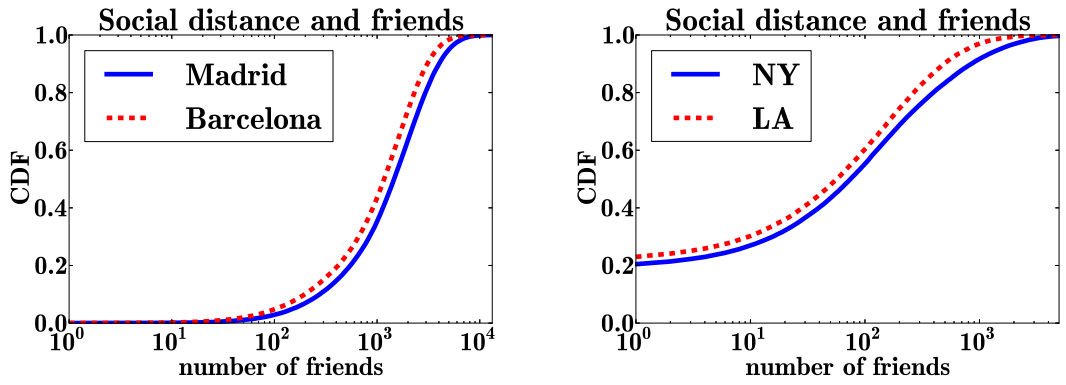
Table 3.7: NY vs. LA. This table shows the difference in ride-sharing potential between New York and Los Angeles, for both `End-Points RS` and `En-Route RS`, in two different scenarios: (1) δ = 1.0 km, τ=10, and σ=30, and (2) δ = 1.0 km, τ=10, σ= 30, and two-hop friends. The ratio is computed as: $((LA - NY)/NY) * 100$

3.2.7 A Tale of Four Cities

In this section, we compare the potential of ride-sharing in the four cities (Madrid, BCN, NY, and LA).

We start by comparing Madrid and BCN. The first row of Tab. 3.6 shows that, for spatio-temporal constraints only, the potential or ride-sharing in the two cities is very similar, with the potential of `En-Route RS` being slightly higher in BCN. In the second row, we show that when also considering social constraints, the relative difference in ride-sharing benefit between the two cities becomes becomes much higher: the potential of `End-Points RS` in BCN is 68% higher, and the potential of `En-Route RS` in BCN is 14% higher. This difference cannot be explained by the social graph, since, as we can see from Fig. 3.17(a), the users in both cities have almost the same number of friends. We attribute the better potential in BCN to its higher population density: Madrid has a density of 5,390 $people/km^2$, while BCN has a density of 15,926 $people/km^2$.

The same observation holds in the comparison between the two US cities. The potential or



(a) Barcelona vs. Madrid

(b) NY vs. LA

Figure 3.17: Comparing the CDF of 2-hop friends for Madrid vs. Barcelona, and NY vs. LA.

ride-sharing in NY is higher that the potential of ride-sharing in LA – see Tab. 3.7. The difference gets even higher when time or social constraints are included – see Tab. 3.7. Again, the difference in the potential of ride-sharing can be explained by the densities of the two cities: LA has a density of 3,124 $people/km^2$, and NY has a density of 10,429 $people/km^2$.

We obtained the mobility data for Madrid and BCN from CDRs, and we obtained the mobility data for NY and LA from geo-tagged tweets, therefore a comparison between European and US cities may lead to incorrect conclusions. However, both comparisons (Madrid *vs.* BCN and NY *vs.* LA) show that ride-sharing is more beneficial in denser cities, especially when time and social constraints are considered.

3.2.8 Summary

We used mobile and social data to demonstrate that there is significant overlap in people's commute in a city, which indicates a high potential benefit from ride-sharing systems. This is clearly an upper bound to any practical ride-sharing system, but the positive result motivates the deployment of such systems and policies. Our results indicate that en-route ride-sharing

with up to two-hop social contacts offers a good trade-off between technological feasibility, people's security concerns, and a substantial impact on traffic reduction. A more detailed summary of our findings is as follows.

We started by considering `End-Points RS` in which rides can be shared only with neighbors with nearby home and work. Even with a modest detour of 1 km we observed a great potential reduction of cars. In the case of Madrid, this reduction is 59%, based on our location data set that captures close to 8% percent of the total population. Our estimation of the ride-sharing potential extrapolated to the total commuting population of the city is significantly higher.

The distribution of home/work locations, which is far from uniform is crucial to the success of ride-sharing: if Madrid had a uniform home and work distribution then the reduction would be 13% assuming only spatial constraints, and 0.2% assuming time constraints too. This is in agreement with [88] and shows that ride-sharing has negligible benefit in a city with uniform home/work distribution.

Adding time constraints, the effectiveness of ride-sharing becomes proportional to the driver/passenger waiting time for a pick-up, and inversely proportional to the standard deviation of the distribution of departure times. With a standard deviation of 30 min, a wait time up to 10 min and a δ of 1km there is a 24% reduction of cars in Madrid.

`En-Route RS`, i.e., allowing passenger to be picked up along the way, yields a great boost in ride-sharing potential with or without time constraints. In the case of Madrid, `En-Route RS` increases the savings from 24% to 53%.

Then, since people are often hesitant to ride with strangers, we decided do add social constrains too. Social ties can be inferred from calls (CDRs), or declared friendship (Twitter). First, we consider ride-sharing only with one-hop friends. Then `En-Route RS` in the city of Madrid using CDR and Twitter friendship provides only a tiny traffic reduction of 1.1%

and 1.2% respectively. This dramatic decrease is attributed to the low density of the social graphs and to the fact that only a small portion of the graphs' nodes have known home/work addresses – each user has the opportunity to share a ride only with a small portion of her neighbors. However, if we relax the social constraints and permit ride-sharing with friends-of-friends, the ride-sharing potential increases significantly, especially in `En-Route RS`. The corresponding numbers are 19% and 8.2% for friendship based on CDRs and Twitter data, respectively. Furthermore, if we project the potential of ride-sharing to the total commuting population of the city (much larger than the number of users with inferred home/work locations), the benefit increases up to 31% for call based filtering and 26% for OSN based filtering.

Finally, we compared the four cities and observed that the population density of a city has a profound effect on its ride-sharing potential, especially when strict social filtering is applied. For example, BCN is denser and has a 14% higher ride-sharing potential than Madrid; LA, on the other hand, has 46% lower ride-sharing potential than NY.

3.3 A Scalable System for Online Ridesharing

3.3.1 Introduction

An anticipated breakthrough in ridesharing is the ability to satisfy on-demand requests that do not require participants to schedule their trips well in advance [50]. When asked how far ahead of time participants would like to organize a shared ride, 43% preferred to plan their ride 15-60 minutes before departure [78]. In this section, we are interested in such an *online ride-sharing (ORS) system*: requests arrive dynamically, possibly with a short notice. We develop an online ride sharing (*ORS*) system for matching users that could share a ride. We evaluate the performance of the system using spatio-temporal data from the city of New

York. The data set was extracted from geo-tagged tweets from the city of New York and contains more than 61K users. In the previous section, we used these datasets to extract commuting patterns: for each user, we have their home/work location, departure times and commuting route. These datasets showcase an important case of *ORS*: ride sharing among commuters between home and work; however, our framework can handle arbitrary driver trajectories and passenger locations.

We break *ORS* into two main components: the constraint satisfier and the matching module. The constraint satisfier, a.k.a *satisfier*, takes as input the itineraries and spatio-temporal constraints of drivers and passengers and provides feasible (driver, passenger) pairs. The core challenge in this module lies in its scalability. We achieve scalability by designing a constraint satisfier using a road networks data structure, specifically optimized for our spatio-temporal queries.

Our *satisfier* component is more scalable than what can be built using state-of-art off the shelf components: query time increases 4.65x slower with the number of users when compared to MongoDB. The key to higher performance is the use of the road network data structure; our specialized *satisfier* exploits the underlying road network of our spatio-temporal data, while MongoDB treats them as generic geo-coordinates.

We use the feasible pairs found by the *satisfier* to define a bipartite graph between possible drivers and passengers, with edge weights representing the length of the shared trip of a pair. The matching module takes as input the weighted bipartite graph and returns the maximum weighted matching (MWM), which captures the objective of real-world ridesharing systems (such as `Lyft Carpool`). We propose an efficient algorithm to solve the MWM problem, which is 51% better than greedy heuristics used by many real systems. Furthermore, the system is designed to handle efficiently requests that arrive on-line, via efficient queries of feasible pairs and incremental updates of the matching solution. The key to higher efficiency lies in its incremental updates: *Online MWM* uses augmenting methods methods to update

the existing matching and do global optimizations. We evaluate the entire ORS system using real mobile datasets to extract driver trajectories and passenger locations in urban environments. We show that *ORS* can provide a ridesharing recommendation to individual users with a sub-second query response time, even at high workloads. We also evaluate the sensitivity of *ORS* performance to various parameters, which provides insights for the design of practical ridesharing systems. Finally, by decoupling the system into two components that interact only through the (dynamically updated) bipartite graph of feasible pairs, we build a system that is modular.

3.3.2 Related Work

This section summarizes the most relevant work in the area of ride-sharing.

Commercial Ridesharing. Ridesharing is an important part of the sharing economy and provides economical, societal and environmental benefits by utilizing "empty" car seats. Relevant ride-sharing startups include *Zimride.com* and *Scoop* [13], whose focus is to facilitate ridesharing between employees of large corporations. This makes the problem less challenging than in the general case (and is thus a popular strategy for bootstrapping ridesharing [18]), since users have the same destination (the company they all work for) and the system considers only the home locations of drivers and passengers, thus reducing the number of potential pairs. Recently, Uber and Lyft, which are primarily on-demand taxi companies, announced ridesharing services for commuters: Uber announced *uberHOP* and *uberCOMMUTE* [15] and Lyft announced *Lyft Carpool* [14]. All the above services facilitate ridesharing among commuters, and the problems they address are the main focus of this section, but since these systems and their algorithms are proprietary, and therefore not publicly available, it is not possible to compare against them. However, they serve as a guideline for our *ORS*, and we formulate our ridesharing problem to be in line with the aforementioned systems.

On-demand Taxi-sharing: On-demand taxi services (e.g. main services of Uber and Lyft) use smartphone devices to schedule trips between drivers and passengers. They employ dedicated drivers who, unlike commuters, have no spatial or temporal constraints. Both of these companies offer cheaper versions of their on-demand taxi services that include ridesharing: Uber offers *uberPOOL* and Lyft offers *LyftLine*. These two services facilitates ridesharing for taxi passengers who ride along a similar route, and they do not takes into account spatial and temporal constraints for the driver. However, there is an one-to-one mapping from the on-demand taxi-sharing to the online ridesharing (between a drive and a passenger) problem. In *uberPOOL* and *LyftLine* the first passenger (primary passenger) decides the trajectory that the taxi will follow, while the extra passenger (the one that will be picked-up later) will be picked-up only if it is convenient for the first one. Therefore, the first passenger is equivalent to the driver, while the extra passenger is equivalent to the passenger of the online ridesharing problem. This means that our *ORS* can be used for this type of taxi-sharing services also. In general our *ORS* can be used to match trajectories (driver) with a pair of source and destination points (passenger), online and in real-time.

Academic Research provides valuable insights into ride-sharing, primarily through surveys on ride-sharing optimization [16] and small-scale ridesharing demos [78]. The work in [78] reports how far in advance ridesharing participants want to schedule their trips. Other studies characterized the behavior of carpoolers [84], identified the individuals who are most likely to carpool and explained what are the main factors that affect their decision [39]. Prior work quantified the potential of ridesharing [18, 34] using offline analysis of datasets. Taxi-sharing [61, 37] resembles online ridesharing, but since it does not consider spatial and temporal constraints for the passenger, requires different algorithms. Excellent surveys on the formulation and optimization of ridesharing and on the key computational challenges include [16] and [50]. The focus of this section is online ridesharing (as opposed to offline analysis of its potential of Sec. 3.2) and system design (design and evaluation of *ORS* that needs to run in real time).

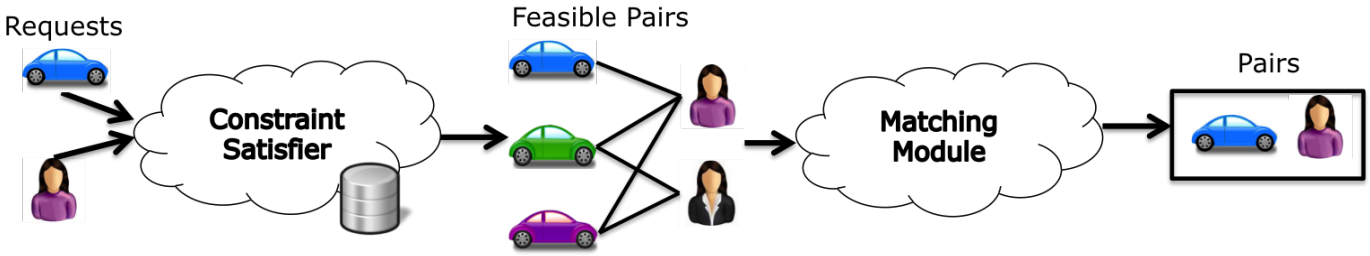


Figure 3.18: ORS System Overview. Drivers and passengers enter their requests. The "constraint satisfier module" finds candidate pairs and builds a bipartite graph. The matching module takes the bipartite graph as an input, produces a matching. Finally, drivers and passengers are notified.

Notation	Definition
$(lat_p^{(h)}, lng_p^{(h)})$	Source location for p.
$(lat_p^{(w)}, lng_p^{(w)})$	Destination location for p.
$t_p^{(h)}$	Earliest departure time for p.
$t_p^{(w)}$	Latest arrival time for p.
$t'^{(h)}_p$	Latest departure time for p.
$\Delta t_2 = t'^{(h)}_p - t_p^{(h)}$	Delay Tolerance of p.
$t_p^{(r)}$	When p sends his request.
$\Delta t_1 = t_p^{(h)} - t_p^{(r)}$	*ahead-of-time notification*
δ	Distance Tolerance.
T_d	Trajectory for driver d.
$(lat_d^{(0)}, lng_d^{(0)}, t_d^{(0)})$	Starting point of d's trajectory.
$(lat_d^{(n_d)}, lng_d^{(n_d)}, t_d^{(n_d)})$	Ending point of d's trajectory.

Table 3.8: Notations for passenger p and driver d.

3.3.3 System Overview

System Requirements

We define ridesharing as a one-time trip shared between a driver and a passenger, according to spatio-temporal constraints that both parties specify. The driver has a trajectory and the passenger has a source and a destination (a.k.a home/work). The drive can pick-up the passenger en-route (e.g. half-ways). In the case of a commuters' ridesharing system, such as *Scoop*, the driver is the commuter who will use his car, while the passenger is the commuter

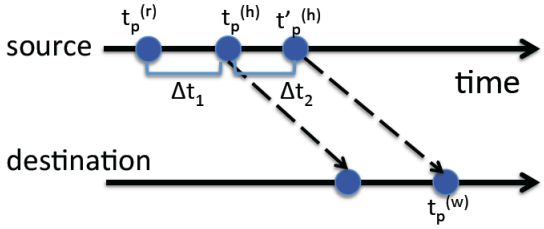


Figure 3.19: **Request by passenger p:** The vertical axis represents space (source and destination locations), the horizontal axis represents time and the dashed lines are example trajectories. At $t_p^{(r)}$ the passenger sends his request in the system. The passenger is ready to leave from the source location at $t_p^{(h)}$ and wants to arrive at the destination by $t_p^{(w)}$; $t'^{(h)}_p$ is the latest time the passenger can leave and not to late. We refer to $\Delta t_1 = t_p^{(h)} - t_p^{(r)}$ as *ahead-of-time notification* and $\Delta t_2 = t'^{(h)}_p - t_p^{(h)}$ as *delay tolerance* (how much can p wait to be picked up).

who will leave his car at home. While in the case of *UberPool* or *LyftLine* the driver is the first passenger (primary passenger) that defines the route of the car, while the passenger is the extra person who will be pick-up as long as the deviation is convenient for the first one. Thus, our definition of ride-sharing is broad enough to contain both applications for commuters or taxi-sharing applications; in both cases we want to match a trajectory with a (source, destination) pair such that certain spatiotemporal constraints are satisfied. Finally, the longer the shared part of the trip the better [3].

Drivers and passengers submit their requests before their desired departure time; this *ahead-of-time notification* can be, for example, a few minutes before departure or the evening before the trip, and in general a parameter that affects performance. When ride requests are submitted, a search for potential matches takes place in real time. If a suitable match is found, the participants are notified immediately. An overview of the system is shown on Fig.3.18.

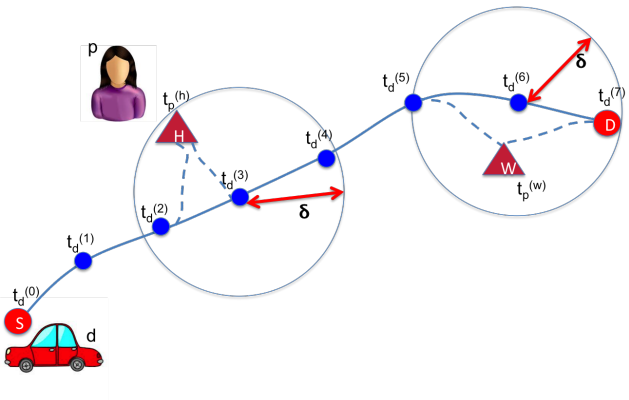


Figure 3.20: Example of spatio-temporal constraints when matching a passenger p with a driver d. The passenger p leaves from her source location (home H) and is going to a destination (work W). The driver has a fixed trajectory (indicated in solid line) and departure time. However, the driver considers deviating at different points on his trajectory and do a detour (indicated in dashed line) to pickup and droppoff the passenger, as long as these detour distances do not exceed his *distance tolerance*: *i.e.*, $dist_H(p, d, i) \leq \delta$ and $dist_W(p, d, j) \leq \delta$. In exchange, the passenger may wait until his latest departure time.

Notation

Let S denote the set of all users, D denote the set of drivers and P denote the set of passengers. Clearly $D \subseteq S$, $P \subseteq S$ and $D \cup P = S$. A location is described with its coordinates (lat, lng). For every passenger $p \in P$, the request entered in the system consists of the following information, also depicted on Fig.3.19. For every driver $d \in D$, the request entered in the system consists of the following information, also depicted on Fig. 3.20. Drivers' inputs, passengers' inputs, and ridesharing parameters are explained in Tab. 3.8.

Driver and Passenger Constraints

Ride-sharing needs to be convenient for both the driver and the passenger: they shouldn't deviate too much from their routine and they shouldn't experience excessive delay or inconvenience.

[3]The passenger will pay the driver and the system (that enables ridesharing) will keep a percentage of the payment. Therefore, a ridesharing system would want long joint traveled distances.

Let us consider a given driver-passenger pair, d and p, depicted on Fig. 3.20. We assume that the driver does not change trajectory or departure time; however, he is willing to do a small detour to pickup drop off the passenger, as long as that detour does not exceed his *distance tolerance* δ. Let i be the pick up (*i.e.* closest point of d's trajectory to the home of p), and j be the drop off (*i.e.* closest point of d's trajectory to the work of p) location, and $i < j$. The passenger conveniently gets a ride, picked up at his source (*e.g.* home H) and dropped off at his destination (*e.g.* work W) location. In exchange, he may have to wait and delay his departure up to his latest departure time $t'^{(h)}_p$ in order to arrive by the latest arrival time $t^{(w)}_p$. Let $delay_H(p, d, i)$ and $delay_W(p, d, j)$ be the pick-up and drop-off delays respectively.[4] A pair $(d, p) \in E$ is feasible if both the passenger and driver constraints are satisfied, *i.e.*:

$$w(d,p) = \begin{cases} dist(i,j), \quad \text{if} \quad t^{(h)}_p > t^{(i)}_d + delay_H(p,d,j) \\ \quad \text{and} \quad t^{(w)}_p < t^{(j)}_d + delay_W(p,d,j) \\ \quad \text{and} \quad max(dist_H(p,d,i), dist_W(p,d,j)) < \delta \\ 0, \quad \text{otherwise} \end{cases}$$

A core challenge in ride sharing is to find feasible passenger-driver pairs and points for pick-up drop-off on the driver's trajectory, that meet all constraints. The response to such search queries must be fast, for the ride-sharing system to be real-time and scale with the number of users. We discuss the constraint satisfier in a following section.

[4]We note that the spatial and temporal constraints are related: the pick-up and drop-off distances result in pick-up and drop-off delays, but we choose to account for both of them separately, because the distance constraints capture additional inconvenience beyond delay.

Matching Drivers and Passengers

Multiple driver-passenger pairs can satisfy the spatial-temporal constraints discussed above. Consider the bipartite graph depicted in the middle of Fig. 3.18. The nodes on the left represent the drivers, the nodes on the right represent the passengers. There is an edge between a driver and a passenger iff having them share a ride is feasible (within the spatiotemporal constraints). The optimization problem that lies at the heart of ride-sharing is how to find a matching between drivers and passengers. There can be many optimization objectives; for an excellent survey on this aspect of the problem please see [16]. In this section, we aim at maximizing the sum of all joint traveled distances. This is the best case scenario for the ridesharing enabler (the system that offers rideshare matching).

In Section 3.3.5, we formulate the problem as maximum weighted matching and we describe an efficient algorithm to solve it. There are two requirements for the matching algorithm. First, it must be *efficient and scale well* with the number of users, in order to support a real-time, large-scale system. Second, the algorithm must be able to perform *incremental updates* due to the online nature of the problem, *i.e.* to support the arrival and expiration of users' requests and to handle updates as the users move and update the time they are at particular locations. Incremental graph processing is a challenging problem, in general.

System Architecture

Fig. 3.18 shows an overview of the architecture of the system, which consists of two main components. The first component is the **constraint satisfier**, whose input are passengers' and drivers' requests and produces feasible passenger-driver pairs, which can be summarized in the bipartite graph, shown in the middle of Fig. 3.18. The second component is the **matching module**: this takes as input the bipartite graph of feasible pairs and finds a maximum weighted matching.

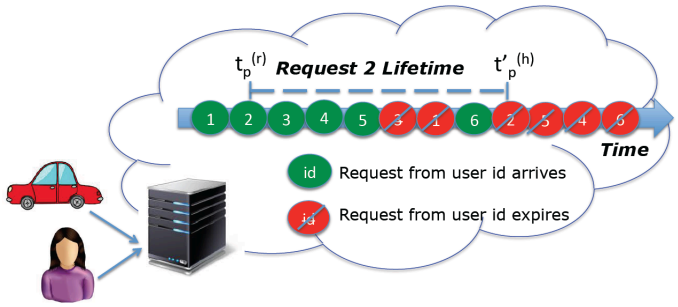


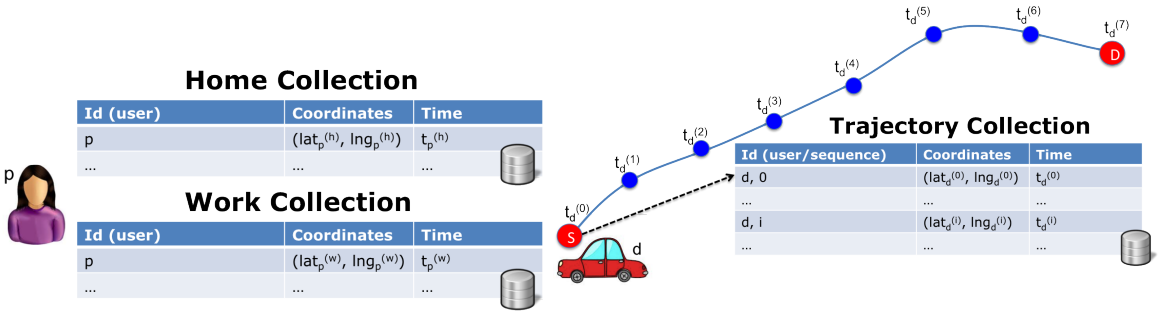
Figure 3.21: Online Ridesharing. The server sees requests from users arriving and expiring. For a driver d, a request arrives at $t_d^{(r)}$ and expires when the driver arrives at the destination. For a passenger p, a request arrives at $t_p^{(r)}$ and expires at the latest departure time $t'_p(h)$.

An important aspect of our system is that it is **online**: requests from drivers and passengers can arrive dynamically (at times $t_d^{(r)}, t_r^{(r)}$, respectively) and also can expire (when a driver arrives at the destination $t_d^{(n_d)}$, or after the latest departure time of a passenger $t'_p(h)$). When arrival/expiration events happen, the two modules need to do incremental updates. More specifically, the first module needs to update the records in the database (driver's trajectory points and passengers' source, destination and constraints) and the bipartite graph of feasible pairs. The second module needs to update the matching solution, based on the changes in the bipartite graph. Another type of event that causes updates is errors in predicting the time estimates for different points of a trajectory: the actual times are updated as a driver moves.

The decomposition of the problem into two parts, is key for enabling a modular, fast, online, optimal system. More details are provided in the following sections.

3.3.4 Constraint Satisfier

This component receives the queries from the drivers and the passengers, and will generate the feasible matching combinations, *i.e.* for every passenger, it will generate the set of drivers who meet her spatio-temporal constraints. The constraint satisfier needs to support



(a) Storing Home/Work (b) Storing trajectories

Figure 3.22: Storing trajectories and Home/Work information in the MongoDB based constraint satisfier.

fast insertion queries (*e.g.* when new requests arrive), and fast searches queries (when looking to match similar passengers and drivers) in real time. Also, it needs to handle the spatial-temporal information of the trajectories and it needs to scale to tens of thousands of users per hour [5].

satisfierDB: Constraint Satisfier with off the shelf Components

We initially build the constraint satisfier using of-the-shelf components that we call *satisfierDB*. We choose MongoDB because speed and scalability are our primary concerns, and we do not require complex join queries that relational databases offer. Moreover, *MongoDB* [4] is very popular in the industry (*e.g.* used by Foursquare [5]). MongoDB is a document-oriented database that stores data in collections made out of individual documents; each document is big JSON file with no particular format or schema. Finally, MongoDB supports spatio-temporal proximity, which is known to be fast and accurate.

• *Spatial Indexing:* MongoDB offers supports for queries that calculate geometries on an earth-like sphere, through the *2dsphere index* – a grid based geohashing scheme [1].

[5]When asked how far ahead of time participants would like to organize a shared ride, 43% responded 15-60 minutes before the targeted departure [78]. For a population of 500K commuters, a ride-sharing system would have to match up to 215K such users. Also, assuming a 25% market penetration – Uber's market penetration in San Francisco [11] – a ride-sharing system has to match more than 54K users within an hour.

• *Fast data Insertion:* MongoDB is designed for fast insertion speed; it employs the cache of the operating system, which significantly reduces write costs. Fast insertion of data is very important for a real-time system, since while data are being inserted in the database, the process that is doing the insertion will lock the data – via a write lock – and during that time no other process can read or write anything. This means that one cannot take advantage of parallelization of insertion queries (which can be easily done for read queries, because they use a read-lock that allows other processes to read from the dataset).

Implementation Details:

In order to take advantage of the *2dsphere* and the proximity queries that come with it, we build and store our data as ¡latitude, longitude¿ points – both for coordinates that represent source/destination points and coordinates that represent points in a route.

Home/Work Collections: We store the data of the passengers in two collections: 1) Home collection for the source points, and 2) Work collection for the destination points (see Fig. 3.22(a)).

Trajectory Collection: We store the trajectory points of the driver in a different Trajectory collection. Each trajectory point is an individual entry in the collection, indexed by the id of the user and its position in the trajectory (*e.g.* first, second, third, ..., or last point) (see Fig. 3.22(b)).

Spatio-temporal queries: When we query one of the collections (either a driver seeking for passengers using the Home and Work collections, or a passenger seeking for drivers using the Trajectory collection), we combine the spatial and the temporal range queries via the $*near* operator), using the $*and* operator.

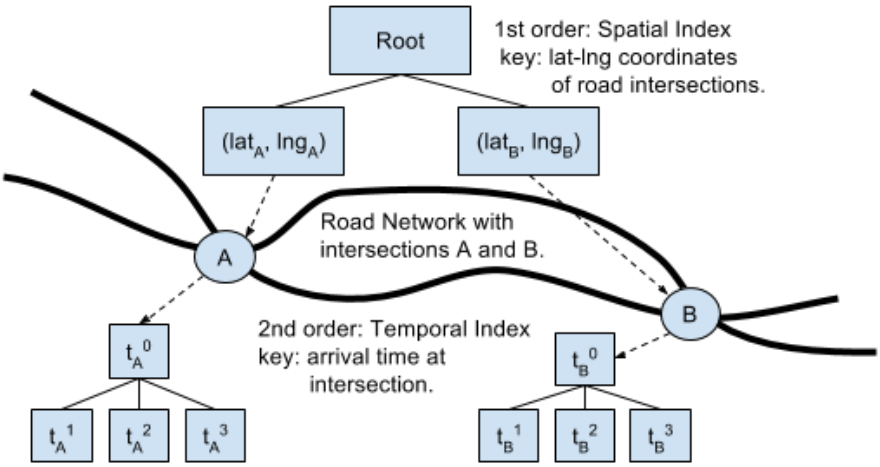


Figure 3.23: Storing spatio-temporal data in the *satisfier*. The intersections of the road network are stored in a tree data structure for fast accessing. That is the 1^{st} order data structure that stores intersections using as keys their (lat, lng) coordinates. Each key is paired with a value, and the values (of the 1^{st} order data structure) are symbol tables that contain $< key, value >$ pairs, refereed to as the 2^{nd} order data structure. In the 2^{nd} order data structure the keys are the times when user appear at the specific intersection and as a value the id of the user.

satisfier: Our Specialized Constraint Satisfier

Next, we build a specialized constrain satisfier, which we call *satisfier*, that is tailored to the needs of our problem and it takes into account the unique characteristics of our data. More specifically, our specialized system takes into account the underlying road network of the trajectories (see Fig. 3.23). Our *satisfier* is inspired by state of the art spatio-temporal data structures [46, 66].

• *Road Network:* The *satisfier* uses a road network to store the (lat, lng) coordinates of the users. The road network contains intersections, in the form of (lat, lng) coordinates. The (lat, lng) coordinates are stored in a tree structure (1^{st} order structure in Fig. 3.23) with the coordinates being the keys. The values that the keys (coordinates) are paired with are symbol tables that contain $< key, value >$ pairs (2^{nd} order structure in Fig. 3.23). In the 2^{nd} order data structure the keys are the times when user appear at the specific intersection and as a value the id of the user. When a new driver request arrives, the (lat, lng) points of

her trajectory are mapped to their closest intersections; from there (the closest intersections) they are mapped to the 2^{nd} order structure (a $< key, value >$ symbol tables) where they are stored based on their times (of arrival at the intersections). When a passenger wants to find the driver that can pick her up, the *satisfier* (1) will look at the intersections that are within δ of her home, and for each one of this intersections (2) it will get the drivers that are within time constraints; then, the *satisfier* (3) will do the same for the work locations, and (4) the drivers that can both pick-up and drop-off the passenger will be returned. A similar procedure will be followed for when a driver is trying to find passengers; for each one of the points of her trajectory the *satisfier* will find the passengers that are withing spatio-temporal constraints and can be both picked-up and dropped-off.

• *Implementation Details:* In our implementation we used KDTree for the 1^{st} order structure and MultiMaps[6] for the 2^{nd} order structure. A KDTree is a space-partitioning data structure for organizing points in a k-dimensional space; for our (lat, lng) coordinates k is 2. We use road intersections, instead of road segments due to practical reasons: our road network (which will be described in the Evaluation Section) is dense and the distances between them are short [12] [7]. Moreover, our (1^{st} order structure is a static road network and we don't have to update or delete nodes; all nodes are added when a priori. We use MultiMaps for the 2^{nd} order data structure structure we want it to be dynamic and support fast add/remove operations.

• *Additional data structures:* We want to be able to remove users once their request has expired. We achieve fast removal of users through hash tables with the user id as the key and their stored location as the value (intersection where the drivers have been stores and their times).

[6] A Hash Table where the same key can map to multiple user ids. To be more specific a key maps to a set of users.

[7] For a sparser road network large distances between intersections, using road segments and an R-Tree would be better.

3.3.5 Matching Drivers-Passengers

In this section, we focus on the matching module that takes as input the bipartite graph of feasible driver-passenger pairs and provides a matching. We formulate the problem as maximum weighted matching (MWM), and we provide and algorithm (referred to as *Online MWM*) that is efficient, and online (*i.e.* it continuously updates the matching in the presence of arrival/expiration of requests). As a baseline for comparison, we compare it to *0-1 Algorithm*, described in [35] and the offline solution of MWM.

Maximum Weighted Matching

Consider the bipartite graph of feasible pairs: $G = (D \cup P, E, W)$ where $E = \{(d, p) : d \in D, p \in P\}$ s.t. that the constraints of d, p, as defined in a previous section, are satisfied. Each edge $(d, p) \in E$ is associated with a weight $W(d, p)$ indicating the length of the shared trip between the driver and the passenger. Finding the Maximum Weighted Matching (MWM) on this bipartite graph is a classic problem that can be solved efficiently (in $O(|V|^2 \cdot |E|)$ time) and optimally using the Hungarian algorithm [71]. However, there are not online optimal algorithms for solving the MWM for the bipartite graph in real-time. Finally, as it will be described in the next section, when the bipartite graph is very dense finding the optimal solution takes a lot of time. In our case, the bipartite graph that we try to solve is in the order of tens of thousands of nodes and millions of edges [8].

Insights from the Data

When comparing the offline solution of the bipartite graph (which took days to produce) with all the graph weights (see Fig. 3.24) we observe that the large weights appear more

[8] We used the popular https://networkx.github.io/ Python package and we were able to find the optimal solution in more than two days.

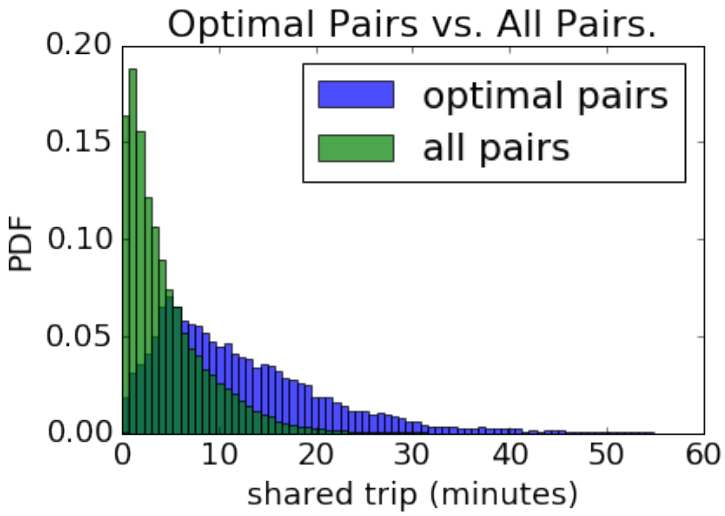


Figure 3.24: Comparison of weights for all pairs, and the weights in the offline solution which contains the optimal pairs. The figure shows that the weights with higher weights have more chances of being in the final solution.

often in the optimal solution than the small weights. Therefore, from the millions of weights of the graph, the vast majority of them does not appear in the optimal solution and their presence makes the solution of the problem harder and slower.

Online MWM

Our *Online MWM* algorithm, used for online matching is based on Max Cardinality Matching (MCM). MCM can be solved efficiently and very fast; our bipartite graph that has tens of thousands of nodes and a few million edges can be solved in seconds. Finding the Maximum Cardinality Matching (MCM) on this bipartite graph is a classic problem that can be solved efficiently (in $O(min(|D|, |P|) \cdot |E|)$ time) and optimally using augmenting paths [71] [9]. This classic (Ford-Fulkerson) algorithm lends itself naturally to an online version that can handle arrivals and departures of requests. Indeed, arrival of driver/passenger requests

[9]The algorithm solves a max flow problem between a super source and super receiver node and the bipartite graph in the middle with unit capacities. A residual graph is used to store and search for augmenting paths; when BFS on the residual graph cannot find augmenting paths, a max-flow is found on the original graph. The max-flow solution then corresponds to a matching (using only the edges with flow 1).

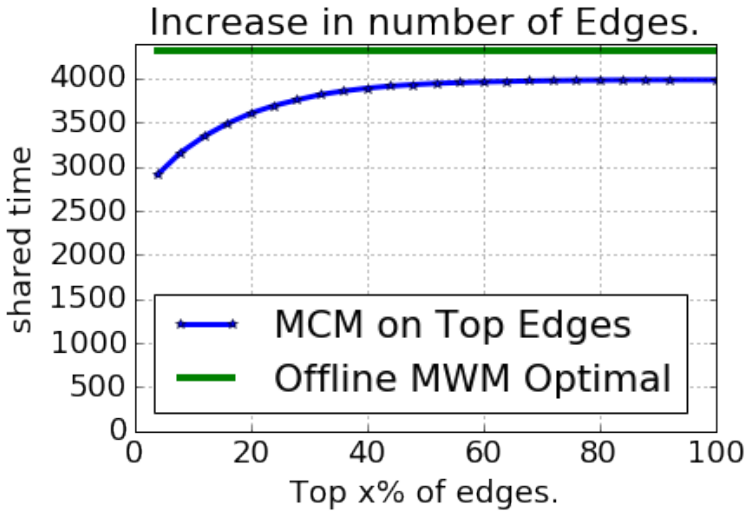


Figure 3.25: This figures shows how well our MCM-based algorithm approximate the MWM offline. *Online MWM* breaks the graph into multiple sub-graphs based on the weight of their edges, e.g. sub-graph-1 contains the top 1% of the edges, while sub-graph-2 contains the top 2% of the edges, e.t.c and then it applies an augmenting path algorithm to find the MCM in each sub-graph. The combined solution of all sub-graphs is the approximation solution to the MWM problem.

lead to edges appearing/disappearing from the bipartite graph, which can be be handled by efficient incremental updates. Indeed, every time a request arrives, this results in one or more edges appearing in the bipartite graph. All we need to do is to find an augmenting path in the new auxiliary graph and update the existing matching (in $O(|E|)$), as opposed to solving the problem from scratch.

We break the bipartite graph in multiple graphs, where each graph contains all the edges from certain level and above. We assume that we know the separation levels [10]. Then, we find the MCM of each graph, starting from the one with the largest weights on the edges. Alg. 1 is an augmenting path algorithm designed to find the online MCM for each sub-graph. Finally, Fig. 3.25 shows how our *Online MWM* compares to the offline MWM solution.

[10]This is a realistic assumptions since the levels can be learned from the data, e.g. from yesterday's requests to the ridesharing system.

Algorithm 1 updateGraph

```
1: // Global structures maintained
2: biGraph: bipartite graph up to now
3: auxGraph: auxiliary graph with augmented paths
4: pairs: pairs of (d, p) matched so far
5: when new request arrives from a user // (d or p)
6:    if passenger p then
7:       // Find the drivers within-constraints
8:       feasibleDrivers:= getDrivers(p)
9:       biGraph:= updateGraph(p, feasibleDrivers)
10:   end if
11:   if driver d then
12:      // Find the passengers within-constraints
13:      feasiblePassengers:= getPasngr(d)
14:      biGraph:= updateGraph(d, feasiblePassengers)
15:   end if
16:   auxGraph:= updateAuxiliary(auxGraph, biGraph)
17:   pairs:= maxBipartiteMatching(pairs, biGraph, auxGraph)
18: end when
```

Greedy Algorithm

Many on-demand taxi services have emerged recently, which act as a broker between a taxi and a passenger, the primary example being Uber. These companies use proprietary matching algorithms, which are however widely believed to be simpler: *e.g.* typically match a passenger "greedily" with the closest driver. As a baseline for comparison, we define a *Greedy Algorithm* as follows: eash request it will be matched it its best available choice, at the time of its arrival, e.g. a passenger is matched to the best unmatched driver at the time her request arrives in the system. We are interested in understanding how our global optimization (*Online MWM*) compares to these simpler, greedy heuristics, in terms of the global objective (*i.e.* total joint distance traveled).

City	NY
Users	61K
Inter-point distance	100m
Average distance	16.1 km
Median distance	8.0 km
Average gps points per trajectory	78.8

Table 3.9: Data set summary

3.3.6 Evaluation

In this section, we use the previously described datasets to evaluate our ridesharing system. We first evaluate the system as a whole, and then we evaluate each component separately. More specifically, we evaluate the *Online MWM* in terms of the global objective (matching rations and sum of total shared trips). Then, we compare *satisfier*, which is our specialized constraint satisfier, with *satisfierDB* which is the constraint satisfier build with off-the-shelf components.

Experimental Setup

In order to assess the performance of our on-line ridesharing algorithm using real data we use the NY data, extracted in Sec. 3.2.3, that contains the home and work locations, as well as trajectories, of a large number of users (see Tab. 3.9).

Given the home/work locations of the users, their departure times and the drivers' routes, we compute the performance of the algorithm for different values of *ahead-of-time notification*. We do a discrete time simulation where all the events appear in a simulated time-line based on the order of their arrival. In our simulation, there are two types of events :(1) *request arrival* – a new users send a request to the system, (2) *request expiration* – the simulated time reach the latest time when the users has to go. We process the events in the order they appear on the timeline. When a request arrives, it creates a new node with attached edges

to the feasible users. A request-expiration event will cause a node and its attached edges to be deleted. We show the speed of our system, as well as the matching ratio and the sum of all shared trips (total shared trips). Our simulations are using the following spatio-temporal constraints: (i) we apply a spatial constraint of 1 km $\delta = 1$ km and (ii) a delay tolerance of 10 minutes $\Delta t_1 = 10$. Finally, we run our experiments using a Linux machine with an Intel Core i5-5300U processor and 20GB of memory.

Road Network:

Our *satisfier* uses the NY road network from [12]; the road networked is represented by a directed graph. We denote $G_{rd} = (V, E, W)$, where V is the set of intersections[11] and $E = \{(u, v) : u \in V, v \in V\}$ the set of edges; an edge (u, v) indicates that there is a road segment connecting nodes u and v. Each edge $(u, v) \in E$ is associated with a weight $W(u, v)$ indicating the required time to go from u to v. G_{rd} has 264K nodes and 734K edges.

Arrival Times:

As denoted earlier, a driver's trajectory, lets call her d, is described by a sequence of spatio-temporal points: $T_d = \{(lat_d^{(0)}, lng_d^{(0)}, t_d^{(0)}), ..., (lat_d^{(n_d)}, lng_d^{(n_d)}, t_d^{(n_d)})\}$ (see Tab. 3.8). The (lat_d, lng_d) coordinates of d's trajectory derives from her commuting route, described in the previous sections. The temporal part of d's trajectory, which represent when d arrives at the specific (lat_d, lng_d) point, was computed using the road network; d's coordinates were mapped to their closest intersection in G_{rd}, and the weights between G_{rd}'s edges were used to generate the arrival times. Similarly, G_{rd} was also used to generate passengers' arrival times.

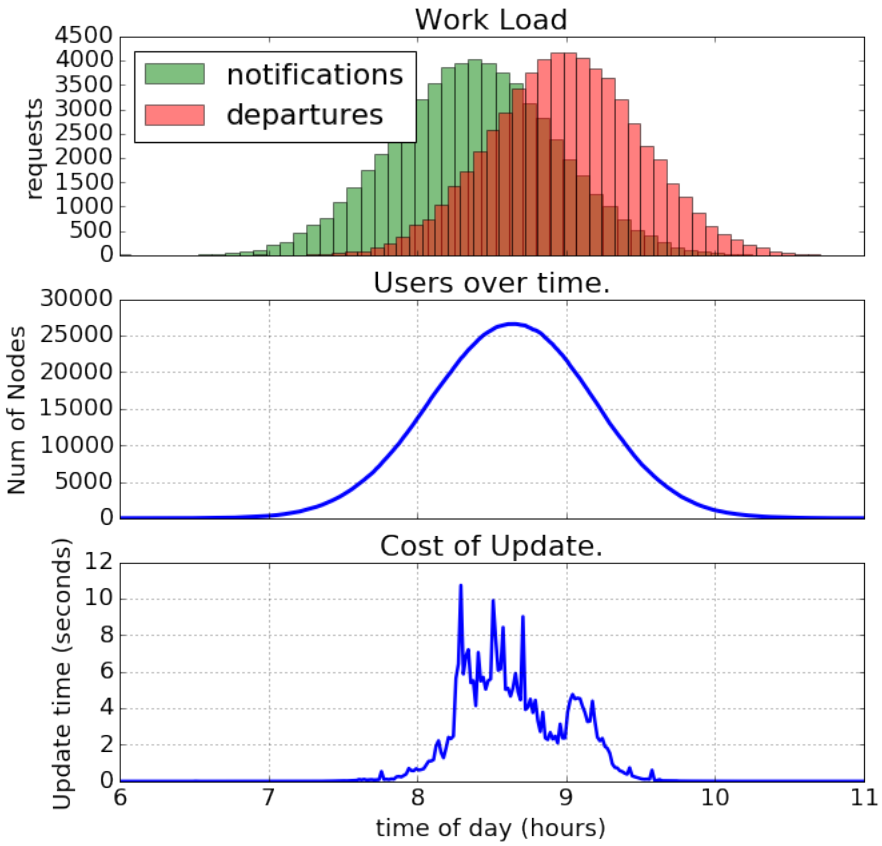


Figure 3.26: The expected work-load from the system, and the end-to-end experiment of the system. We assume that the requests will arrive randomly before the departure. The probabilistic distribution to generate how long before departure users will notify the system is a uniform distribution in range [15 minutes, 60 minutes]. According to the figure, the highest stretch for our system occurs when the number of users (or nodes in the bipartite graph) is at its pick. At that time each update – that contains multiple new request – can required up to 12 seconds.

	Shared Trip Sum	Comparison to Offline MWM (%)
Offline MWM	4320	–
Online MWM	3957	−8.4
0-1 Algorithm	3460	−20
Greedy Algorithm	2623	−39

Table 3.10: Offline Result Comparison for a graph with size 61K nodes (out of which only 48054 had a neighbor), and 1.5M edges. *Online MWM* is 14.4% better than *0-1 Algorithm* and 51% better than *Greedy Algorithm*.

End-to-End Experiment

In Fig. 3.26 you can see the end-to-end experiment. According to the figure, the highest stretch for our system occurs when the number of users (or nodes in the bipartite graph) is

[11]Latitude and longitude coordinates of the real-world road intersections in NY.

at its pick. At that time each update – that contains multiple new request – can required up to 12 seconds. When compared to the offline optimal, which takes more than two days to compute, *Online MWM* is 8.4% worse, but it's 14.4% better than *0-1 Algorithm* (see Tab. 3.10).

Breaking up Components

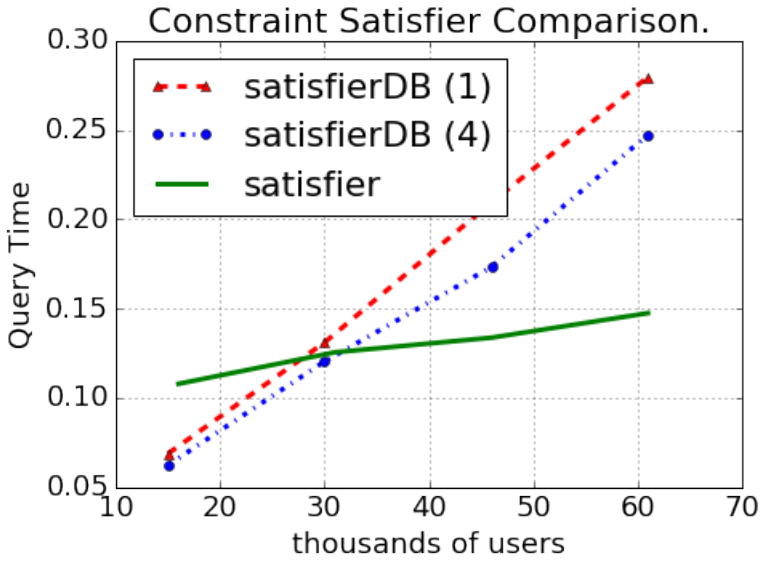
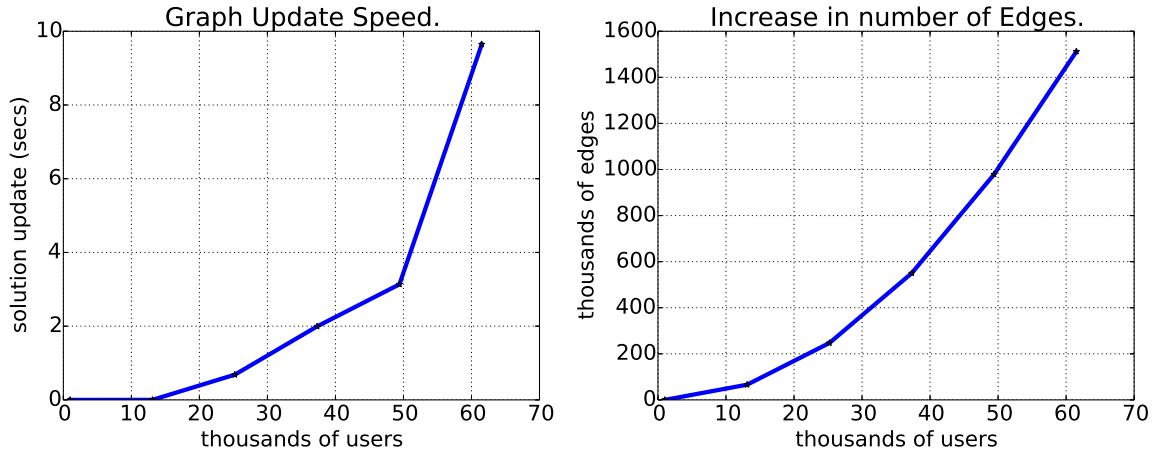


Figure 3.27: Comparing the scalability of *satisfier* and *satisfierDB*. Both *satisfier* and *satisfierDB* scale linearly, but the slope of the *satisfierDB* in (fig:query-speed) is 4.65 times greater than the slope of *satisfier*. Therefore, we say that the specialized *satisfier* is 4.65 times more scalable when compared to the *satisfierDB*. When there are a few users *satisfierDB* tends to be faster; this happens because *satisfier* is using a road network, with hundred of thousands of intersection, and when the number of users is low all this intersections are a burden. However, as the number of users grows the query time of *satisfier* grows at a much slower rate than *satisfierDB*; the higher complexity of better design of *satisfier* pays of as the number of users keeps growing.

We now do separate measurements for the two main components of the ridesharing system: (1) *satisfier*, and (2) matching module.

• *satisfier*: Fig. 3.27 show the query speed of three different implementations of the constraint satisfier: (1) *satisfierDB* with one process, (2) *satisfierDB* with four parallel process, and (3)

the specialized *satisfier*. We see that parallelization can improve the speed of *satisfierDB*, but the *satisfier* is still faster and much more scalable. Both *satisfier* and *satisfierDB* scale linearly, but the slope of the *satisfierDB* in (fig:query-speed) is 4.65 times greater than the slope of *satisfier*. Therefore, we can say that the specialized *satisfier* is 4.65 times more scalable when compared to the *satisfierDB* that is build using off the shelf components.



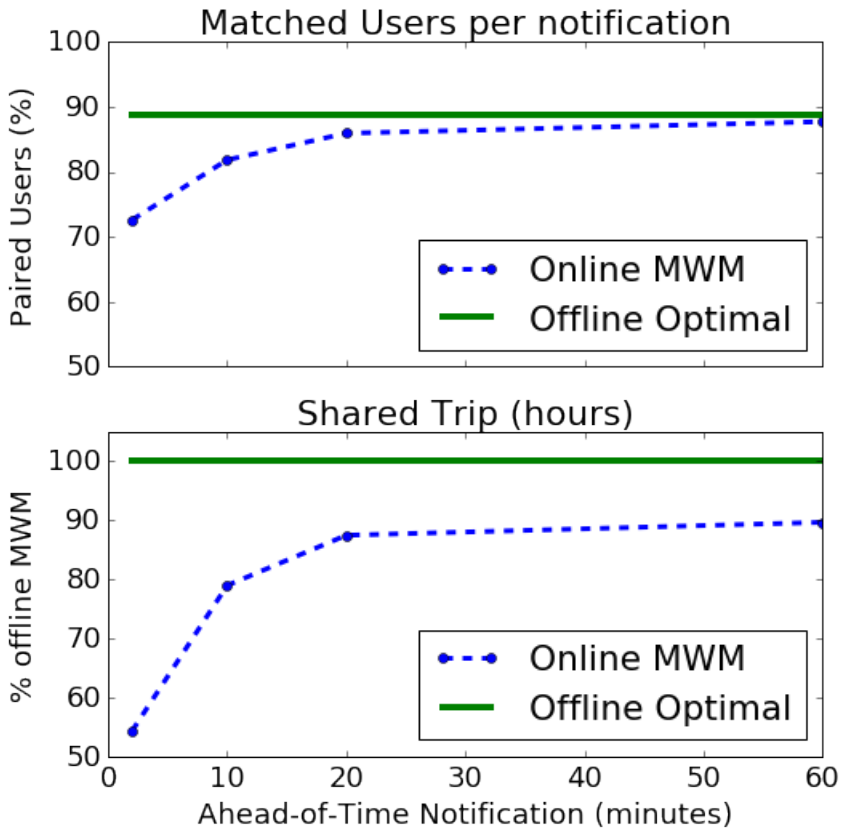
(a) (b)

Figure 3.28: The speed for the graph updates. As expected the speed of graph updates is affected primarily by the number of edges. We see that the number of edges increases quadratically to the number of users. The graph update speed increases quadratically too.

• Graph Update Speed: Fig. 3.28 show the speed of graph updates. As expected the speed of graph updates is affected primarily by the number of edges. We see that the number of edges increases quadratically to the number of users. The graph update speed increases quadratically too. This is because the augmented path algorithms, that we used to do the incremental updates has a time of (in $O(|E|)$).

The Importance of Early Notifications

In Fig. 3.29 we show how important the ahead-of-time notification is for ridesharing system. With an *ahead-of-time notification* of 2 minutes the *Online MWM* is at around 54% of the offline optimal solution, while with an *ahead-of-time notification* of 10 minutes is at close to



(a) Matching Ratios

Figure 3.29: This figures shows how the ahead of time notification affects the number of matched pairs, and the total shared trip. The x-axis shows the ahead-of-time notification, while the y-axis shows the comparison with the offline mwm.

80% of the optimal solution. This shows that *ahead-of-time notification* is very important for approaching the offline optimal result.

We hope that this result will give insight to companies, such as Uber and Lyft, on how to improve the performance of their ridsharing applications. Ridesharing companies need to incentivize their users to send their request way before their departure e.g. 10 or 20 minutes before. Apart from incentivizing users, they can design their products in such as way as to increase the *ahead-of-time notification*. For example UberPool requires that passengers to walk to a rendezvous point in order to get picked-up. By making the passengers walk more [12] they can increase the *ahead-of-time notification* artificially and thus have more time to

[12]They can assign the rendezvous point so that the passenger has to walk 5 or 10 minutes.

improve their matching; if the passengers have to walk they will perceive less delay [8].

3.3.7 Summary

Ridesharing has a great potential for reducing the number of cars in the streets of a city according to recent studies [34]. An anticipated breakthrough in ride-sharing is the ability to satisfy on-demand requests that do not require participants to schedule their trips in advance [50]. Such an online ride sharing system will provide a participant the reassurance that they would still be serviced if their travel-needs change unexpectedly; when asked how far ahead of time participants would like to organize a shared ride, 43% desired organizing their ride 15-60 minutes before departure [78]. In the last couple of years a plethora or smartphone-based ridesharing systems has emerged, such as *Scoop*, *uberHOP*, *uberCOMMUTE* and *Lyft Carpool*. These systems enable ridesharing for commuters and, in contrast to taxi-sharing, they consider spatio-temporal constraints both for the passenger and the driver. Inspired by the aforementioned academic studies and the real-world ride sharing system, we design and evaluate an *online ride-sharing (ORS) system* that handles dynamic ridesharing requests, possibly with a short notice. The evaluation is done using spatio-temporal data from New York.

We break *ORS* into two main components: the constraint satisfier and the matching module. The constraint satisfier, a.k.a *satisfier*, takes as input the itineraries and spatio-temporal constraints of drivers and passengers and provides feasible (driver, passenger) pairs. We achieve scalability by designing a constraint satisfier using a road networks data structure, specifically optimized for our spatio-temporal queries. Our *satisfier* system is more scalable than what can be built using state-of-art off the shelf components: query time increases 4.65x slower with the number of users when compared to MongoDB.

We use the feasible pairs found by *satisfier* to define a bipartite graph between possible

drivers and passengers, with edge weights representing the length of the shared trip of a pair. The matching module takes as input the weighted bipartite graph and returns the maximum weighted matching (MWM), which captures the objective of real-world ridesharing systems (such as `Lyft Carpool`). We propose an efficient algorithm to solve the MWM problem, which is 51% better than greedy heuristics used by many real systems. Furthermore, the system is designed to handle efficiently requests that arrive on-line, via efficient queries of feasible pairs and incremental updates of the matching solution. We evaluate the entire *ORS* system using real mobile datasets to extract driver trajectories and passenger locations in urban environments. We show that *ORS* can provide a ridesharing recommendation to individual users with a sub-second query response time, even at high workloads. We also evaluate the sensitivity of *ORS* performance to various parameters, which provides insights for the design of practical ridesharing systems.

Chapter 4

Conclusion

Thanks to the pervasiveness of smartphones and their applications there is an abundance of data generated from mobile devices, that reflect underlying human activitys and provide rich information about spatio-temporal and social patterns. This can be used to enable a variety of new services. We focus on how to mine mobile phone data to improve a variety of Smart City applications. In particular, we focus on Call Description Records (CDRs) that are generated every time a user makes a phone call. In this dissertation we use their rich and unique combination of insights into human dynamics in order to: (1) understand and improve transportation in a city via ridesharing, (2) characterize various areas of the city, as well as how these areas interact with each other (Urban Ecology), and (3) predict communication between different areas of the city.

In Chapter 2, we focus on Aggregate CDRs. First, we studied the decomposition of cell phone activity series, via FFT, into two series: (1) the seasonal communication series (SCS) produced from high-amplitude frequencies, and (2) the residual communication series (RCS) produced after subtracting SCS from the original series. As shown, the SCS can be used to characterize typical patterns of socio-economic activity within an area, while the RCS can

be used to capture both irregularities due to novel events and the influence of one area on another. The RCS and SCS thus provide distinct probes into the structure and dynamics of the urban environment, both of which can be obtained from the same underlying data. Then, we applied machine learning techniques to predict cell-to-cell activity, based solely on past cellular activity records. We were able to achieve 85% accuracy and 94% recall, for the voice call data set provided by Telecom Italia for the city of Milan.

In Chapter 3, we use the spatio-temporal and social information, per individual user, in CDRs in order to understand and improve transportation in a city via ridesharing. First, we used mobile and social data to demonstrate that there is significant overlap in people's commute in a city, which indicates a high potential benefit from ridesharing systems. This is clearly an upper bound to any practical ridesharing system, but the positive result motivates the deployment of such systems and policies. Our results indicate that en-route ridesharing with up to two-hop social contacts offers a good trade-off between technological feasibility, people's security concerns, and a substantial impact on traffic reduction. Then, we design and evaluate an *online ridesharing (ORS) system* that handles online ridesharing requests, possibly with a short notice. We break *ORS* into two main components: the constraint satisfier and the matching module. The constraint satisfier, a.k.a *satisfier*, takes as input the itineraries and spatio-temporal constraints of drivers and passengers and provides feasible (driver, passenger) pairs. The matching module takes as input the feasible (driver, passenger) pairs and returns the maximum weighted matching (MWM), which captures the objective of real-world ridesharing systems (such as `Lyft Carpool`). We propose an efficient algorithm to solve the MWM problem, which is 51% better than greedy heuristics used by many real systems.

Bibliography

[1] Geospatial indexes in mongodb. http://docs.mongodb.org/manual/core/geospatial-indexes/.

[2] Instituto de estadistica de la comunidad de madrid: http://www.madrid.org/iestadis/.

[3] Milan's public data. http://dati.comune.milano.it/, '14.

[4] Mongodb. http://www.mongodb.org/.

[5] Scaling mongodb at foursquare. http://www.10gen.com/presentations/mongonyc-2012-scaling-mongodb-foursquare.

[6] Twitter's streaming api. https://dev.twitter.com/docs/streaming-apis.

[7] World urbanization prospects: The 2011 revision. United Nations Department of Economic and Social Affairs/Population Division.

[8] Why waiting is torture. http://www.nytimes.com/2012/08/19/opinion/sunday/why-waiting-in-line-is-torture.html, 2012.

[9] 2013 Italian social protests. http://en.wikipedia.org/wiki/2013_Italian_social_protests, 2013.

[10] Big data challenge. http://www.telecomitalia.com/tit/en/bigdatachallenge.html, 2014.

[11] Now i know why investors are going hog wild about uber ... http://www.businessinsider.com/ubers-revenue-2014-11, 2014.

[12] 9th dimacs implementation challenge - shortest paths. http://www.dis.uniroma1.it/challenge9/download.shtml, 2015.

[13] Scoop. https://www.takescoop.com/, 2015.

[14] Meet lyft carpool: A new way to commute. http://blog.lyft.com/posts/meet-lyft-carpool, 2016.

[15] More people in fewer cars. https://newsroom.uber.com/us-washington/more-people-in-fewer-cars/, 2016.

[16] N. Agatz, A. Erera, M. Savelsbergh, and X. Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295 – 303, 2012.

[17] H. Ali-Ahmad, C. Cicconetti, A. La Oliva, V. Mancuso, M. Reddy Sama, P. Seite, and S. Shanmugalingam. An SDN-Based Network Architecture for Extremely Dense Wireless Networks. In *IEEE Conf. for Future Networks and Services (SDN4FNS)*, pages 1–7, Nov. 2013.

[18] A. M. Amey. Real-Time Ridesharing: Exploring the Opportunities and Challenges of Designing a Technology-based Rideshare Trial for the MIT Community. Master's thesis, Massachusetts Institute of Technology, 2010.

[19] J. G. Andrews, H. Claussen, M. Dohler, S. Rangan, and M. C. Reed. Femtocells: Past, Present, and Future. *IEEE Journal on Selected Areas in Communications*, 30(2):497–508, 2012.

[20] A. S. Berger. *The City: Urban Communities and their Problems, '78.*

[21] N. Bicocchi and M. Mamei. Investigating ride sharing opportunities through mobility data analysis. *Pervasive and Mobile Computing*, 2014.

[22] P. J. Brockwell and R. A. Davis. *Time Series: Theory and Methods.* Springer-Verlag, New York, second edition, 1991.

[23] C. T. Butts. Perform quadratic assignment procedure (qap) hypothesis tests for graph-level statistics., 2014.

[24] F. Calabrese, M. Colonna, P. Lovisolo, D. Parata, and C. Ratti. Real-time urban monitoring using cell phones: A case study in rome. *IEEE Transactions on Intelligent Transportation Systems*, 2011.

[25] F. Calabrese, J. Reades, and C. Ratti. Eigenplaces: Segmenting space through digital signatures. *IEEE Pervasive Computing*, 2010.

[26] S. Carmi, S. Havlin, S. Kirkpatrick, Y. Shavitt, and E. Shir. A model of internet topology using k-shell decomposition. *PNAS*, 2007.

[27] R. Cavalcante, S. Stanczak, M. Schubert, A. Eisenblaetter, and U. Tuerke. Toward Energy-Efficient 5G Wireless Communications Technologies: Tools for decoupling the scaling of networks from the growth of operating power. *IEEE Signal Processing Magazine*, 31(6):24–34, 2014.

[28] K. Cechlárová and T. Fleiner. On a generalization of the stable roommates problem. *Transactions on Algorithms*, 1(1):143–156, 2005.

[29] C. Chatfield. *The Analysis of Time Series: An Introduction.* 1995.

[30] E. Cho, S. Myers, and J. Leskovec. Friendship and mobility: user movement in location-based social networks. In *SIGKDD*, 2011.

[31] M. D. Choudhury, W. A. Mason, J. M. Hofman, and D. J. Watts. Inferring Relevant Social Networks. In *Proc. of WWW*, 2010.

[32] B. Cici, M. Gjoka, A. Markopoulou, and C. T. Butts. On the Decomposition of Cell Phone Activity Patterns and their Connection with Urban Ecology. In *Proc. ACM MobiHoc 2015*, Hangzhou, China, June 2015.

[33] B. Cici, A. Markopoulou, E. Frias-Martinez, and N. Laoutaris. Quantifying the Potential of Ride-Sharing using Call Description Records. In *Proc. of HotMobile*, 2013.

[34] B. Cici, A. Markopoulou, E. Frias-Martinez, and N. Laoutaris. Assessing the potential of ride-sharing using mobile and social data: A tale of four cities. In *Proc. of Ubicomp*, 2014.

[35] B. Cici, A. Markopoulou, and N. Laoutaris. Designing an On-Line Ride-Sharing System. In *Proc. of SIGSPATIAL (short paper)*, 2015.

[36] Cisco. Visual Networking Index: Global Mobile Data Traffic Forecast Update 2015–2020 White Paper. `http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html`, Feb. 2016.

[37] J. F. Cordeau and G. Laporte. The Dial-a-Ride Problem (DARP): Variants, modeling issues and algorithms. *4OR*, 1:89–101, 2003.

[38] J. Cranshaw, R. Schwartz, J. Hong, and N. Sadeh. The Livehoods Project: Utilizing Social Media to Understand the Dynamics of a City. In *Proc. of ICWSM*, 2012.

[39] K. D. Carpooling: Status and potential. Final Report U.S. Department of Transportation, DOT-TSC-OST-75-23, 1975.

[40] K. Dasgupta, R. Singh, B. Viswanathan, D. Chakraborty, S. Mukherjea, A. a. Nanavati, and A. Joshi. Social ties and their relevance to churn in mobile telecom networks. *in Proc. of EDBT*, 2008.

[41] S. Dawoud, A. Uzun, S. Gondor, and A. Kupper. Optimizing the Power Consumption of Mobile Networks Based on Traffic Prediction. In *IEEE 38th Annual Computer Software and Applications Conf. (COMPSAC)*, pages 279–288. IEEE, 2014.

[42] N. Eagle, A. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *PNAS*, 2009.

[43] S. L. Feld. Why your friends have more friends than you do. *American Journal of Sociology*, 96(6):1464–1477, 1991.

[44] M. Ficek and L. Kencl. Inter-Call Mobility Model: A Spatio-temporal Refinement of Call Data Records Using a Gaussian Mixture Model. In *Proc. of Infocom*, 2012.

[45] C. S. Fischer. Toward a subculture theory of urbanism. *AJS*, 1975.

[46] E. Frentzos. Indexing objects moving on fixed networks. In *Proc. of SSTD*, 2003.

[47] E. Frias-Martinez, V. Soto, and H. Hohwald. Characterizing urban landscapes using geolocated tweets. In *Proc. of SocialCom*, 2012.

[48] V. Frias-Martinez, C. Soguero, and E. Frias-Martinez. Estimation of urban commuting patterns using cellphone network data. In *Proc. UrbComp*, 2012.

[49] T. Fujisaka, R. Lee, and K. Sumiya. Exploring urban characteristics using movement history of mass mobile microbloggers. In *Proc. of Hotmobile*, 2010.

[50] M. Furuhata, M. Dessouky, F. Ordónez, M.-E. Brunet, X. Wang, and S. Koenig. Ridesharing: The state-of-the-art and future directions. *Transportation Research Part B: Methodological*, 57(0):28 – 46, 2013.

[51] R. Grannis. The importance of trivial streets: Residential streets and residential segregation. *American Journal of Sociology*, 1998.

[52] W. He, D. Li, T. Zhang, L. An, M. Guo, and G. Chen. Mining regular routes from gps data for ridesharing recommendations. In *Proc. of UrbComp*, 2012.

[53] N. Hodas, F. Kooti, and K. Lerman. Friendship paradox redux: Your friends are more interesting than you. In *Proc. of ICWSM*, 2013.

[54] S. Isaacman, R. Becker, R. Cáceres, S. Kobourov, M. Martonosi, J. Rowland, and A. Varshavsky. Identifying Important Places in People's Lives from Cellular Network Data. In *Proc. of Pervasive Computing*, 2011.

[55] S. Isaacman, R. Becker, R. Cáceres, M. Martonosi, J. Rowland, A. Varshavsky, and W. Willinger. Human Mobility Modeling at Metropolitan Scales. In *Proc. of MobiSys*, June 2012.

[56] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman. Analysis of a local search heuristic for facility location problems. *Journal of Algorithms*, 37:146–188, 2000.

[57] D. Krackhardt. QAP Partialling as a Test of Spuriousness. *Social Networks*, 9:171–186, 1987.

[58] R. Li, Z. Zhao, X. Zhou, J. Palicot, and H. Zhang. The Prediction Analysis of Cellular Radio Access Network Traffic: From Entropy Theory to Networking Practice. *Communications Magazine, IEEE*, 52(6):234–240, 2014.

[59] R. N. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New Perspectives and Methods in Link Prediction. In *Proc. of the 16th ACM Int. Conf. on Knowledge discovery and data mining (SIGKDD)*, pages 243–252, 2010.

[60] D. López-Pérez, A. Ladányi, A. Jüttner, H. Rivano, and J. Zhang. Optimization Method for the Joint Allocation of Modulation Schemes, Coding Rates, Resource Blocks and Power in Self-Organizing LTE Networks. In *Proc. IEEE INFOCOM.*, pages 111–115, Apr. 2011.

[61] S. Ma, Y. Zheng, and O. Wolfson. T-Share: A Large-Scale Dynamic Taxi Ridesharing Service. In *Proc. of ICDE*, 2013.

[62] G. Magdalena Nohrborg. Self-Organizing Networks. `http://www.3gpp.org/technologies/keywords-acronyms/105-son`, 2015.

[63] B. McKenzie. Out-of-state and long commutes: 2011. American Community Survey Reports, 2011.

[64] B. McKenzie and M. Rapino. Commuting in the united states: 2009. American Community Survey Reports, 2009.

[65] S. Morosi, P. Piunti, and E. Del Re. A Forecasting Driven Technique Enabling Power Saving in LTE Cellular Networks. In *Wireless and Mobile Computing, Networking and Communications (WiMob), 2013 IEEE 9th Int. Conf. on*, pages 217–222, 2013.

[66] K. Mouratidis and M. L. Yiu. Anonymous query processing in road networks. *IEEE Trans. Knowl. Data Eng.*, 22(1), 2010.

[67] A. Noulas, C. Mascolo, and E. Frias-Martinez. Exploiting foursquare and cellular data to infer user activity in urban environments. In *Proc. of MDM*, 2013.

[68] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. Exploiting semantic annotations for clustering geographic areas and users in location-based social networks. In *Proc. of SMW*, 2011.

[69] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. Exploiting semantic annotations for clustering geographic areas and users in location-based social networks. In *Proc. of SMW*, 2011.

[70] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and a L Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences of the United States of America*, 104:7332–7336, 2007.

[71] C. H. Papadimitriou and K. Steiglitz. Algorithms for matching. In *Combinatorial Optimization, Algorithms and Complexity*, chapter 10, pages 221–226. 1998.

[72] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das. Understanding Traffic Dynamics in Cellular Data Networks. In *Proc. IEEE INFOCOM*, pages 882–890, 2011.

[73] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das. Understanding Spatial Relationships in Resource Usage in Cellular Data Networks. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 244–249. IEEE, 2012.

[74] B. Pittel, J. Spencer, and N. Wormald. Sudden emergence of a giant k-core in a random graph. *Journal of Combinatorial Theory*, 1996.

[75] Qualcomm Technologies. 1000x Data Challenge. https://www.qualcomm.com/invention/technologies/1000x, 2015.

[76] C. Ratti, S. Williams, D. Frenchman, and R. Pulselli. Mobile landscapes: using location data from cell phones for urban analysis. *Environment and Planning b Planning and Design*, 33(5):727, 2006.

[77] C. J. Riederer, A. Chaintreau, J. Cahan, and V. Erramilli. Challenges of keyword-based location disclosure. In *Proc. of WPES*, 2013.

[78] H. S. Implementing Real-Time Ridesharing in the San Francisco Bay Area. . Master's thesis, Mineta Transportation Institute, San Jose State University, CA, USA, 2010.

[79] R. J. Sampson, S. W. Raudenbush, and F. Earls. Neighborhoods and violent crime: A multilevel study of collective efficacy. *Science 1997*.

[80] C. Smith-Clarke, A. Mashhadi, and L. Capra. Poverty on the cheap: Estimating poverty maps using aggregated mobile communication networks. In *Proc. of CHI*, 2014.

[81] F. Soldo, A. Le, and A. Markopoulou. Predictive Blacklisting as an Implicit Recommendation System. In *Proc. IEEE INFOCOM*, 2010.

[82] V. Soto and E. Frias-Martinez. Automated Land Use Identification using Cell-Phone Records. In *Proc. of HotPlanet*, 2011.

[83] Statista. Number of mobile phone users worldwide from 2013 to 2019 (in billions). http://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/.

[84] R. Teal. Carpooling: Who, how and why. *Transportation Research*, 21A(3):203–214, 1987.

[85] D. Tikunov and T. Nishimura. Traffic Prediction for Mobile Network Using Holt-Winter's Exponential Smoothing. In *2007 15th Int. Conf. on Software, Telecommunications and Computer Networks*.

[86] J. L. Toole, M. Ulm, M. C. González, and D. Bauer. Inferring land use from mobile phone activity. In *Proc. of UrbComp*, 2012.

[87] R. Trasarti, F. Pinelli, M. Nanni, and F. Giannotti. Mining mobility user profiles for car pooling. In *Proc. of UrbComp*, 2011.

[88] H.-S. J. Tsao and D. Lin. Spatial and temporal factors in estimating the potential of ride-sharing for demand reduction. California PATH Research Report, UCBITS-PRR-99-2, 1999.

[89] S. Wakamiya, R. Lee, and K. Sumiya. Urban area characterization based on semantics of crowd activities in twitter. In *GeoSpatial Semantics*, Lecture Notes in Computer Science. 2011.

[90] J. Yuan, Y. Zheng, and X. Xie. Discovering regions of different functions in a city using human mobility and pois. In *KDD*, 2012.

[91] H. Zang and J. Bolot. Anonymization of Location Data Does Not Work: A Large-Scale Measurement Study. In *Proc. of the 17th ACM Int. Conf. on Mobile computing and networking (MobiCom)*, pages 145–156. ACM, Apr. 2011.

[92] Y. Zhao and G. Karypis. Criterion functions for document clustering: Experiments and analysis. Technical report, 2001.