

**MOCHA: UM FRAMEWORK PARA  
CARACTERIZAR E COMPARAR TRACES DE  
MOBILIDADE**



JAVIER JESUS MEDINA DIAZ

**MOCHA: UM FRAMEWORK PARA  
CARACTERIZAR E COMPARAR TRACES DE  
MOBILIDADE**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

ORIENTADOR: PEDRO OLMO STANCIOLI VAZ DE MELO  
COORIENTADOR: ANTONIO ALFREDO FERREIRA LOUREIRO

Belo Horizonte

Março de 2015



JAVIER JESUS MEDINA DIAZ

**MOCHA: A FRAMEWORK TO CHARACTERIZE  
AND COMPARE MOBILITY TRACES**

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computer Science.

ADVISOR: PEDRO OLMO STANCIOLI VAZ DE MELO  
CO-ADVISOR: ANTONIO ALFREDO FERREIRA LOUREIRO

Belo Horizonte

March 2015

© 2015, Javier Jesus Medina Diaz.  
Todos os direitos reservados

**Ficha catalográfica elaborada pela Biblioteca do ICEx - UFMG**

Medina Diaz, Javier Jesus.

M491m MOCHA: a framework to characterize and compare mobility traces / Javier Jesus Medina Diaz. — Belo Horizonte, 2015.  
xxv, 63 f. : il. ; 29cm.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais  
Departamento de Ciência da Computação.

Orientador: Pedro Olmo Stancioli Vaz de Melo  
Coorientador: Antônio Alfredo Ferreira Loureiro.

1. Computação - Teses. 2. Redes de computadores - Teses. 3.  
Roteamento (Administração de redes de computadores) – Teses. I.  
Orientador. II. Coorientador. III. Título.

519.6\*22(043)



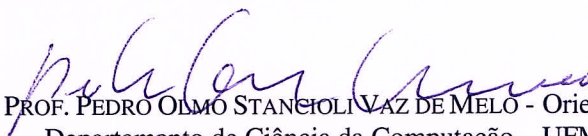
UNIVERSIDADE FEDERAL DE MINAS GERAIS  
INSTITUTO DE CIÊNCIAS EXATAS  
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

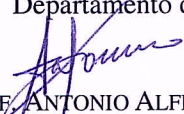
## FOLHA DE APROVAÇÃO


Mocha: a framework to characterize and compare mobility traces

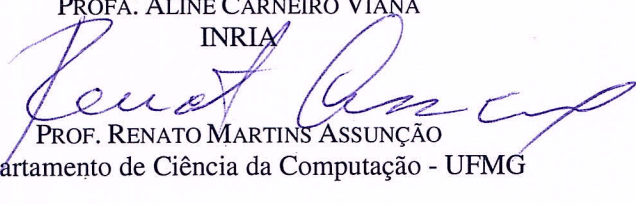
**JAVIER JESUS MEDINA DIAZ**

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

  
PROF. PEDRO OLMO STANCIOLLI VAZ DE MELO - Orientador  
Departamento de Ciência da Computação – UFMG

  
PROF. ANTONIO ALFREDO FERREIRA LOUREIRO - Coorientador  
Departamento de Ciência da Computação - UFMG

  
PROFA. ÁLINE CARNEIRO VIANA  
INRIA

  
PROF. RENATO MARTINS ASSUNÇÃO  
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 27 de março de 2015.





*Para meu pai, Gustavo Medina, por ser meu maior exemplo de vida; para minha mãe, Maira Diaz de Medina, que sempre sabe as palavras perfeitas para cada situação; e para as minhas irmãs, Mariana e Andreina, por serem fontes constantes de amor e alegria na minha vida.*



# Agradecimentos

Primeiramente gostaria de agradecer a Deus, por todas as graças, dons e virtudes concedidas ao longo desta jornada e da minha vida.

Ao meu pai, Gustavo, por todos os conselhos e debates que contribuíram para a minha formação e a deste trabalho. Obrigado por me mostrar através do teu exemplo que nenhuma carga é pesada quando acreditamos no propósito daquilo que estamos fazendo. À minha mãe, Maira, pelo apoio incondicional em todas as minhas decisões e pelo conforto nos momentos mais difíceis. Agradeço também às minhas irmãs, Mariana e Andreina, e a toda à minha família que, mesmo na distância, se fez presente a todo momento.

Obrigado à minha amiga Sthéfanni, por todas as conversas, risadas e situações inusitadas que a nossa amizade nos proporcionou durante esta etapa da minha vida. Ao meu amigo Khalil, pela confiança, pelo constante apoio e por todos os momentos resultantes da nossa convivência diária em Belo Horizonte. À Jocelyn, por sempre estar disposta a ouvir as minhas mais loucas teorias e principalmente por incentivá-las. Aos meus amigos Lucas e Diego por apoiar constantemente meu trabalho e me manterem sempre motivado.

Quero agradecer a todas as pessoas que me acolheram em Belo Horizonte e me fizeram sentir em casa desde o momento em que cheguei, especialmente ao Daniel, à Isadora, sua família e ao Marcos Martins. Agradeço também aos meus colegas de mestrado, Alessandro e Rodrigo Borges, pelos conselhos e pelo apoio em todos os momentos que precisei. Agradeço ao Rodrigo Maués, por me mostrar a resiliência de uma amizade e como o trabalho em equipe sempre pode nos levar mais longe.

Muito obrigado ao Pedro Olmo, meu orientador, pela paciência, contribuições e constantes direcionamentos durante o meu trabalho. Agradeço também ao meu co-orientador, Antonio Loureiro, por apoiar as minhas decisões ao longo desta jornada, por se mostrar compreensivo e zelar sempre pela qualidade da minha pesquisa.



*“You can’t connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future.”*

(Jobs, Steve)



# Resumo

A avaliação de algoritmos em MANETs utilizando simulação é fortemente dependente da representação precisa do cenário de mobilidade, ou seja, o modelo de mobilidade e/ou os *traces* utilizados na simulação devem refletir a realidade da forma mais fiel possível. Existem inúmeros *traces* na literatura que são utilizados como *benchmark* para validar modelos de mobilidade, geradores e soluções para redes sem fio. Além disso, considerando a heterogeneidade dos *datasets* reais de mobilidade que estão publicamente disponíveis é crucial, se não necessário, entender as diferenças e semelhanças entre eles e as suas características. Este trabalho propõe o MOCHA (*MObility CHaracterization Framework*), um framework que extrai, classifica e compara propriedades de mobilidade baseado na suas distribuições marginais. Estas propriedades contemplam diferentes aspectos da mobilidade humana e são divididas em três categorias: *sociais*, *espaciais* e *temporais*. MOCHA é composto por três módulos: um *parser*, um *extrator* e um *classificador*. O *parser* é responsável por converter qualquer *trace* de mobilidade em um formato padrão que possa ser interpretado pelo MOCHA. O *extrator* é responsável por extrair até 11 propriedades de mobilidade do *trace* no formato padrão. Por último, o *classificador* analisa a distribuição marginal de cada propriedade, as separa em *cabeça* e *cauda*, e as classifica de acordo com a *cauda*. Uma vez que o MOCHA extrai e classificou cada propriedade, é possível observar as semelhanças entre diferentes *datasets* utilizando um algoritmo *k-means* e fazendo uma Análise de Componentes Principais (*PCA*). MOCHA foi validado utilizando 13 *traces* reais que são conhecidos na literatura. Além disso, a validação contou com 18 *traces* sintéticos gerados com ferramentas conhecidas na literatura. Ao comparar *traces* reais e sintéticos mostrou-se como os geradores de mobilidade considerados falham em reproduzir cenários realistas e como a metodologia utilizada pelo MOCHA é mais elegante que a dos seus predecessores e mais completa, pois considera todos os passos desde a conversão de um *trace* de mobilidade, até a classificação das propriedades de mobilidade.

**Palavras-chave:** mobilidade social, classificação, mocha.





# Abstract

The evaluation of mobile network algorithms via simulation is strongly dependent of the accurate representation of the mobility scenario. i.e., the mobility models and/or traces used in simulation should reflect reality as much as possible. There are numerous traces in the literature that are commonly used as benchmark to validate mobility models, generators and wireless networking solutions. Thus, considering the heterogeneity of the real mobility datasets that are publicly available, it is crucial, if not necessary, to understand the differences and similarities among them and also their characteristics. In this work we propose MOCHA, a MObility CHaracterization Framework that extract, classify and compare mobility properties, based on their marginal statistical distribution. These properties cover different aspects of human mobility and can be divided into three categories: *social*, *spatial* and *temporal*. MOCHA is composed by three different modules: a *parser*, a property *extractor* and a *classifier*. The *parser* is responsible for converting any mobility dataset into a standard form that can be understood by MOCHA. The *extractor* is responsible for extracting up to 11 mobility properties from the parsed dataset. Finally, the *classifier* analyzes the marginal statistical distribution of each mobility property, splits them into head and tails, and classify them according to the tails. Once that MOCHA extract and classify each property, it is possible to visualize the similarities among different datasets by using a  $k$ -means algorithm and the Principal Component Analysis method. We validate our proposed framework with 13 real mobility traces that are frequently used as benchmark, in addition to 18 synthetic traces generated by different and well known mobility models. By comparing real and synthetic datasets, our results show how the considered mobility generation tools fail to reproduce realistic scenarios. We also show how MOCHA's methodology is more elegant than their predecessors and more complete, for considering all the steps from the original trace parsing process to the mobility properties classification.

**Palavras-chave:** mobility properties, social mobility, classification, mocha.



# List of Figures

3.1	All Dartmouth mobility properties analyzed by MOCHA . . . . .	17
3.2	Empirical distributions for graphical comparison . . . . .	18
4.1	MOCHA structure . . . . .	26
5.1	Classification of all properties according to MOCHA . . . . .	46
5.2	Dartmouth INCO fitting and AIC error . . . . .	48
5.3	SLAW5 INCO fitting and AIC error . . . . .	49
5.4	$k$ -means ( $k = 2$ ) clustering of all evaluated datasets according to MOCHA using all available properties from all datasets . . . . .	51
5.5	$k$ -means ( $k = 4$ ) clustering of all evaluated datasets according to MOCHA using only social properties from all datasets . . . . .	52
5.6	$k$ -means ( $k = 2$ ) clustering of all evaluated datasets according to MOCHA using only social properties from all datasets . . . . .	53



# List of Tables

2.1	Mobility traces of real scenarios . . . . .	14
2.2	Mobility traces of synthetic scenarios . . . . .	14



# Contents

<b>Agradecimientos</b>	<b>xi</b>
<b>Resumo</b>	<b>xv</b>
<b>Abstract</b>	<b>xvii</b>
<b>List of Figures</b>	<b>xix</b>
<b>List of Tables</b>	<b>xxi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Problem . . . . .	1
1.2 Motivation . . . . .	2
1.3 Objectives . . . . .	3
1.4 Contributions . . . . .	4
1.5 Work organization . . . . .	4
<b>2 Related Work</b>	<b>5</b>
2.1 Mobility traces . . . . .	5
2.1.1 Campus scenario . . . . .	5
2.1.2 Vehicular scenario . . . . .	6
2.1.3 Conference scenario . . . . .	7
2.2 Synthetic mobility generators . . . . .	8
2.2.1 SWIM . . . . .	8
2.2.2 SLAW . . . . .	8
2.2.3 Working Day Model . . . . .	9
2.3 Comparing Mobility Traces . . . . .	10
2.4 Mobility-based networking solutions . . . . .	11
2.5 Overview . . . . .	12

2.6	Datasets . . . . .	13
<b>3</b>	<b>Mobility Properties</b>	<b>15</b>
3.1	Discussion . . . . .	16
3.2	Social properties . . . . .	16
3.2.1	Inter-contact time (INCO) . . . . .	18
3.2.2	Contact duration (CODU) . . . . .	19
3.2.3	Maximum contacts per hour (MAXCON) . . . . .	19
3.2.4	Encounter regularity (EDGEPR) . . . . .	20
3.2.5	Topological overlap (TOPO) . . . . .	20
3.3	Spatial properties . . . . .	21
3.3.1	Radius of gyration (RADG) . . . . .	21
3.3.2	Travel distance (TRVD) . . . . .	22
3.4	Temporal properties . . . . .	22
3.4.1	Visit time (VIST) . . . . .	22
3.4.2	Travel time (TRVT) . . . . .	23
3.4.3	Entropy (ENTROPY) . . . . .	23
<b>4</b>	<b>MOCHA: MObility CHaracterization Framework</b>	<b>25</b>
4.1	Overview . . . . .	25
4.2	The parser . . . . .	27
4.2.1	Normalization . . . . .	28
4.2.2	Raw mobility trace . . . . .	29
4.2.3	Check-in traces . . . . .	31
4.2.4	Contact traces . . . . .	33
4.3	The extractor . . . . .	34
4.3.1	Social properties . . . . .	34
4.3.2	Spatial properties . . . . .	38
4.3.3	Temporal properties . . . . .	40
4.4	The classifier . . . . .	42
<b>5</b>	<b>Results</b>	<b>45</b>
5.1	Analysis . . . . .	45
5.2	Sanity Check . . . . .	47
5.3	Classification of traces . . . . .	50
<b>6</b>	<b>Conclusions and Future Work</b>	<b>55</b>
6.1	Conclusions . . . . .	55



6.2 Future Work . . . . .	56
<b>Bibliography</b>	<b>59</b>



# Chapter 1

## Introduction

### 1.1 Problem

Mobility traces are receiving considerable attention by researchers in the last years due to the popularization of particular networks such as opportunistic, vehicular and social networks. These traces are mainly used in simulations to reproduce realistic scenarios in which the mobility plays a crucial role in the process of validating an algorithm, technique or protocol [Aschenbruck et al., 2011; Karamshuk et al., 2011; Treurniet, 2014].

A mobility trace is a collection of movements performed by autonomous agents, such as people, animals, vehicles, among others, containing information about this agent. An important class of trace is the one that contains the time and duration of an encounter between two agents, and sometimes their location. Moreover, the traces can describe very large scenarios, such as taxis in a city [Piorkowski et al., 2009], or smaller ones, such as attendees of a conference [Scott et al., 2006]. Also, the amount of agents in a trace and the duration of their encounters can also vary significantly, going from dozens of nodes or hours to thousands of nodes and several weeks. Therefore, a key step is to thoroughly understand the properties of such traces to make sure they accurately represent the scenarios targeted by the given algorithm, technique or protocol.

The first modern attempts to understand mobile scenarios were through the analysis of real mobility datasets. Such datasets include, for instance, the Dartmouth trace [Henderson et al., 2004], the USC trace [jen Hsu and Helmy, 2008], the Infocom trace [Scott et al., 2006], among others [Aschenbruck et al., 2011; Karamshuk et al., 2011; Treurniet, 2014]. These traces are commonly used as benchmark to validate network protocols [Aschenbruck et al., 2011; Karamshuk et al., 2011; Treurniet, 2014] and also synthetic mobility generation tools that came after them [Kosta et al., 2014; Lee

et al., 2012; Vastardis and Yang, 2014; Karamshuk et al., 2014; Ekman et al., 2008]. The fact is that mobility traces are the basis of every networking solution based on human mobility.

## 1.2 Motivation

Mobility can be characterized by several properties [Karamshuk et al., 2011], but most of the generation tools tend to simplify the comparison between generated traces and real traces to a few statistical (and sometimes visual) properties, what harms the process of validating the generative model. Even in the studies that compare mobility traces, there is no standard methodology to perform this comparison. Different types of frameworks have been proposed in the literature without any relation whatsoever among themselves, making difficult to perform a benchmark with tools and datasets that are not considered by the authors.

Bai et al. [2003] use mobility, connectivity and communication performance metrics to compare traces. Thakur and Helmy [2013] propose a mobility model and a comparison framework with some mobility properties as well as network metrics. Unfortunately, both studies rely on visual comparison and do not consider how to quantitatively characterize the scenarios.

On the other hand, some studies [Munjal et al., 2010; Bezerra et al., 2009] use a quantitative approach to analyze mobility models by focusing on statistical curves and parameter fitting. Munjal et al. [2010] propose a model to construct MANET scenarios by defining values for three specific parameters. However, the meaning of each parameter and how to replicate a specific real scenario are questions not answered in that work. Bezerra et al. [2009] propose a methodology to help fitting simulation parameters to desired scenarios, but once again the work fails to explain the nature of each scenario and how to explain why there are similarities between different datasets.

Several studies, such as Aschenbruck et al. [2011], describe the characteristics of both real and synthetic traces. Karamshuk et al. [2011] classify those characteristics into three categories: temporal, spatial and social. The characterization of such properties is usually done by the analysis of the probability distribution of the desired property. However, usually this analysis is simply a visual comparison between empirical distributions, neglecting any sort of quantitative comparison [Kosta et al., 2014; Lee et al., 2012].

Another relevant point regarding those characteristics, which we will refer for now on as mobility properties, is that even different studies use the same properties to

compare different scenarios. Furthermore, there is no formal methodology regarding how they should be extracted from each mobility trace. At first sight, this does not look like a real problem because each property has a formal definition, giving us an insight about how to extract it. However, when analyzing real and synthetic datasets, we find several *corner cases* needing to be considered to guarantee a fair and standard comparison among datasets.

Finally, it is also important to highlight that in the literature each work compares datasets using a specific set of mobility properties, but never all the well-documented properties. By comparing a limited set of properties, it is hard to benchmark datasets between different studies, creating difficulties in determining whether there are actual similarities between scenarios in a standardized and reliable way or not.

## 1.3 Objectives

The motivation of this work is the establishment of a comparison methodology between mobility scenarios and datasets. Considering this, we propose a MObility CHaracterization Framework (MOCHA) that extracts and compares several mobility properties.

First, we propose a taxonomy to classify the different types of mobility traces found in the literature. It is very important to do this because, given the nature of each trace, we need to translate them to a common form in order to guarantee they can be evaluated and compared. Moreover, given a common form, it is possible to extract and compare mobility properties. This will be important for our next goal.

Once we have the entry trace in a parsed form, we use an algorithm to analyze and extract all the considered mobility properties present in it. More specifically, after setting up MOCHA parameters properly, we want to extract quantitative properties from each trace and classify them according to different categories previously identified. The extracted properties represents different aspects of human mobility (such as social ties and regularity [Karamshuk et al., 2011]) and can be used to propose network solutions, especially in ad hoc scenarios.

MOCHA classifies each property into three different categories regarding their marginal statistical distribution and flag all the properties that cannot be extracted from a specific trace. By doing so, we ensure that properties that are equally classified represent similar behavior on their respective traces. There is only one property not evaluated using its marginal distribution. In that case, MOCHA uses a temporal correlation instead.

We validate our framework with 31 different mobility traces, 18 containing move-

ments of real scenarios, and 13 generated synthetically by well-known models. Our last goal is to categorize these traces using Principal Component Analysis (PCA) to determine whether different traces can be considered similar in nature.

## 1.4 Contributions

This work proposes MOCHA, a complete framework to extract, analyze and compare mobility properties available in mobility traces. By doing so, this is the starting point to allow a fair comparison between mobility scenarios in order to help us understand their real similarities and differences. With MOCHA, we offer a detailed comparison between traces, improving our understanding on the similarities and differences among mobility scenarios.

By using PCA to several well-known mobility traces used in literature, we present a simple but yet practical form to visualize the similarity of different traces using a quantifiable method instead of a visual comparison between arbitrarily selected mobility properties. Some of the traces used in our evaluation are Dartmouth [Henderson et al., 2004], Infocom and Cambridge [Scott et al., 2006], San Francisco [Piorkowski et al., 2009] and synthetic traces generated by popular tools such as SWIM [Kosta et al., 2014], SLAW [Lee et al., 2012] and Working Day Model [Ekman et al., 2008].

## 1.5 Work organization

The rest of this work is organized as follows. In Chapter 2, we describe the related work and present an overview about mobility traces, synthetic mobility generators and some insights about trace comparison and applications. Then, in Chapter 3, we explain the mobility properties that can be found in mobility traces. In Chapter 4, we propose MOCHA, our MObility CHAracterization Framework, and, in Chapter 5, we discuss the results obtained when comparing different synthetic and real traces. Finally, in Chapter 6, we conclude our work explaining the challenges, applications and some possible improvements to our work.

# Chapter 2

## Related Work

### 2.1 Mobility traces

There are several traces commonly seen as benchmark among human mobility models. They can be divided into different categories: campus, urban pedestrian, vehicular and conference. The difference between these categories is not only the geographic scenario itself, but also the methods used to collect the data and the behavior of the agents in each case. In the next sections, we describe each of these categories in details.

#### 2.1.1 Campus scenario

In the campus category, agents are usually students or professors and their movement is restricted to campus locations. Traces such as Dartmouth [Henderson et al., 2004], USC [jen Hsu and Helmy, 2008], SASSY [Bigwood et al., 2011], UPB [Ciobanu and Dobre, 2012] and Cambridge [Scott et al., 2006] are the most known and used in the literature and commonly used to validate networking solutions [Vaz de Melo et al., 2013; Thakur and Helmy, 2013; Alshanyour and Baroudi, 2008] and generative models [Kosta et al., 2014; Lee et al., 2012; Ekman et al., 2008].

Regarding the agents' behavior in this type of scenario, we notice that they tend to have constant velocity when moving with low acceleration (most of the agents walk instead of running). Usually the agents do not have predefined paths or directions restricting their movement but there are paths that are more used than others in specific times of the day. The social network of a campus can be very well defined considering the nature of the scenario itself: students encounter with their colleagues in a daily or weekly basis for months or years, have the same instructors, have lunch at the same restaurant, etc.

Finally, the data collection method used in a campus scenario is usually an association/disassociation method with the Wi-Fi routers of the campus itself. Every time an agent connects or disconnects to a router, a log entry is generated with the time of the event, the agent ID and the router ID. However, some traces such as Sassy [Bigwood et al., 2011] and [Ciobanu and Dobre, 2012] use Bluetooth instead of Wi-Fi, increasing the precision but limiting the amount of nodes.

This type of data collection method has as main drawback the problem of consecutive handovers between nearby routers generating, sometimes, noise in the trace. When analyzed, the trace might pass the idea that an agent is constantly moving between locations when it is actually in a handover zone. Another problem this type of data collection emerges is that the geographical coordinates of each router are not always available to the public, making impossible to estimate the actual distances that a node traveled during the trace collection.

### 2.1.2 Vehicular scenario

There are also vehicular traces, where the agents are vehicles and streets with specific directions and speed restrictions guide their movement. As examples, we can cite San Francisco Cabs [Piorkowski et al., 2009] and Cologne [Uppoor and Fiore, 2012] traces as the most commonly used [Aschenbruck et al., 2011; Karamshuk et al., 2011; Treurniet, 2014].

In this type of scenario, the velocity and acceleration of the agents is more variable than in a pedestrian scenario [Ekman et al., 2008]. Also, it is important to highlight that the paths used by the agents in a vehicular scenario usually have velocity and direction constraints, which are explored by data dissemination protocols and opportunistic networks [Bai et al., 2003].

The social network of a vehicular scenario usually has a random behavior [Vaz de Melo et al., 2013]. Even when the same agents use the same constrained path between two points, it is highly unlikely that they do that at the same time every day. This type of behavior makes more difficult to encounter the same pair of nodes near each other with a certain regularity, giving the social network of a mobility scenario a more random structure.

GPS is the most used technology to collect data in vehicular scenarios. By having a GPS device at each agent (generally a vehicle), it is possible to track the position of each node with high granularity, knowing its exact position at every moment of the trace. Besides, if we know the city (or road) where the agent is moving, it is possible to use a map to tune the path in order to match the actual road.



The GPS technology has a major drawback when regarding mobility in urban scenarios despite allowing a constant track of each agent. GPS only works well with a clear line of sight between the device and the GPS satellites. In that case, in a city with high buildings, the precision of a GPS can be compromised and the position of a node can be misplaced to a different road or street than the one that is being actually used. This type of behavior might present itself as noise in the mobility trace.

### 2.1.3 Conference scenario

Conference scenarios are a specific type of what we can call an indoor scenario. In the conference category, agents are usually students, professors and other kind of participants of the conference, but the same behavior can be compared to hospitals, schools and other indoor scenarios. The Infocom [Scott et al., 2006] and other traces related to hospital and high school [Isella et al., 2011; Vanhems et al., 2013; Fournet and Barrat, 2014] are examples of such traces.

In conference scenarios, the agents move in a similar way of a campus scenario. However, the duration of the trace and the area where the agents move is usually smaller than in a campus trace. Moreover, there are usually no paths, velocity and direction constraints, meaning that the agents move according to a social and not geographic pattern.

The social network of this type of scenarios strongly depends on the scenario itself. A school, for example, will have a very well defined social network, given that students share a lot of time together and have common friends. However, in the case of an hospital, the social network might be represented mostly by the hospital staff, when the patients will configure a random social network between themselves and the staff [Isella et al., 2011].

The data collection method used in indoor scenarios might vary from Wi-Fi association/disassociation, as in a campus scenario, to the use of proximity devices to determine when an encounter between two nodes actually occurred. When using proximity devices, each agent has a unique radio device representing him/her. Once two devices are within each other range, a log entry is generated with each node ID and the time of the beginning of the contact. The same procedure is repeated once the nodes lost communication with each other.

The main drawback of collecting mobility data using proximity devices is the lack of geographic information. Most of the time, the encounter coordinates are not available for logging, making impossible to determine the most popular locations present in the trace.

## 2.2 Synthetic mobility generators

Considering the problem of generating mobility traces, the fundamental step is to observe the characteristics of real scenarios, usually done by extracting their statistical properties and represent them as a probability distribution. Previous analysis, as the ones presented by Chaintreau et al. [2007], Song and Kotz [2007] and Helgason et al. [2014a], also show that the behavior of some statistical properties is common to human mobility independently of the environment [Karagiannis et al., 2010]. Based on such analysis, several tools have been proposed to synthetically generate human mobility, some of them discussed below.

However, here we focus on only three: SWIM, SLAW and WDM. We consider those three tools because they are publicly available and are easy to set up and install, making possible to anyone to reproduce the results presented in this work. Moreover, these generators are very popular and constantly used by researchers to validate networking solutions [Munjal et al., 2010; Sandulescu et al., 2013; Batabyal and Bhaumik, 2014].

### 2.2.1 SWIM

Kosta et al. [2014] propose a generator called SWIM (Small World In Motion) to generate synthetic small world scenarios. In SWIM, each node receives a home location and moves around cells, calculating each cell popularity according to the number of nodes present at each location. Until a cell is visited, its popularity to the node is defined as zero. The intuition behind SWIM is that nodes visit with higher probability nearby or popular cells.

SWIM assumes that the time spent between two locations by any node will be always the same. This is based on the fact that agents tend to accomplish short distance jumps on foot and long distance jumps by car or plane. Later in this work, it will be shown when this assumption is in fact valid and in which cases are not.

According to [Kosta et al., 2014], SWIM can successfully reproduce real traces behavior such as the Cambridge mobility traces [Scott et al., 2006] and Dartmouth [Henderson et al., 2004] by comparing three different mobility properties: inter-contact time, contact duration and contacts per node. However, as we show in this work, SWIM fails to model the social regularity present in real traces.

### 2.2.2 SLAW

Lee et al. [2012] propose SLAW, a model based on four basic premises: truncated

power-law flights and pause times, heterogeneously bounded mobility areas, truncated power-law inter-contact times and fractal waypoints. They show these premises are intrinsically related, since the validation of one of them tends to generate the behavior expected in some of the others.

Lee et al. [2012] also show that their model successfully fits power-law distribution in properties such as inter-contact time and flight distances using the Akaike test. However, their comparison is performed with traces collected by themselves and are not publicly available. This creates obvious difficulties to benchmark their tool with other real datasets.

The decision algorithm of SLAW agents is strongly based on what they call as *Leas action trip planning*, or LAPT. LAPT states that every time agents need to visit more than one location, they choose their next destination primarily based on the shortest distance. Only when all the possible locations are within a specific radius (say 30 meters) the decision criteria is not the distance but the importance of the location itself.

### 2.2.3 Working Day Model

Finally, Ekman et al. [2008] propose a mobility model called Working Day Model (WDM), which has the purpose of modeling realistically daily routines. To accomplish such modeling, they created separated sub-models to represent the different parts of human routine: home sub-model, office sub-model and evening activity sub-model.

They also model different types of mobility such as walking, bus and cars. By assigning to each node a home location and a work location, WDM sets the mobility of each agent according to the time of the day and its specific parameters (e.g., working hours and how often each agent uses a car or a public transportation). For example, by the morning each node goes from home to work using its preferred transportation method (e.g., foot, car or bus) and spends all their working hours moving according to an office mobility pattern.

Once the working hours finish, each agent decides with a given probability whether it will go back home or to a social meeting with friends using an evening activity mobility model. In the evening activity model, each agent is assigned to a meeting location and, once it arrives, it waits until all friends arrive, too. Then, the whole group start visiting night locations until certain time, when everyone goes back home.

When compared with known traces such as Dartmouth [Henderson et al., 2004] and Cambridge [Scott et al., 2006], WDM seems to successfully reproduce some real

datasets properties. Yet, WDM does not clearly evaluate all the statistical mobility properties that can be extracted from generated mobility traces.

## 2.3 Comparing Mobility Traces

In order to determine whether two or more scenarios are similar, a comparison is needed. This comparison is usually done by extracting properties from each trace and comparing the statistical distribution of their values [Thakur and Helmy, 2013; Bezerra et al., 2009; Bai et al., 2003]. Another common approach is to use network protocols and evaluate the underlying network using metrics such as delay, throughput and overhead to determine if the protocol exhibits a similar performance in both scenarios [Meghanathan and Milton, 2009; Boldrini et al., 2007]. Several studies compare mobility traces by creating different scenarios in NS-2 [Mccanne et al., 2007] considering parameters such as simulation time, area, number of nodes, transmission range, pause time, traffic rate, etc [Meghanathan and Milton, 2009; Boldrini et al., 2007; Bai et al., 2003].

The compared properties of each study are usually different. However, even when considering different comparison parameters and methodologies, most of those studies conclude that mobility affect the performance of opportunistic networks and there are underlying social properties that need to be considered in order to better understand their results.

Munjal et al. [2010] propose a methodology to construct synthetic mobility scenarios using SLAW [Lee et al., 2012]. By studying carefully the impact of all the configuration parameters of SLAW, they provide models to construct realistic mobility scenarios. However, they do not consider any mobility property during their evaluation such as inter-contact time or contact duration. Thus, they only use network metrics to evaluate their results.

Bezerra et al. [2009] propose a framework to compare mobility models by using mobility properties instead of network metrics. By extracting properties, such as agent speed, acceleration and pause time, they use a fitting method to determine how to set the mobility models parameters in order to imitate real data. They conclude that some mobility models, such as *Levy-walk*, MMIG and *Smooth*, have too many parameters, making very hard to determine in which intervals these parameters can actually generate mobility that resembles real scenarios.

While some studies use only realistic mobility models to evaluate their results, other studies, such as [Thakur and Helmy, 2013], actually compare their results with

well-known real datasets, such as the ones presented in Section 2.1. Thakur and Helmy [2013] propose COBRA, a mobility model based on communal behavior, and benchmark it with real datasets using an analysis framework. COBRA compares mobility properties, such as inter-contact times and contact duration, besides network metrics. However, they do not explain how the mobility properties are extracted from each trace. They validate the extracted data through a visual graphical comparison, but neglect any quantitative analysis.

Helgason et al. [2014b] compare pedestrian scenarios generated by Legion Studio [Helgason et al., 2010] using mobility properties, such as inter-contact times, contact duration and path duration. Once the properties are extracted, they compare different scenarios by fitting statistical curves using Kolmogorov-Smirnov method. They conclude that the considered properties are not enough to understand exactly why the compared traces are different and state the need of an extensive suit of benchmark traces for evaluating mobility in different environments rather than attempting to derive a one-size-fits-all analytical model for pedestrian mobility.

By proposing MOCHA, our work presents a benchmark framework with 18 well-known real mobility datasets, using more than 10 mobility properties to compare and classify them. Considering its structure, MOCHA focus on understanding the mobility characteristics of each scenario, rather than creating a generic mobility model that suits all scenarios.

## 2.4 Mobility-based networking solutions

While the previously described studies focus on understanding mobility models and compare them to validate their findings, another line of research focus on developing networking solutions based on mobility analysis. Vaz de Melo et al. [2013] propose RECAST, a strategy used to classify interactions among wireless users based on social regularity and similarity. They also apply their strategy to well-known real datasets and use network metrics to validate it.

Foroozani et al. [2014] use real mobility collected data to propose a mobility model. They benchmark it with some other real scenarios mobility properties, such as flight length and flight time. They consider four types of mobility properties: spatial, temporal, connectivity properties and geographical properties. However, they do not detail how to extract these properties and their property comparison is qualitative (visual) instead of quantitative.

Similarly to [Foroozani et al., 2014], Shah and Rathod [2014] use mobility prop-

erties such as flight time and pause time to propose a novel scheduling algorithm for opportunistic networks. They use mobility scenarios generated only from theoretical models, neglecting real mobility traces. Moreover, they validate their results using only network metrics. In the same direction, Alshanyour and Baroudi [2008] evaluate AODV protocol performance.

Finally, mobility-based networking solutions directed to opportunistic scenarios [Fischer et al., 2010] and disaster scenarios [Aschenbruck et al., 2004] can be found in the literature. However, most of the previously described studies fail to provide a detailed framework that allows a standard comparison among different mobility models or traces. While the authors continue to extract mobility properties in an arbitrary way, it will not be possible to properly benchmark different solutions and reach a common understanding of how social mobility truly affects protocols and other networking solutions.

## 2.5 Overview

One of the main purposes of mobility traces is to test and validate applications to be used in real scenarios. However, since it is not possible to collect real data from all possible scenarios, simulators are used to approximate the results according to the specific situations where the applications will be implemented. When regarding human mobility, several mobility properties have been studied in order to enable their synthetic reproduction using simulators. However, considering that real scenarios are very heterogeneous, it is really hard to reproduce each possible scenario using only a few parameters.

Simulators like SWIM bet on their simplicity to enable an intuitive form of mobility generation, but they do not consider the fact that human mobility is more complex than the probability of moving far away from home or to a popular location. Considering that, SLAW tries to capture the social behavior within the mobility by introducing more parameters and by using the locations (or fractal points) as points of social convergence, making possible to the agents to imitate social bounds by meeting large amounts of nodes (and even the same nodes sometimes) at high entropy locations. However, as we show later in this work, SLAW fails to reproduce the social regularity that exists in real datasets.

Finally, WDM relies on the routines of agents to recreate the social behavior. With an impressive amount of parameters, WDM gives the possibility to recreate whatever scenario suiting our needs at the cost of its complexity. However, is it possible

to reproduce a scenario that we do not fully understand? Even with a high set of tunable parameters, how much can we approximate a real scenario with synthetic data if we only rely on a few properties to compare? Moreover, given that the parameter space is very large, it is likely that we generate scenarios with very distinct properties. Thus, a tool to provide a reliable comparison of mobility traces is required.

In summary, the main problem about realistic traces is that they are very heterogeneous, even when they are collected in similar scenarios such as campus, city and conferences. The mobility nature of the nodes might be different at each case, making difficult to create a mobility model based on them [Song et al., 2010]. In the next chapters, we will explain the known mobility properties used in the literature. Moreover, we present MOCHA, a framework that allows us to compare different scenarios, helping us to understand how exactly two scenarios can be considered similar or different.

## 2.6 Datasets

In this work, we analyze several real and synthetic datasets. Each trace has unique mobility properties, but all of them are composed of a set of agents moving around a limited area. The amount of information collected at each case is related to the duration of each trace. While some of them present data collected during a couple of days, some of them were collected for periods longer than 60 days.

It is not the purpose of this work to give all the technical details of each mobility trace. However, in Table 2.1 we present the basic information of all the real mobility traces used in this work. From now on, we will refer to each one of those traces using their name according to Table 2.1. We also present in Table 2.2 the same information regarding all the synthetic traces generated to compare with the real datasets.

Name	Agents	Duration	Reference
Dartmouth	1156	60 days	Henderson et al. [2004]
Cambridge	12	6 days	Scott et al. [2006]
Sassy	27	79 days	Bigwood et al. [2011]
USC	4558	60 days	jen Hsu and Helmy [2008]
UPB	35	4 days	Ciobanu and Dobre [2012]
Infocom	41	3 days	Scott et al. [2006]
Hypertext	113	1 day	Isella et al. [2011]
Hospital	75	4 days	Vanhems et al. [2013]
Highschool 2011	126	7 days	Fournet and Barrat [2014]
Highschool 2012	126	7 days	Fournet and Barrat [2014]
Infectious	416	1 day	Isella et al. [2011]
San Francisco	551	30 days	Piorkowski et al. [2009]
Cologne	700.000 car trips	24 hours	Uppoor and Fiore [2012]

**Table 2.1.** Mobility traces of real scenarios

Trace	Agents	Duration	Reference
Working day model (WDM)	100 agents	15 days	Ekman et al. [2008]
SLAW 5	100 agents	15 days	Lee et al. [2012]
SLAW 35	100 agents	15 days	Lee et al. [2012]
Ostermalm 90	1000 agents	7 days	Kouyoumdjieva et al. [2014]
Ostermalm 70	1000 agents	7 days	Kouyoumdjieva et al. [2014]
Ostermalm 50	1000 agents	7 days	Kouyoumdjieva et al. [2014]
Ostermalm 40	1000 agents	7 days	Kouyoumdjieva et al. [2014]
Ostermalm 30	1000 agents	7 days	Kouyoumdjieva et al. [2014]
SWIM 88	100 agents	7 days	Kosta et al. [2014]
SWIM 85	100 agents	7 days	Kosta et al. [2014]
SWIM 83	100 agents	1 day	Kosta et al. [2014]
SWIM 58	100 agents	30 days	Kosta et al. [2014]
SWIM 55	100 agents	24 hours	Kosta et al. [2014]
SWIM 53	100 agents	15 days	Kosta et al. [2014]
SWIM 38	100 agents	15 days	Kosta et al. [2014]
SWIM 35	100 agents	15 days	Kosta et al. [2014]
SWIM 33	100 agents	15 days	Kosta et al. [2014]

**Table 2.2.** Mobility traces of synthetic scenarios



# Chapter 3

## Mobility Properties

As previously described in Karamshuk et al. [2011], the statistical properties extracted from mobility traces can be categorized in: social, spatial and temporal properties. Despite all mobility properties being intrinsically related, we can divide them in categories according to their application regarding networking solutions and social mobility.

Social properties, such as *inter-contact time* and *contact duration*, can be directly related to **connectivity**. These properties help us to explain how agents connect among themselves, how long they stay within each other range, how they relate to each other and, consequently, how we can explore these properties to benefit our *delivery ratio* or *delivery throughput* in a real network scenario. It is very useful to understand how long agents interact with each other, when developing applications that require data transmission or mobility analysis.

Spatial properties, such as *radius of gyration* and *travel distances*, are related to **movement**, giving us some insights about distances covered by each agent and its relation to locations. Intuitively, we can affirm that people prefer to go to the bakery nearby their home instead of going to a distant bakery, unless, for instance, the distant one is really popular or has been recommended by a friend. Simulators like SLAW [Lee et al., 2012] exploit this kind of intuition by generating mobility using fractal points, as previously described.

Finally, temporal properties, such as visit time, travel time and location entropy, represent **regularity**. This **regularity** reflects the circadian rhythm defining each person's routine. The temporal properties help us to understand how long the agents are moving from one location to another, how long they stay at each place and how agents choose between going to one place or another one. It is important to highlight that, to the best of our knowledge, this work covers the most used mobility properties present in the literature until so far.

### 3.1 Discussion

Mobility properties give us insights about how agents behave in different scenarios. This information can be used to reproduce real datasets with synthetic generators. As previously explained in Sections 3.2, 3.3 and 3.4, most of the properties can be modeled as statistical curves and most of them are represented by heavy-tailed distributions. Here, we plot all the statistical distributions using their complementary cumulative distribution function, or CCDF.

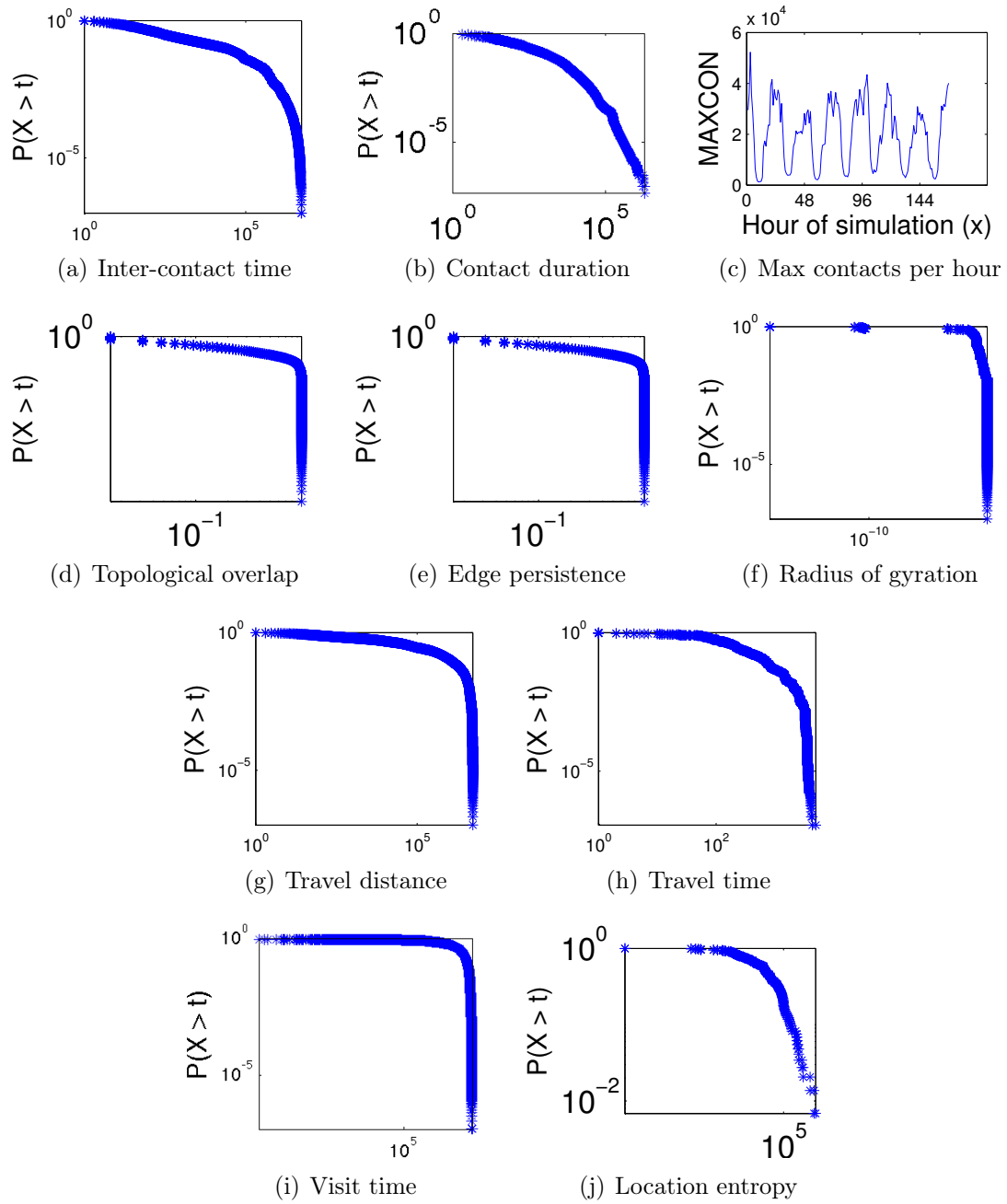
In Figure 3.1, we see all the extracted mobility properties from the Dartmouth dataset. We use this dataset as reference because it is one of the most famous dataset and is commonly used as a benchmark in the literature [Vaz de Melo et al., 2013; Zyba et al., 2011]. We can also observe in Figures 3.2(a), 3.2(b) and 3.2(c) three empirical distributions considered by MOCHA: exponential, pareto and lognormal.

As previously described in Section 1.2, most of the current studies regarding mobility properties rely on a visual comparison of statistical distributions to validate if different datasets are somehow similar or equivalent. Later in this work, we will show how this visual comparison can be tricky and misleading sometimes, showing why it is important to use a quantitative comparison method instead of a visual matching. Note that several plots in Figure 3.1 seem to be heavy-tailed. However, as we show later, some of the distributions are best fitted by a pareto curve while other are best fitted by a lognormal one. This shows, again, how important it is to perform a quantitative analysis when comparing different mobility scenarios.

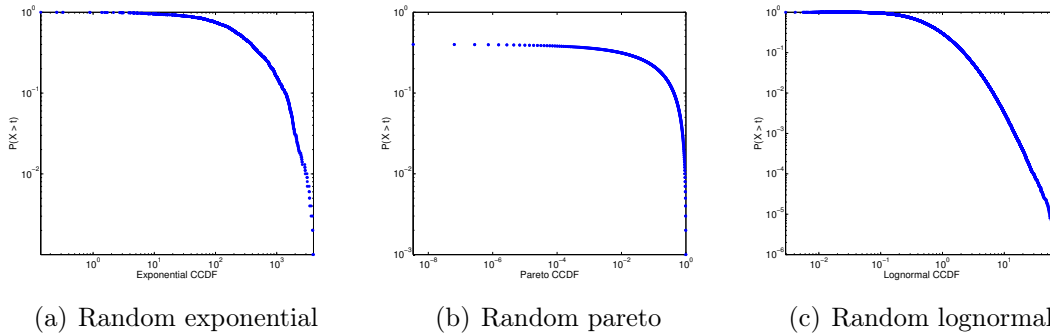
### 3.2 Social properties

In the following, we consider a set of  $m$  agents  $\{N_1, N_2, \dots, N_m\}$ , each one having a position  $N_i(t) = (x, y)$ ,  $i = 1 \dots m$ , as the position  $(x, y)$  in the cartesian plane at time  $t \geq 0$ . Each scenario has several *encounters* between different pairs of nodes. For GPS traces, an encounter between  $N_i$  and  $N_j$  occurs whenever  $dist(i, j) \leq R$ , where  $dist(i, j)$  is the Euclidean distance between  $N_i$  and  $N_j$  and  $R$  is the communication radius of each agent. For traces collected with router association/disassociation, an encounter happens when  $N_i$  and  $N_j$  are connected to the same access point at the same time. In this case, the encounter begins at the moment and ends when one of them disconnects.

Thus, each scenario is composed of an ordered set of encounters  $E = \{e^1, e^2, \dots, e^n\}$ , where each encounter  $e^k = \{N_i, N_j, t_{ini}, t_{fin}\}$  is composed of the two agents  $N_i$  and  $N_j$ , the initial time of the encounter  $t_{ini}$ , and the final time  $t_{fin}$ . The set  $E$  is ordered by the initial time of each encounter. Finally, we define



**Figure 3.1.** All Dartmouth mobility properties analyzed by MOCHA



**Figure 3.2.** Empirical distributions for graphical comparison

$E_{i,j} = (N_i, N_j, t_i, t_f) \in E$  as the set of all the consecutive encounters between  $N_i$  and  $N_j$ .

### 3.2.1 Inter-contact time (INCO)

Inter-contact time can be defined as the time interval between consecutive encounters of a pair of nodes [Kosta et al., 2014]. While inter-contact time is represented by the time between consecutive contacts of the same pair of agents, some authors, such as Helgason et al. [2010], prefer to use a more generic definition called inter-any-contact time, which considers consecutive contacts between any pair of agents. To the purpose of this work, we will use the first definition because it is more popular in the literature. Thus, for a given ordered set  $E_{i,j} = \{e_{i,j}^1, e_{i,j}^2, \dots\}$  of encounters between agents  $N_i$  and  $N_j$ , we compute the inter-contact time between two encounters  $e_{i,j}^k$  and  $e_{i,j}^{k+1}$  as:

$$\text{INCO}_{i,j}^k = t_{ini}^{k+1} - t_{fin}^k.$$

When considering the statistical representation of this property, we usually obtain a heavy-tailed distribution (such as power-law) [Helgason et al., 2014b]. This intuitively explains that, independently of the encounters duration, most of the nodes do not encounter with each other very frequently [Kosta et al., 2014]. For legibility purposes, we will refer the property as INCO and will use the acronym INCO-D when referring to its statistical behavior (i.e., distribution). We will use this notation for all properties in this work.

INCO is very useful when considering opportunistic scenarios because it gives us insights about how often each pair of agents encounter, representing the opportunity to deliver a message [Song and Kotz, 2007]. In Figure 3.1(a), we show INCO-D for the Dartmouth scenario. As we can see, there are INCO entries of different orders of

magnitude, indicating that despite some pairs of agents encountering with each other very frequently, most of them pass long periods without meeting.

### 3.2.2 Contact duration (CODU)

The definition of the contact duration property is very straightforward. It is the amount of time a pair of agents is located within each other's communication range without leaving it. In other words, we can define the CODU of an encounter  $e^k = \{N_i, N_j, t_{ini}, t_{fin}\}$  as:

$$\text{CODU}_{i,j}^k = t_{fin}^k - t_{ini}^k.$$

This behavior can be observed in Figure 3.1(b).

In an analogous way to INCO, CODU also represents an opportunity to deliver a message in an opportunistic network. However, if INCO gives us an insight about how frequently a message can be delivery, CODU helps us to understand for how long an encounter takes places, i.e., how much data can be transferred during an encounter. The understanding of this property is directly related to the possible output that each agent can transmit while in contact with another agent.

### 3.2.3 Maximum contacts per hour (MAXCON)

As explained by Song and Kotz [2007] and Ekman et al. [2008], the maximum contacts per hour is the total amount of contacts that occurred at each hour of the considered dataset. Formally, we define MAXCON as:

$$\text{MAXCON}_h = E^h = \{e = (N_i, N_j, t_{ini}, t_{fin}) | (t_{ini} > h \wedge t_{fin} < h + 1) \vee (t_{ini} \leq h \wedge t_{fin} \geq h)\},$$

where  $E^h$  is the subset of all encounters between every pair  $(N_i, N_j)$  that occurred during the hour  $h$ , and  $|E_h|$  is the sum of those encounters. The set of all the encounters occurring during a scenario evaluation can be defined as  $H = \{|E^1|, |E^2|, \dots\}$ .

Regarding the statistical behavior of  $H$ , we can describe it as a curve with high autocorrelation in 24 h periods [Song and Kotz, 2007; Karamshuk et al., 2014]. The autocorrelation can be explained by the daily routine behavior occurring in any scenario in which agents are humans. The statistical representation of  $H$  will be described as MAXCON-D and its autocorrelation as MAXCON- $C_w$  where  $w$  is the time window considered to calculate the autocorrelation (e.g., 12 h, 24 h, 48 h and 72 h). In Figure 3.1(c), we observe MAXCON-D for the first week of observations in the Dartmouth dataset.

MAXCON is a property that helps us to understand how many contacts occurred during a specific hour. Considering it is a *connectivity* property, we observe that it gives us insights about how many different opportunities of sending a message an agent has during each hour of the day and, in which parts of the day, the agent has more (or less) interactions with other agents (e.g., the number of social contacts during the day is usually higher than during the night).

### 3.2.4 Encounter regularity (EDGEPE)

Edge persistence is a complex network metric that maps the regularity of a social relationship [Vaz de Melo et al., 2013]. By considering a set of encounters  $E_{i,j} = (N_i, N_j, t_i, t_f) \in E$  EDGEPE $_{i,j}$  measures the times the encounter  $e_k = \{N_i, N_j, t_{ini}, t_{fin}\}$  occurred from  $t_0$  to  $t$ , where  $0 \leq t \leq t_{max}$ . Formally, we have:

$$\text{EDGEPE}_{i,j} = \frac{1}{t} \sum_{k=1}^t I_{[(i,j) \in \varepsilon_k]},$$

where  $I_{[(i,j) \in \varepsilon_k]}$  is an indicator function that assumes the value 1 if the encounter  $e_k = \{N_i, N_j, t_{ini}, t_{fin}\}$  occurred in  $\varepsilon_k$  at time  $k$ , and 0 otherwise. For example, if Bob and Patrick met each other twice during a week their edge persistence will be equal to the number of times they encountered, i.e., 2 divided by the total number of time steps, i.e., 7.

According to [Vaz de Melo et al., 2013], we can expect a heavy tailed distribution when considering the statistical behavior of EDGEPE-D. By observing Figure 3.1(e), we observe that there are many EDGEPE with low values (i.e., pairs of nodes encountering sporadically) and few high EDGEPE values, which represent the most strong social ties.

Intuitively we can affirm that an agent tends to encounter more frequently with its friends or acquaintances while random encounters rarely repeat [Vaz de Melo et al., 2013]. The more each pair of nodes encounters in a specific time window, the more probable is to consider a social tie between those nodes. When considering opportunistic scenarios, EDGEPE represents how often it will be possible to deliver a message between a specific pair of nodes.

### 3.2.5 Topological overlap (TOPO)

Another property that gives us insights about the social network of each scenario is the topological overlap. TOPO represents the social overlap existing between each pair of nodes when considering all the encounters in which they participated. First, we consider as  $NG_i$  or *neighbors $_i$*  the set of nodes  $\{N_1, N_2, \dots, N_m\}$  encountering at least once with  $N_i$ . Formally, we define  $TOPO_{i,j}$  as:

$$\text{TOPO}_{i,j} = \frac{|NG_i \cap NG_j|}{|NG_i \cup NG_j|}.$$

Analogous to EDGE-P-D, TOPO-D also presents a heavy-tailed behavior according to [Vaz de Melo et al., 2013], as observed in Figure 3.1(d). TOPO-D gives us insights about the social structure of each scenario by quantifying the similarities between agents.

By considering each pair of agents, we observe in Figure 3.1(d) that most of them have few, or even none, *common friends*, indicating that there is no social tie between them. On other hand, nodes presenting a high TOPO value are more likely to belong to the same communities and have a higher amount of *common friends*.

When regarding network applications, TOPO is helpful to determine opportunities to deliver messages within communities. Intuitively it is easier to deliver a message to a member of the same social community than to a complete stranger.

## 3.3 Spatial properties

### 3.3.1 Radius of gyration (RADG)

In social mobility, it is common to assume that each agent has a location called *home*. citeswim defines *home* as a randomly and uniformly chosen point over the network area. Ekman et al. [2008] consider it as the starting point of each agent’s daily activities. To the purpose of this work, we will consider *home* as the most visited place at each agent’s routine and where normally the agent returns at the end of the day.

It is plausible to assume that an agent tends to move to other locations nearby its home in order to attend commitments, buy food, visit friends, etc. There is a mobility property measuring the distance an agent “orbits” around its home and is called *radius of gyration*. We consider a time-ordered set  $L_i = \{l_1, l_2, \dots, l_n\}$  of locations visited by  $N_i$  and  $H_i$  as its home. Formally,

$$\text{RADG}_{k,i} = \text{dist}(l_k, H_i),$$

where  $\text{dist}(l_k)_i$  represents the Euclidean distance between location  $l_k \in L_i$  and  $N_i$  home  $H_i$ . When considering all locations  $L_i$ , we obtain RADG-D. As observed in Figure 3.1(f), the agents of the Dartmouth trace tend to visit more often locations near their home’s vicinity instead of making long journeys.

### 3.3.2 Travel distance (TRVD)

When an agent moves from a location to another one, we usually describe that action as a travel, jump or flight [Karamshuk et al., 2011; Ekman et al., 2008]. The travel distance definition is very straightforward. It is the distance traveled between two consecutive locations [Ekman et al., 2008]. Formally, by considering a time-ordered set  $L_i = \{l_1, l_2, \dots, l_n\}$  of locations visited by  $N_i$  we have:

$$\text{TRVD}_{k,k+1} = \text{dist}(l_k, l_{k+1}),$$

where  $\text{dist}(l_k, l_{k+1})$  is the Euclidean distance between  $l_k$  and  $l_{k+1}$ . When considering all the travels performed by an agent, we obtain TRVD-D. As can be observed in Figure 3.1(g), TRVD-D presents a heavy-tailed behavior. This behavior indicates that agents perform short distance jumps more often than long distance ones. This observation is according to the least action principle of Maupertuis [1744], and used by Lee et al. [2012] to model SLAW.

TRVD is very related to RADG and the main difference between them is that RADG considers the travel distance according to the nodes' home, and not the distance between consecutive visited locations. The analysis of these properties helps us to understand how agents move from a place to another one. This information can be exploited, for instance, to improve message dissemination in vehicular and opportunistic networks.

## 3.4 Temporal properties

### 3.4.1 Visit time (VIST)

Visit time can be defined as the time spent at each location visited by an agent [Karamshuk et al., 2011]. Considering a time-ordered set  $V_i = \{v_1, v_2, \dots, v_n\}$  of visits  $v_k = (l_k, t_{ini}, t_{fin})$ ,  $l_k$  being the location visited at moment  $k$ , we have:

$$\text{VIST}_k = t_{fin} - t_{ini}.$$

Usually, agents tend to stop their movement at specific locations in order to accomplish some task (to buy groceries, for example). While most of the visits are short, sometimes agents stay more than the usual in a specific location. This behavior is represented by VIST-D and can be observed in Figure 3.1(i).



### 3.4.2 Travel time (TRVT)

In an analogous form to the travel distance, we define the travel time TRVT as the time an agent spends moving from one location to the consecutive one. Considering a time-ordered set  $V_i = \{v_1, v_2, \dots, v_n\}$  of visits  $v_k = (l_k, t_{ini}, t_{fin})$ ,  $v_k$  being the location visited at moment  $k$ , we have:

$$\text{TRVT}_{k,k+1} = t_{ini}^{k+1} - t_{fin}^k.$$

Considering all the agent’s visits, we plot TRVT-D in Figure 3.1(h). When compared to the *travel distance* property (described in Section 3.3.1), the statistical behavior matches the idea that long trips are less frequent, but demand more time to be completed, while short travels are faster and common, i.e., leading to a heavy tail distribution.

### 3.4.3 Entropy (ENTROPY)

It is known that in mobility scenarios agents develop social ties between themselves and those ties influence the way agents move from one place to another one [Shah and Rathod, 2014; Foroozani et al., 2014; Helgason et al., 2014a]. However, it is possible to state that agents also develop social ties with locations. In real scenarios, people tend to visit more often places that are somehow familiar to them or are visited by their family, friends or acquaintances.

In this work, we define an *entropy* property, which quantifies the amount of visits each location receives in a specific scenario. By considering a set of locations  $L = \{l_1, l_2, \dots\}$  we define:

$$\text{ENTROPY}_k = \sum_i i = 1^n \sum_k = 1^m \lambda(i, k),$$

where  $\lambda(i, k)$  is a function representing the number of times agent  $N_i$  visited location  $loc_k$ ,  $n$  is the total of agents and  $m$  the total of considered locations.

Intuitively, considering human mobility patterns, only few locations receive a high amount of visits (e.g., malls, subway stations and parks), while a large number of locations receive few visits (each home, for example). As shown in Song et al. [2010], ENTROPY-D is usually represented by a power-law distribution, a behavior observed in Figure 3.1(j) for the Dartmouth trace.



# Chapter 4

## MOCHA: MObility CHaracterization Framework

The purpose of this work is to design a framework to extract, analyze and compare mobility properties of different mobility scenarios. To the best of our knowledge, there is no study that analyzes, compares and classifies mobility scenarios in a detailed and broad way as we do in this work.

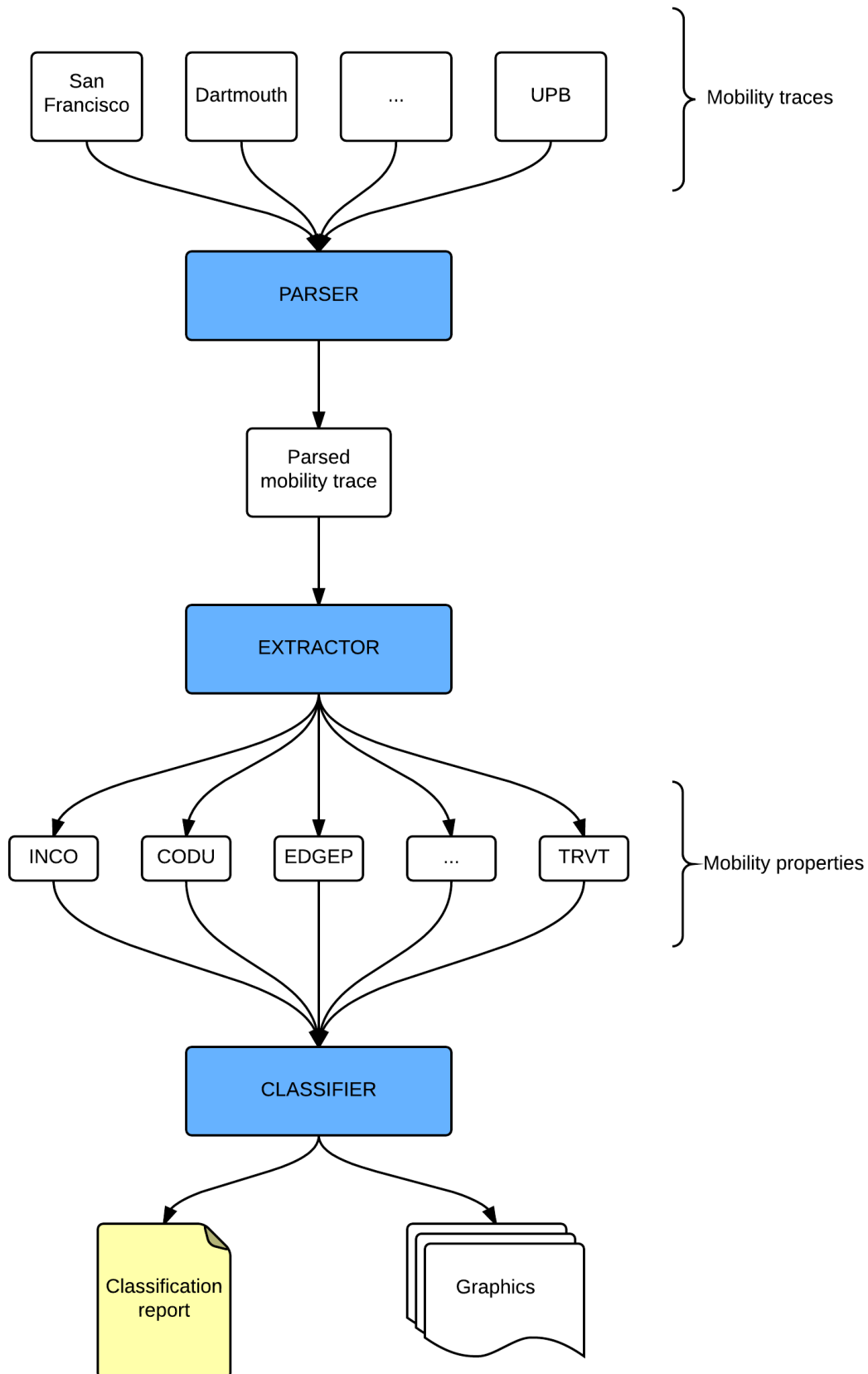
MOCHA encompasses the most commonly used mobility properties, dividing them into three groups as proposed by Karamshuk et al. [2011]: social, spatial and temporal.

The social properties considered in this framework are: inter-contact time (INCO), contact duration time (CODU), maximum contacts per hour (MAXCON), topological overlap (TOPO), edge persistence (EDGEPE) and the social correlation (SOCOR). The spatial properties are: the travel distance (TRVD) and radius of gyration (RADG). Finally, the temporal properties are: visit time (VIST), travel time (TRVT) and the locations entropy (ENTROPY). All these properties, but SOCOR were described previously in Chapter 3.

### 4.1 Overview

MOCHA is composed of three modules: *parser*, *extractor* and *classifier*. The *parser* is responsible for parsing datasets to MOCHA input form. The *extractor* analyzes the input trace and extracts all the considered properties. Finally, the *classifier* evaluates the properties related to statistical distributions and classifies them according to the defined taxonomy. Figure 4.1 shows how MOCHA is structured and the information from between modules.

Figure 4.1. MOCHA structure



The initial MOCHA entry is a mobility trace collected from a real environment or generated by a synthetic tool. Although these traces may contain any type of information, only a few are mandatory for MOCHA. Thus, the first step is to parse the trace into MOCHA's format. Once the mobility trace is parsed, we obtain a new trace in a defined form to be analyzed by MOCHA.

After that, the parsed mobility trace is used as input to the extractor. MOCHA's *extractor* generates a file for each considered property with all the extracted entries. Most of the properties are independent of each other and MOCHA allows the extraction of each property individually.

Finally, after all the mobility properties are extracted, MOCHA's *classifier* analyzes the data and generates a classification report with the category of each property. MOCHA can also generate graphics for each property according to its type. The graphics that MOCHA can generate are: complementary cumulative distribution function (CCDF), cumulative distribution function (CDF) and histogram.

The following sections will explain in detail how each MOCHA module works and which considerations were made along its design in order to be able to consider and analyze different scenarios.

## 4.2 The parser

In the design of MOCHA, we have to consider the heterogeneous nature of the traces found in the literature [Aschenbruck et al., 2011] and the different forms of data collection used regarding human mobility [Aschenbruck et al., 2011; Karamshuk et al., 2011; Treurniet, 2014]. In that direction, the first module of the framework is a parser, responsible for converting any trace to a common (internal) form with all the data needed to extract the statistical properties described in Chapter 3.

This step is important to ensure generality and extensibility of the framework. By using a standard input form, we allow anyone to evaluate a mobility trace with all the benchmarks.

In order to better explain how the parser works, we need to understand which types of mobility traces we can find in the literature and how we convert them to our standard form. To define our trace classification, we have analyzed 13 different real datasets publicly available. Darmouth, USC, UPB, Sassy and Cambridge are traces collected in different campi. Hospital, High school 2011, High school 2012 and Infectious are indoor traces of different natures, and Infocom and Hypertext are indoor traces collected in conferences. The number of agents and the total duration of each

dataset are described in Table 2.1.

Finally, San Francisco and Cologne are vehicular traces. These traces are presented in different forms related to the way they were collected. Some of them were collected using WLAN connection logs and others by using GPS or iMotes or other proximity detection devices. The differences between these traces are explained in Chapter 2.

Given these traces, we divided them into three categories: raw mobility traces, check-in traces and encounter traces. It is important to highlight that there might be other kinds of traces available, but we have chosen this three types because they are the most popular and almost every other kind of trace can be easily converted to one of them.

Each trace is composed of different entries representing a travel or an encounter that occurred in the considered scenario. By considering these three main types of mobility traces, we propose the utilization of the following common form for the entries in any dataset evaluated by MOCHA in order to maintain the homogeneity of our framework:

$$N_i \ N_j \ t_{fin} \ t_{ini} \ \delta t \ x_i \ y_i \ x_j \ y_j,$$

where  $N_i$  and  $N_j$  are two distinct agents,  $t_{ini}$  is the initial time of the encounter,  $t_{fin}$  is the final time of the encounter,  $\delta t$  is the encounter duration and  $(x, y)_i$  and  $(x, y)_j$  are  $N_i$  and  $N_j$  coordinates at the beginning of the encounter.

Our format is based on encounters because they can summarize a wide variety of statistical properties without the need to use too much memory or processing, simplifying the framework and the evaluation process. Considering that, we leave to the *parser* the responsibility to digest the input data that will be analyzed by the next modules.

### 4.2.1 Normalization

Before we can explain how the mobility properties are extracted and classified, we need to define the methodology used to convert any mobility trace into MOCHA's common form. This explanation is important to ensure reproducibility and, thus, the homogeneity of the evaluation. First, it is important to normalize the trace. The trace normalization will ensure the correct evaluation of each dataset and will avoid calculation errors.

There are three pieces of information that must be normalized: time, agent's IDs and geographic coordinates. To normalize time entries, we simply need to discover

what is the earliest time entry present in the trace and subtract it from all of the time values. To normalize the IDs we create a dictionary and map all the agents to a new ID value between 0 and  $n$ , where  $n$  is the total amount of distinct nodes minus one. The normalization of the coordinates is similar to the time normalization with one difference: it can only be done when the coordinates used on the trace are planar, not geographical. In traces that use GPS coordinates, it is not recommended to do a normalization considering that geographical coordinates are not linear. We will address this scenario later in this work.

Finally, all trace entries must be ordered by time. This ordering is necessary because some mobility properties to be extracted are time-dependent and it is easier to analyze them in an ordered way.

### 4.2.2 Raw mobility trace

The raw mobility trace is the most generic type of trace that can be found in the literature. It basically informs us the position of each agent in a specific time, but gives us no information about visited locations and social encounters. The most important aspect of this type of trace is the granularity of the time window, since it affects the precision of extracted properties. For example, a 1-second time window will give us a better information about social encounters than a 1-hour or even half-hour time window. However, the smaller the time window is, the more complex the entry process will be. Sometimes, an agent can move very short distances or even stay at the same place for several hours, making unnecessary to constantly track its behavior.

Raw traces are very popular among vehicular traces and even in some synthetic mobility generators such as NS-2 [Mccanne et al., 2007]. The main drawback of this type of trace, besides the difficulty of its processing, it is that we need to infer which locations are visited by each agent. Because raw mobility traces often give us specific coordinates, we need to infer where the social locations are, such as malls, train stations and schools. Considering that our framework is oriented to social mobility, there is no much we can obtain by evaluating the raw mobility only, so we need to use the mobility information to generate encounters and geographic locations instead of only individual coordinates.

The method to extract encounters from raw mobility traces, explained next, represents the steps and decisions we did to do the conversion between the raw mobility form and our common form. First, we use as starting point a trace  $T$  in which each line represents an event of a node and has the following format:

$$N_i \quad x_i \quad y_i \quad t,$$

where  $N_i$  is the agent's ID,  $(x, y)_i$  are the coordinates where the event occurred and  $t$  is the time of the event. It is important to point out that other data might be found in raw mobility traces, such as nodes' speed, for example, but we will focus on the information used by our method to generate our common trace form.

Assuming that  $T$  is the time ordered according to the preprocessing, we will analyze entry-by-entry using a graph  $G$  and a matrix  $M$  to support the encounter generation. We also define an encounter criterion stating that an encounter between two nodes occurs every time the Euclidean distance between them is equal or inferior to a threshold  $R$ . This assumption is valid according to how encounters happen in mobile network, where  $R$  is the radius of communication of each device [Helgason et al., 2014a]. The Euclidean distance  $d(p, q)$  is defined as follows:

$$d_{(p,q)} = d_{(q,p)} = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \dots + (q_n - p_n)^2} = \sqrt{\sum_{i=1}^n (q_i - p_i)^2},$$

where  $p = (p_1, p_2, \dots, p_n)$  and  $q = (q_1, q_2, \dots, q_n)$  are two points in the Euclidean  $n$ -space. For notation purposes, we define  $max_x$  and  $max_y$  as the maximum  $X$  and  $Y$  values encountered in the preprocessing step. In order to assist us in the encounter calculation, we have created a grid  $M$  of the size of the trace scenario using  $max_x$  and  $max_y$  in a way that each cell has a diagonal equal to  $R$ . By doing that, we guarantee that all nodes present at each cell are within the encounter range of each other. Formally, we define  $M$  as:

$$M = \left\{ c_{k,l} \mid \left( 0 \leq k \leq \frac{max_x}{R} \right) \wedge \left( 0 \leq l \leq \frac{max_y}{R} \right) \right\},$$

where  $c_{k,l}$  represents a cell in  $M$ . After creating the grid, we are ready to process the trace  $T$ . In this stage, we create a new node  $N_i$  in  $G$  to every node found in  $T$  that has not being already created, and we store the node coordinates at the time of its creation. Considering the coordinates  $(x, y)_i$  registered at the trace entry, we allocate  $N_i$  in its respective cell  $c_{k,l}$  where  $k = \frac{x_i}{R}$  and  $l = \frac{y_i}{R}$ .

Once  $N_i$  its allocated to  $c_{k,l}$ , we retrieve all nodes  $N_j$  present in the range  $c_{k-1,l-1}$  to  $c_{k+1,l+1}$  to verify if the Euclidean distance between  $N_i$  and  $N_j$  is equal or inferior to  $R$ . If so, we create an edge  $E_{(i,j)}$  in  $G$  between  $N_i$  and  $N_j$ , with a value equal to the time  $t$  present in the current evaluated entry of  $T$ . We also consider as neighbors, or  $NG_i$ , all nodes  $N_j$  discovered in this step. If the evaluated entry in the previous step represents the first occurrence of node  $N_i$  in the trace, we consider that  $NG_i$  is empty. However, if there is a previous entry with  $N_i$ , it is probable that  $NG_i$  is not empty. Therefore, we need to evaluate if all the previously existent nodes in  $NG_i$  are still within the range of  $N_i$ . To do that, every time that entry is evaluated, we compare



the current set of neighbors with the previous one and compute all nodes that are not neighbors anymore.

Once we identify that a node  $N_k$  is not a neighbor of  $N_i$  anymore, we search for the edge  $E_{(i,k)}$ , whose value is equal to the time  $t_p$  when the encounter between  $i$  and  $k$  began, and register a new entry in our log according to the standard proposed previously:

$$N_i \ N_k \ t_c \ t_p \ (t_c - t_p) \ x_i \ y_i \ x_k \ y_k,$$

where  $N_i$  and  $N_k$  are the nodes IDs,  $t_c$  is the current entry time,  $t_p$  is the time when the edge between  $i$  and  $k$  was created,  $(x_i, y_i)$  are the current  $N_i$  coordinates and  $(x_k, y_k)$  are the  $N_k$  coordinates when the encounter began. By repeating the described procedure for every entry in  $T$ , we can successfully parse a raw mobility trace to our common form. This process is fully described in Algorithm 4.1.

---

**Algorithm 4.1.** Raw mobility trace parser algorithm

---

```
# entry = Ni xi yi ti
for entry in T:
    if G.has_node(Ni):
        for Nk in neighbors(Ni):
            if not euclidean(Ni, Nk):
                generate_entry(Ni, Nk, ti, G[Ni][Nj[time]],
                    (ti - G[Ni][Nj[time]]), xi, yi, xk, yk)
                G.remove_edge(Ni, Nk)
    else:
        G.add_node(Ni)
        for Nj in neighbors(Ni):
            if euclidean(Ni, Nj):
                G.add_edge(Ni, Nj, time = tc)
```

---

### 4.2.3 Check-in traces

The check-in trace is a type of trace that can be easily extracted from social networks, such as Foursquare, Twitter, Facebook or even Instagram. However, other situations can generate check-in traces, such as the associations in all the access points of a university campus, association of communication tower in telephonic networks and even dollar bill tracking. This type of trace is much simpler to parse, but it is by nature a very poor trace. If we only consider the arrival moment at a specific location,

check-in traces give us no information about how long the node stayed there and which locations the node might have visited between consecutive check-ins.

Because of the information gap check-in traces have, we need to infer how the social encounters happen in every location, how long each node stayed in the location, how long they took to travel from a location to another and so on. The usual format of this type of trace is:

$$N_i \quad l_i \quad t,$$

where  $N_i$  is the agent's ID,  $l_i$  is the location's ID and  $t$  is the time of the check-in. It is important to point out that most of the times there are no geographical coordinates available and only the arrival time is informed. Later on, we will explain how this lack of information affects the property extraction, but, for now, we will focus on the parsing.

Assuming that  $T$  is time ordered according to the preprocessing, we will analyze entry-by-entry using a graph  $G$  and a dictionary  $L$  indexed by the location IDs. First, we create a dictionary of locations. For each entry in  $T$  we include  $N_i$  in the location  $L_i$  once a check-in occurs. At the check-in moment, we create an edge  $E_{(i,j)}$  in  $G$  between  $N_i$  and all nodes present in  $L_i$ .

Considering that the checkout time is not provided, we define the values  $max_t$  as the longest duration each node can stay in a specific location and  $mov_t$  as the average movement time between two locations. If the same node  $N_i$  happens to check-in in another location before the expiration of  $max_t$ , we create a new log entry for all edges  $E_{i,j}$  with the following format:

$$N_i \quad N_j \quad (t_c - mov_t) \quad t_p \quad (t_c - t_p) \quad 0 \quad 0 \quad 0 \quad 0,$$

where  $N_i$  and  $N_j$  are the nodes IDs,  $t_c$  is the current entry time minus  $mov_t$ ,  $t_p$  is the time when the edge between  $i$  and  $j$  was created and the zeros represent the lack of information regarding the coordinates of the locations. If the coordinates of the location  $L_i$  happen to be available, the zeros can be replaced with  $L_x$ ,  $L_y$ ,  $L_x$ ,  $L_y$ , respectively. Otherwise, if  $max_t$  is reached before  $N_i$  performs another check-in, the checkout from  $L_i$  will be automatically executed creating a new log entry for all edges  $E_{i,j}$  with the same format but the value of  $t_c$  will be equal to  $(t_p + max_t)$  instead of  $(t_c - mov_t)$ . This process is fully described in Algorithm 4.2.

---

**Algorithm 4.2.** Check-in mobility trace parser algorithm

```
# entry = Ni L ti
```

```

for entry in T:
    if L.has_node(Ni):
        if ti > (L(Ni,t) + max_t):
            for Nj in L.nodes:
                generate_entry(Ni,Nj, (ti - movt), G[Ni][Nj][time],
                    (ti - G[Ni][Nj][time]), 0, 0, 0, 0)
                G.remove_edge(Ni,Nj)
                L.remove_node(Ni)
    else:
        L.add_node(Ni, ti)
        for Nj in L.nodes:
            G.add_edge(Ni,Nj,time = ti)

```

---

#### 4.2.4 Contact traces

A contact trace, which we will interchangeably call encounter trace, can be considered the simplest to analyze and the richest trace because it gives us the core information about a mobility scenario: the social interactions. Usually, encounter traces have the initial and the final time of each encounter between every pair of nodes and, sometimes, they offer us the position where each encounter happened. However, only the coordinates are not enough to know the location where the encounter occurred, i.e., a mall or a bakery, for instance. Considering that, we also require some inferring to extract all the evaluated properties, but usually we do not require extensive parsing to convert them to the common form used by the framework.

By presenting two different nodes IDs, such as  $N_i$  and  $N_j$ , the initial time of the contact and its duration, contact traces require no more than a value rearrangement to be represented according to MOCHA requirements. For example, some contact traces present the positions of the initial and final times swapped and some of them do not have the duration of the encounter previously calculated, making necessary to run over all the entries to add that specific field to them. This procedure is described in Algorithm 4.3.

**Algorithm 4.3.** Contact mobility trace parser algorithm

---

```

# entry = Ni Nj xi xj yi yj ti tf
for entry in T:
    generate_entry(Ni, Nj, tf, ti, (tf - ti), xi, yi, xj, yj)

```

---

## 4.3 The extractor

Once we generate a new trace in MOCHA common format, we can proceed to extract the desired properties. The goal of the *extractor* is to extract from the trace all the mobility properties described in Chapter 3. After the extraction, we will be able to analyze the statistical curve generated by the aggregation of all the occurrences of each property. For example, we need to compute all the contacts between each pair of nodes and their duration to be able to analyze CODU-D.

However, before we describe the extraction process itself, it is important to highlight some assumptions that we need to consider. First, not all traces analyzed by MOCHA have all properties available for extraction. For example, a check-in trace without venues' coordinates makes it impossible to calculate properties such as TRVD or TRVT, and a contact trace without coordinates give us only the information needed to calculate the social properties, but none of the spatial nor the temporal properties. However, it is always possible to extract all the social properties. Considering that, all methods described below can only be used if the entry trace meets the requirements associated to each property extraction.

In addition, there are some properties whose extraction relies on some parameters configured according to the needs of the experiment. For example, all contacts are evaluated based on a contact radius of 50 meters, but this parameter can be tuned to 5 or even 500 meters, if necessary. The value of 50 meters is considered as default because well-known traces, such as Dartmouth, are based on proximity with Wi-Fi routers and a contact occurs whenever two different users are connected to the same router. It is common in the literature to define the radius of a Wi-Fi network to 50 meters or so [Dimatteo et al., 2011].

Finally, by considering that the extraction step is executed only for MOCHA parsed traces, all traces should be ordered by time and each entry be in the format described in Section 4.2. Those are some of the general assumptions considered by MOCHA. Any other assumption related to a specific property will be described during the extraction process itself.

### 4.3.1 Social properties

#### 4.3.1.1 Inter-contact time (INCO)

The INCO property is extracted by registering the time between two consecutive contacts regarding each specific pair of nodes, as previously explained in Section 3.2.1. We define  $T$  as a time-ordered parsed trace and create a graph  $G$  in which we insert all nodes

present in  $T$  with no edges between themselves. Once  $G$  is created, MOCHA evaluates all the encounters in  $T$  and for each encounter  $e_{i,j} = \{N_i, N_j, t_f, t_i, \delta t, x_i, y_i, x_j, y_j\}$  two different cases might occur:

1. **Encounter beginning:** There is no edge  $E_{i,j}$  in  $G$ , indicating  $N_i$  and  $N_j$  have never encountered before. In that case, an edge  $E_{i,j}$  is created with a value equal to  $t_f$ , registering the last moment of the contact between  $N_i$  and  $N_j$ .
2. **Encounter end:** There is an edge  $E_{ij}$  in  $G$ , which indicates that  $N_i$  and  $N_j$  already encountered before. In that case, an INCO entry is generated by subtracting the current entry  $t_i$  with the value registered in  $E_{ij}$ , the previous edge is removed and a new one is created with the current entry  $t_f$  value. In Algorithm 4.4, we describe the INCO extraction process with a generic algorithm.

---

**Algorithm 4.4.** INCO algorithm

---

```
# entry = Ni Nj xi xj yi yj ti tf (tf-ti)
for entry in T:
    if G.has_edge(Ni, Nj):
        generate_entry( ti - G[Ni][Nj][time] )
        G[Ni][Nj][time] = tf
    else:
        G.add_edge(Ni, Nj, time = tf)
```

---

#### 4.3.1.2 Contact duration (CODU)

The CODU property is extracted by registering the duration of each contact between any pair of nodes. For example, if nodes  $A$  and  $B$  encountered from time  $t_i$  to  $t_f$ , a CODU entry will be generated with the value  $t_f - t_i$ . However, by considering the common trace format used by MOCHA, we only need to get the encounter duration ( $\delta t$ ) value from all entries in  $T$  to register the CODU distribution. This process can be observed in Algorithm 4.5.

---

**Algorithm 4.5.** CODU algorithm

---

```
# entry = Ni Nj tf ti (tf-ti) xi xj yi yj
for entry in T:
    generate_entry( tf - ti )
```

---

### 4.3.1.3 Maximum contacts per hour (MAXCON)

The MAXCON property is extracted by registering the number of contacts that occurred during each hour of the trace  $T$  duration. In order to compute it, MOCHA first determines the duration of  $T$ . Giving that all entries are registered in seconds, MOCHA converts the total of seconds of the simulation to hours and creates an array  $H$  with a size equivalent to the number of hours present in  $T$ . Once  $H$  is created, MOCHA analyzes each entry and increases a counter for each slot in  $H$  regarding the hour that every contact begins.

It is important to highlight that a single encounter can start at a specific hour and extend for more than one hour, making necessary to increase different slots in  $H$ . However, considering that the MAXCON purpose is to represent social regularity among the days, we decided to increase only the counter associated to the hour of the beginning of each contact. By considering the nature of the real datasets used in this work, we realize that most of the time, the beginning of the contact is more significant than its end. For example, check-in traces always have the initial time of the check-in but very rarely they present the check-out time. The complete MAXCON extraction process can be observed in Algorithm 4.6.

**Algorithm 4.6.** MAXCON algorithm

---

```
# entry = Ni Nj xi xj yi yj ti tf (tf-ti)
H.size = total_time/3600      # total of hours
for entry in T:
    h = convert_to_hour(ti)
    H.at(h) = H.at(h) + 1
for hour in H.size:
    generate_entry(hour)
```

---

### 4.3.1.4 Encounter regularity (EDGEPE)

The EDGEPE property is extracted by calculating the persistence of the encounters between each pair of nodes during the trace time. In order to do so, we define a time window  $W$  that is the maximum time a pair of nodes can encounter with each other. For example, if the evaluated trace considers 15 days of encounters and we want to calculate a daily EDGEPE, then  $W = 15$ . If instead of days we want to consider an hourly EDGEPE, then  $W = 15 \times 24 = 360$ .

MOCHA standard configuration uses a daily window and calculates the  $W$  value automatically by using the maximum  $t$  value encountered during the preprocessing

stage. We evaluate all entries in  $T$  and create an edge  $E_{i,j}$  every time a pair of nodes encounters for the first time. The initial value of each edge is always 1.

Considering  $w_0, w_1, w_2, \dots, w_n$  where  $w_0$  is the first time window of the trace (e.g., a day) and  $w_n$  is the last time window, we increment the edge  $E_{i,j}$  once for each day for the nodes  $N_i$  and  $N_j$ . Even when both nodes encounter more than once a day, we can only increase the edge value by 1 for each day.

Once we evaluate all the entries in  $T$ , we will have a graph  $G$  in which edges have the maximum number of days that each pair of nodes encountered with each other. Finally, we evaluate each edge  $e$  in  $G$  and divide its value by  $w_n$  to obtain the final EDGE-P value for each pair of nodes. In Algorithm 4.7, we observe the detailed description for the EDGE-P extraction process.

---

**Algorithm 4.7.** EDGE-P algorithm

---

```
# entry = Ni Nj xi xj yi yj ti tf (tf-ti)
for entry in T:
    enc_day = encounter_day(entry)
    if G.has_edge(Ni,Nj) and G[Ni][Nj][day] != enc_day:
        G[Ni][Nj][per] = G[Ni][Nj][per] + 1
        G[Ni][Nj][day] = enc_day
    else:
        G.add_edge(Ni,Nj,day = enc_day, per = 1)
for edge in G.edges():
    generate_entry(edge[per])
```

---

#### 4.3.1.5 Topological overlap (TOPO)

The TOPO property is extracted by calculating the social overlap of each agent's neighbors in  $G$ . To do so, we evaluate all entries in  $T$  and create an edge  $E_{i,j}$  for all nodes encountering at least once. Once we evaluate all entries in  $T$ , we iterate within all edges  $E_{i,j}$  of  $G$  calculating the intersection of the neighbors of  $N_i$  and  $N_j$ . We observe the TOPO extraction algorithm in Algorithm 4.8.

---

**Algorithm 4.8.** TOPO algorithm

---

```
# entry = Ni Nj xi xj yi yj ti tf (tf-ti)
for entry in T:
    G.add_edge(Ni,Nj)
for edge in G.edges():
    topo = find_common(neighbors(Ni), neighbors(Nj))
```

---

```
generate_entry(topo)
```

---

#### 4.3.1.6 Social correlation (SOCOR)

The calculation of the SOCOR value is trivial considering that it only consists in calculating the Pearson’s correlation coefficient between the TOPO and the EDGEF values. As we are going to see below in this work, SOCOR presents high values in almost all real traces (over 0.8) and low values in synthetic traces.

#### 4.3.2 Spatial properties

As previously described, not all traces have geographical information available, making impossible to extract any spatial or even temporal properties (all the temporal properties are related to space and time while spatial properties are only related to space). However, even traces having explicit coordinates for each encounter rarely have information about the venue where the encounter occurred.

In that case, before we consider properties like RADG or TRVD, we need to define how to find locations in a trace, given that geographical information consists in consecutive coordinates instead of venues. Check-in traces, on other side, give us information about the venues themselves but rarely present geographical coordinates, making impossible to calculate travel distances, for example.

Therefore, each input trace needs to be properly analyzed to determine which properties can or cannot be extracted from it. For this topic, we will assume a mobility trace with geographical coordinates of each encounter. In addition, we present an algorithm to estimate locations, which will be used as reference to extract all the spatial and temporal properties.

##### 4.3.2.1 Finding locations

Let us assume a trace  $T$  in which every entry has the geographical coordinates of the encounter. In order to estimate the locations of  $T$ , MOCHA uses a Voronoi algorithm to group all the  $T$  coordinates in  $n$  different centroids, which will be used as the locations (or venues) of the trace. The value of  $n$  can be defined according to the user’s needs via MOCHA configuration file. We suggest that  $n$  be a value related to the total area of the evaluated scenario, for example,  $n = (\max_x \times \max_y) / R^2$ , where  $R$  is the agent’s communication radius.

Once the venues are calculated,  $T$  is analyzed once more to include the venue in which each point “checked-in”. To do so, each coordinate is analyzed to determine to



which venue it belongs to and, then, a new log entry is generated in a new trace  $V$  with the agent ID and its current venue. A new entry will be generated only when the node leaves the last venue to a new one. This process is described in Algorithm 4.9.

---

**Algorithm 4.9.** Finding locations algorithm

---

```
# entry = Ni Nj xi xj yi yj ti tf (tf-ti)
n = max_venues
for entry in T:
    locations.add(xi, yi)
    locations.add(xj, yj)
venues = voronoi(locations, n)
# the following for statement is repeated for Ni and Nj entries
  where N is presented
for entry in T:
    v = closer_venue(venues, N, x, y)
    if not (N[last_venue]):
        N[last_venue] = v
        N[time] = t
    if (N[last_venue] != v):
        generate_entry(N, v, (tf - N[time]))
```

---

Once we have all venues visited in  $T$ , we can use  $V$  to extract our spatial properties using the processes described in Sections 4.3.2.2 and 4.3.2.3.

#### 4.3.2.2 Radius of gyration (RADG)

The RADG property is extracted using a two-step algorithm, where the first one consists in determining each agent’s “home”, and the second step consists in calculating the distance between each visited location and the agent’s “home”. In a real scenario, each agent would indeed have a house that could be used to calculate this property. However, considering that none of the real traces used in this study actually present the house’s coordinates of each node, we consider the “house” as the most visited venue by each node.

To do that, we create a vector  $A$  with a size equal to the number of agents present in  $T$ . Inside each position of  $A$ , we store a vector  $L$  with all locations present in  $V$ . Once we do that, we analyze  $V$  to compute all the venues  $v$  visited by each node  $N_i$  and increment the respective position  $vi$  in  $L_i$  according to the trace entry.

Once we compute all the visits in  $A$ , we will have the amount of visits that each agent did to every venue and, consequently, we can determine its home  $h$  by looking up

for the most visited venue. Finally, after determining  $h$ , we iterate over  $L$  calculating the distance between  $h$  and all venues, and registering that value as a RADG entry. Algorithm 4.10 describes the RADG extraction process.

---

**Algorithm 4.10.** RADG algorithm

---

```
# entry = N x y tf ti tf-ti v
for entry in V:
    A[N][v] = A[N][v] + 1
for entry in V:
    home = most_visited(A[N])
    if home != v:
        radg = euclidean(home, v)
        generate_entry(radg)
```

---

### 4.3.2.3 Travel distance (TRVD)

The TRVD extraction is very similar to the RADG one. However, when calculating RADG, we register the distance between all visited locations and the agent's home, whereas TRVD extractions consist in calculating the distance between consecutive visited locations.

To do so, we use a vector  $A$  where we register the last venue visited by each agent and start iterating through  $V$ . Once we find a venue  $v$  for agent  $N$ , we register that venue into  $A_n$ , if there was no previous venue registered. In case a previous venue exists in  $A_n$ , we calculate the distance between the current venue and the new one and register a TRVD entry. Algorithm 4.11 details this process.

---

**Algorithm 4.11.** TRVD algorithm

---

```
# entry = N x y tf ti tf-ti v
for entry in V:
    if A[N]:
        generate_entry(euclidean(A[N], v))
    A[N] = v
```

---

### 4.3.3 Temporal properties

The last group of properties that can be extracted from mobility traces are the *temporal properties*. It is important to highlight that all the temporal properties are related not

only to time but also to space. Considering that, the following properties require the venues to log  $V$ , which was described in Section 4.3.2.

#### 4.3.3.1 Visit time (VIST)

The VIST property is extracted by registering the time that each agent spent at each location. Once we generate the trace locations using the method described in Section 4.3.2, the VIST entries are automatically calculated and saved in  $V$ . To put the VIST entries into a separated log, all we have to do is to iterate through  $V$  and register the  $t$  value of each entry, as described in Algorithm 4.12.

**Algorithm 4.12.** VIST algorithm

---

```
# entry = N x y tf ti tf-ti v
for entry in V:
    generate_entry(tf - ti)
```

---

#### 4.3.3.2 Travel time (TRVT)

The TRVT property is extracted by registering the difference between an agent's departure time at a location and the arrival time at a consecutive one. This extraction process is done in an analogous way to TRVD, but instead of registering the venue position we register the departure time, i.e.,  $t_f$ . This process is described in Algorithm 4.13.

**Algorithm 4.13.** TRVT algorithm

---

```
# entry = N x y tf ti tf-ti v
for entry in V:
    if A[N]:
        generate_entry(ti - A[N])
    A[N] = tf
```

---

#### 4.3.3.3 Entropy (ENTROPY)

The ENTROPY property is extracted by calculating the amount of visits that each location received and dividing it by the total of visits that all locations received. To do so, we iterate through  $V$  and create a vector  $L$  with all locations present in  $V$ . During the iteration, we increase  $L_v$  by one every time a new visit occurs in  $v$  and increment our total counter  $c$  by one to compute the total amount of visits. Once we analyze all

entries in  $V$ , we have to divide all the values in  $L$  by  $c$  and register all the ENTROPY entries. Algorithm ?? describes the ENTROPY extraction process in details.

---

**Algorithm 4.14.** ENTROPY algorithm

---

```
# entry = N x y tf ti tf-ti v
for entry in V:
    L[v] = L[v] + 1
    c = c + 1
for l in L:
    generate_entry(1/c)
```

---

## 4.4 The classifier

As previously described, most of the synthetic trace generators validate their models by comparing generated traces with real ones. This evaluation is usually done by a visual graphical comparison or a comparison of network metrics, as described in Section ?. In addition, each study compares only a limited set of mobility properties. Considering the nature of different datasets and the variety of mobility properties, we propose a methodology that can be generalized to different types of mobility traces and consider all the mobility properties described in Chapter 3.

To quantitatively compare these properties, we propose the use of the Akaike's Information Criterion (AIC). The AIC is based on the Maximum Likelihood Estimation (MLE) to determine the type of curve that best fits each property distribution followed by a categorization of each distribution.

A distribution can be separated into two parts: head and tail. In the case of a heavy-tailed distribution, the tail is the part that contains most of the values. However, in this work, we will consider the tail of the distribution all values beyond the median of the distribution. MOCHA uses the tail of the distribution to classify the mobility properties.

The technique of maximum likelihood is the core of parameter estimation in AIC. Maximum likelihood is used to estimate the parameters of a model, given observations of the variables in the model [Stigler, 2007]. Once we have the best estimate for the model parameters, we can verify which model provides the best fit to the empirical data.

Several methods are commonly used for comparing models. If two models have the same set of parameters, the most suitable one is simply the one that has the

higher likelihood between the two. However, if two models have different numbers of parameters, then they may be compared using Akaike information criterion (AIC):

$$\text{AIC} \equiv -2\log(L) + 2(p + 1),$$

where  $L$  is the likelihood and  $p$  is the number of parameters of the model. The best model has the lowest AIC [Hilborn and Mangel, 1997].

The use of MLE and the Akaike's information criterion (AIC) to determine the type of each distribution is the basis of our proposed framework MOCHA. In this way, we can formally and quantitatively say if a given property distribution has, for instance, a heavy-tail or not. This is necessary because a comparison based on visualization can be misleading, since sometimes distributions can look as a power-law but, however, can be better fitted by an exponential curve (or vice-versa) [Clauset et al., 2009]. With the Akaike's information criterion (AIC), we dismiss any doubt about the best fit for any distribution that can be extracted from both real and synthetic traces. We try to fit any presented distribution with several different known distributions: gamma, weibull, exponential, normal, lognormal, pareto and loglogistic.

Once that we fit a distribution, we classify them according to three different categories regarding the tail of the distribution: pareto tail (or heavy tail) as **PAR**, exponential tail as **EXP** and lognormal tail as **LOG**. Gamma, Weibull, exponential and normal curves are categorized as **EXP**; lognormal curves are categorized as **LOG**; and Pareto and log logistic curves are categorized as **PAR** [Nair et al., 2013]. Whenever a distribution could not be extracted we categorize it as **NVA** (no values found).

At the end of the comparison, MOCHA generates a report with the classification of each property, the MAXCON correlation value and the SOCOR value too. Besides the comparison itself, MOCHA's *classifier* is also responsible for generating all the graphics regarding the statistical distributions discussed in this work. It was using MOCHA itself that all the curves of this work were generated.



# Chapter 5

## Results

In this chapter we present the results obtained with MOCHA by analyzing all the traces described in Tables 2.1 and 2.2. We extract all the possible mobility properties from each trace, according to the restrictions described in the previous chapters. After the properties extraction, we classify them according to their statistical distribution. By performing a sanity check, we show how the results obtained by MOCHA match previous studies. Finally, we use  $k$ -means algorithm and Principal Component Analysis to cluster all the used datasets according to their similarities regarding the classification performed by MOCHA.

### 5.1 Analysis

Figure 5.1 compares all the selected traces, where the top segment represents the real datasets and the bottom represents the synthetic ones. By using the taxonomy proposed in Chapter 3, we divide all statistical distributions in four different categories: EXP (exponential tail), PAR (pareto tail), LOG (lognormal tail) and NVA. This last classification is used in the cases when a particular trace does not have the data needed to extract the given property. Only MAXCON and SOCOR are represented by numerical values, i.e., the Pearson's correlation coefficient

As previously defined in Section 3.2.3, MAXCON is directly related to social *regularity*. However, when we compare the values presented in Figure 5.1, we observe that, while real traces have significant MAXCON values, synthetic traces have values close to 0. In real traces MAXCON's average is around 0.4, while on synthetic traces the average drops to 0.07. This is an important observation to highlight the fact that even if synthetic traces can reproduce social interactions at some level, most of them

**Figure 5.1.** Classification of all properties according to MOCHA

TRACE	INCO	CODU	MAXCON	TOPO	EDGEPE	SOCOR	RADG	TRVD	TRVT	VIST	ENTROPY	TYPE
DARTMOUTH	PAR	LOG	0.73	EXP	PAR	0.98	PAR	PAR	LOG	EXP	PAR	CAMPUS
CAMBRIDGE	PAR	PAR	0.09	LOG	LOG	0.97	NVA	NVA	NVA	NVA	NVA	CAMPUS
SASSY	PAR	PAR	0.01	LOG	LOG	0.87	NVA	NVA	NVA	NVA	NVA	CAMPUS
USC	LOG	PAR	0.82	LOG	PAR	0.80	NVA	NVA	NVA	NVA	NVA	CAMPUS
UPB	LOG	LOG	0.01	LOG	PAR	0.74	NVA	NVA	NVA	NVA	NVA	CAMPUS
INFOCOM	LOG	PAR	0.24	LOG	PAR	0.94	NVA	NVA	NVA	NVA	NVA	CONFERENCE
HYPertext	PAR	LOG	0.50	PAR	PAR	0.87	NVA	NVA	NVA	NVA	NVA	CONFERENCE
HOSPITAL	PAR	LOG	0.65	LOG	PAR	0.95	NVA	NVA	NVA	NVA	NVA	INDOOR
HIGH SCHOOL 11	PAR	LOG	0.47	LOG	PAR	0.90	NVA	NVA	NVA	NVA	NVA	INDOOR
HIGH SCHOOL 12	PAR	LOG	0.43	LOG	PAR	0.91	NVA	NVA	NVA	NVA	NVA	INDOOR
INFECTIOUS	LOG	LOG	0.51	LOG	PAR	0.84	NVA	NVA	NVA	NVA	NVA	INDOOR
SAN FRANCISCO	PAR	PAR	0.78	EXP	PAR	0.93	LOG	PAR	EXP	PAR	LOG	VEHICULAR
COLOGNE	LOG	EXP	0.00	PAR	PAR	0.00	NVA	NVA	NVA	NVA	NVA	VEHICULAR
WDM	LOG	PAR	0.95	LOG	PAR	0.89	LOG	LOG	EXP	PAR	PAR	URBAN
SLAW 5	LOG	LOG	0.01	PAR	PAR	0.48	PAR	PAR	EXP	LOG	PAR	-
SLAW 35	LOG	LOG	0.05	PAR	PAR	0.43	PAR	LOG	EXP	PAR	LOG	-
SUBWAY	LOG	PAR	0.00	LOG	PAR	0.66	PAR	LOG	LOG	PAR	PAR	PEDESTRIAN
OSTER 90	LOG	PAR	0.00	PAR	PAR	0.66	PAR	LOG	LOG	PAR	PAR	PEDESTRIAN
OSTER 70	LOG	PAR	0.00	PAR	PAR	0.74	PAR	LOG	LOG	PAR	LOG	PEDESTRIAN
OSTER 50	LOG	PAR	0.00	PAR	PAR	0.73	PAR	LOG	EXP	PAR	LOG	PEDESTRIAN
OSTER 40	EXP	PAR	0.00	PAR	PAR	0.73	PAR	LOG	LOG	PAR	PAR	PEDESTRIAN
OSTER 30	EXP	PAR	0.00	PAR	PAR	0.78	PAR	LOG	EXP	PAR	LOG	PEDESTRIAN
SWIM 88	LOG	PAR	0.03	LOG	PAR	0.0	LOG	PAR	EXP	EXP	PAR	-
SWIM 85	LOG	PAR	0.04	LOG	PAR	0.0	LOG	PAR	EXP	LOG	PAR	-
SWIM 83	LOG	PAR	0.11	LOG	PAR	0.0	LOG	PAR	EXP	PAR	PAR	-
SWIM 58	LOG	PAR	0.03	LOG	PAR	0.0	LOG	PAR	EXP	PAR	PAR	-
SWIM 55	LOG	PAR	0.02	LOG	PAR	0.0	LOG	PAR	EXP	EXP	LOG	-
SWIM 53	LOG	PAR	0.00	LOG	PAR	0.0	PAR	PAR	EXP	PAR	LOG	-
SWIM 38	LOG	PAR	0.01	EXP	PAR	0.84	PAR	PAR	EXP	PAR	LOG	-
SWIM 35	LOG	PAR	0.02	EXP	PAR	0.78	LOG	PAR	EXP	PAR	LOG	-
SWIM 33	LOG	PAR	0.07	EXP	PAR	0.45	LOG	PAR	EXP	EXP	LOG	-

fail to reproduce the *regularity* present in a daily routine. The reason why WDM does not fail to reproduce *regularity* is because WDM is a routine-based mobility model.

Most of the mobility generators and mobility models described in Chapter 2 were validated through a graphical comparison of the mobility properties of the synthetic traces generated by them. By observing INCO and CODU columns in Figure 5.1, we realize that synthetic and real scenarios have similar results for both columns. In some cases, such as USC and any SWIM dataset, we could even affirm that those scenarios are equivalent. However, by observing INCO and CODU in addition to MAXCON column, we realize that even the statistical distribution of properties, such as INCO and CODU, can be similar between real and synthetic scenarios, but only in real scenarios the agents move according to a routine.

In an analogous way to INCO, CODU and MAXCON, we observe EDGEPE, TOPO and SOCOR. There are real and synthetic traces with the same classification results for EDGEPE and TOPO, such as Infocom and SWIM88. However, while in Infocom those results are highly correlated, i.e., 0.94, in SWIM 88 there is no correlation at



all. Also, in real traces MAXCON’s average is around 0.82, while on synthetic traces the average drops to 0.45. These observations match with the fact that, even when mobility generators can reproduce statistical distributions, they fail to reproduce the underlying social relations that rule human mobility.

Regarding *temporal* and *spatial* properties, we observe that most of the real datasets have several properties classified as **NVA**. This happens because those traces have no information available about the coordinates of each encounter, making impossible to calculate any position-related property. However, when considering all the properties, obtained through analysis of available data, most of the statistical distributions are heavy-tailed distributions, such as PAR and LOG. This observation validates MOCHA when considering previous studies showing that all the mobility properties considered in this work are represented by heavy-tailed distributions, as discussed in Chapters 2 and 3.

## 5.2 Sanity Check

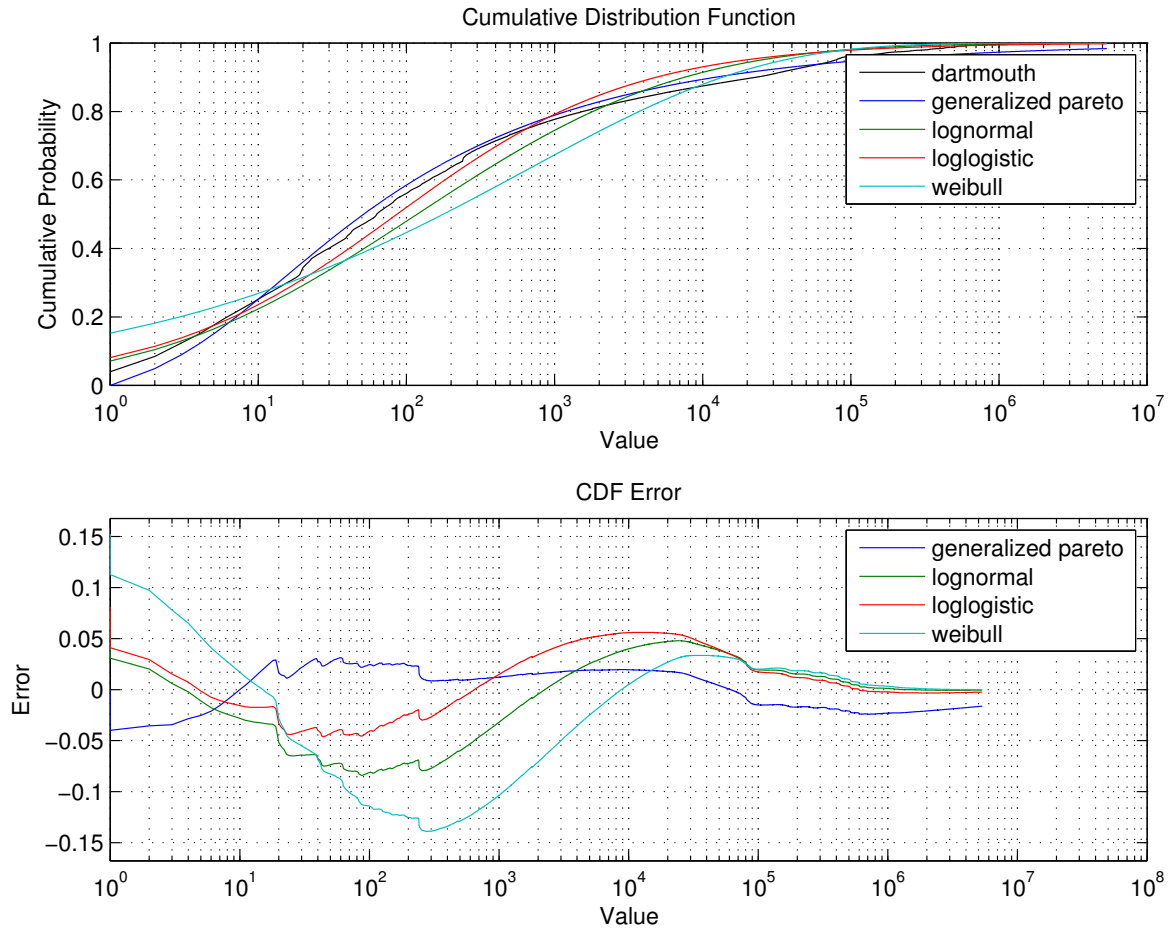
Before we start discussing the results described in the previous section, let us take a look at Figure 3.1 where we can see three different distributions generated with random variables: exponential 3.2(a), pareto 3.2(b) and lognormal 3.2(c). As we can see, the distributions are visually similar, but when we try to fit them with our framework the results are EXP, PAR and LOG respectively.

In Figures 3.1(a) and 3.1(b) we observe INCO and CODU distributions for Dartmouth, a real dataset widely used for comparison and validation of mobility generation tools. If we try to visually fit those distributions to our categories, we might conclude that the contact duration distribution is more likely a LOG fit but it is not completely clear if the inter-contact time is better fitted by an exponentially tailed distribution or a pareto tailed distribution. When looking at Figure 5.1, we know that the best fit is actually PAR.

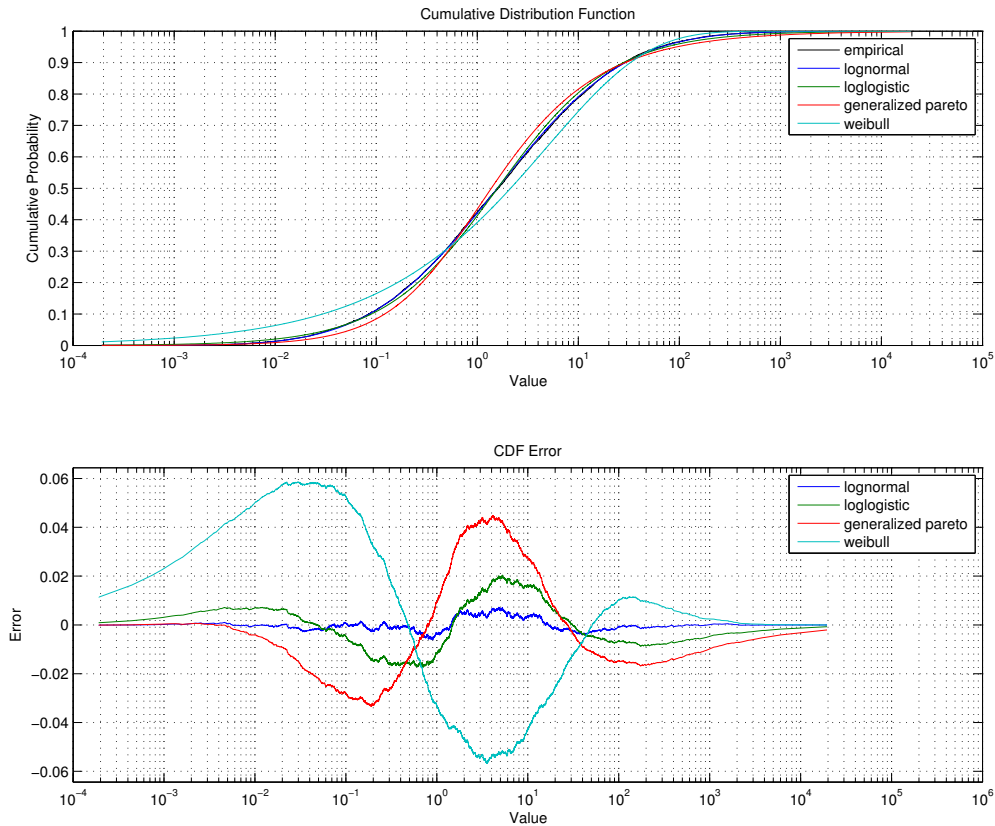
Unlike previous studies, MOCHA does not rely on visual comparison to classify mobility properties. In Figure 5.2, we observe the curve fitting to determine in which category the INCO property from Dartmouth dataset is classified. By looking at the first graphic, we observe how misleading and confusing a visual comparison between different curves can be. Despite the *generalized pareto* to appears closer to Dartmouth data, there is no warranty that it is in fact a *loglogistic* or even a *lognormal*. Only when we look at the second figure we can have a clearer view that the cumulative error is actually lower for a *generalized pareto* than to other distributions. The visual

comparison, however, can become more tricky in cases such as the one presented in Figure 5.3, where it is not possible to visually determine whether the SLAW5 data better fits a *loglogistic* curve or a *lognormal*. Once again, by observing the cumulative error, it is clear which distribution best fit the data.

**Figure 5.2.** Dartmouth INCO fitting and AIC error



Another important result is related to the methodology used to extract mobility properties. Lee et al. [2012] perform an experiment using the INCO property and the AIC test to classify and compare mobility datasets. Moreover, their results do not match the ones presented in this work. While they classified INCO with PAR, we classified it as LOG. One possible reason to that difference could be the set of parameters used to generate mobility data. However, they also evaluate the Dartmouth dataset and classify its INCO property with EXP, while we obtained PAR. Considering that we both used the same dataset, the most probable cause for this different might be related to the extraction process of the mobility properties. By using MOCHA, this problem will no longer exist because all the datasets can be evaluated using the same

**Figure 5.3.** SLAW5 INCO fitting and AIC error

methodology.

Despite some of the results obtained with MOCHA do not match previous studies, some of the premises considered by most of the previous proposals regarding the mobility generation can be validated by MOCHA. When considering the synthetic traces presented in Figure 5.1, we observe that all the datasets generated by SWIM, SLAW and WDM models respect the power-law distributions in properties such as inter-contact time, contact duration, travel distance, radius of gyration and entropy, as defined by Lee et al. [2012]. Also, SWIM [Kosta et al., 2014] proposes its model by assuming that the travel time between consecutive locations is always the same. This consideration is based on the premise that we tend to perform short trips by foot and longer trips using different types of transportation (e.g., bike, car, plane and boat). The SWIM premises regarding these properties are revealed in Figure 5.1, where all the generated traces have exponentially bounded tails (EXP) in the TRVT column. However, this same type of behavior can be observed in traces generated by SLAW, Legion Studio, Working Day Model and in the San Francisco vehicular trace. Also,

when considering the property radius of gyration, SWIM also gives each node a home and generates travels to another locations by using power-law distribution, a behavior observed by MOCHA in the analysis of the distributions presented in Figure 5.1.

Considering the mobility scenario, some traces, such as Cambridge and Sassy, have very similar results to all the evaluated properties and both represent campi scenarios. However, Dartmouth, USC and UPB also represent campi scenarios but the distributions found by MOCHA do not match among themselves. This observation is important to show how even related scenarios might present different characteristics. This finding matches the observations made in Vaz de Melo et al. [2013]. Both PAR and LOG represent heavy tailed distributions, the difference between them indicates that their agent’s behaviors are not equivalent. This work does not consider the implications of this specific difference and future research will be needed to answer this question.

Regarding the INCO and CODU columns, which are the most used properties to benchmark mobility models, we can observe that indeed most of the synthetic traces present a heavy tailed behavior. However, if the real purpose of synthetic models is to imitate reality, Figure 5.1 shows us that it is naive to compare two scenarios by only considering a limited set of properties. If that is the case, we could choose whatever properties we find suitable to state the similarity of different models and scenarios when their real differences rely on the overall picture.

While observing the TRVD and TRVT properties, despite being very related properties (intuitively travel distance should be somehow connected to the travel time), these properties only present an equal behavior in some of the pedestrian traces generated by Legion Studio. It is important to highlight that all results for the spatial and temporal properties can be also related to the location extraction algorithm that was presented in Section 4.3

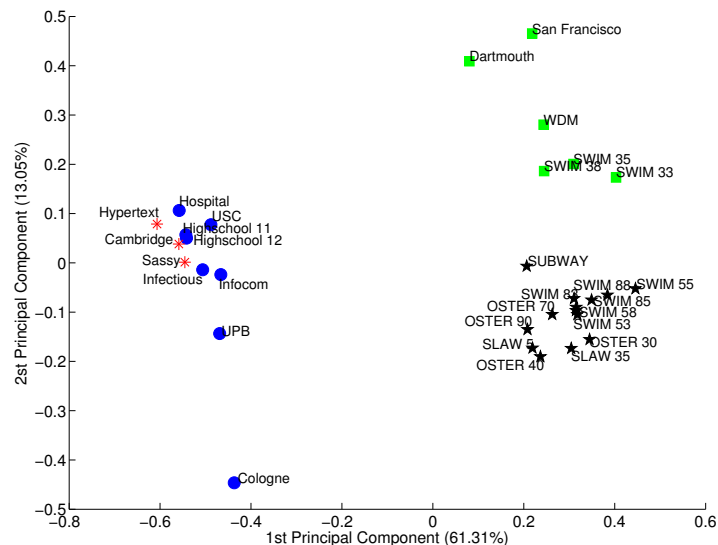
### 5.3 Classification of traces

In Figure 5.1, we have datasets from different campi, conferences and indoor environments. We could expected that campi traces were similar all the time, but our evaluation shows that this premise is not necessarily valid all the time. By using a clustering algorithm, we will show how traces can be grouped by their similarities according to the results observed in Table 5.1. However, to use the *k-means* algorithm, we need first to define which value of  $k$  best attends our purpose. To do so, we use the *gap statistic* method [Tibshirani et al., 2000] and obtained  $k = 4$ . By using Principal

Component Analysis (PCA), we display our clusters in a bi-dimensional space, in order to better visualize our results.

In Figure 5.4, we show the  $k$ -mean clustering for all the evaluated traces, to better understand which similarities are present between them. To generate this graphic, we considered all the values present in Figure 5.1, including the NVAs and all the datasets. As we can observe, most of the synthetic traces are grouped in the black cluster while the real datasets are divided among the other three clusters. The only cluster with both synthetic and real datasets is the green one, presenting three SWIM scenarios. Besides the real/synthetic division that is observable in Figure 5.4, there is no other group such as *campus*, *indoor*, *conference* or *vehicular* that is clearly clustered with its most similar scenario. When comparing Figure 5.4 with Figure 5.1, we observe that only Dartmouth and San Francisco have a classification value for all the mobility properties, while the other real datasets present only *social* properties.

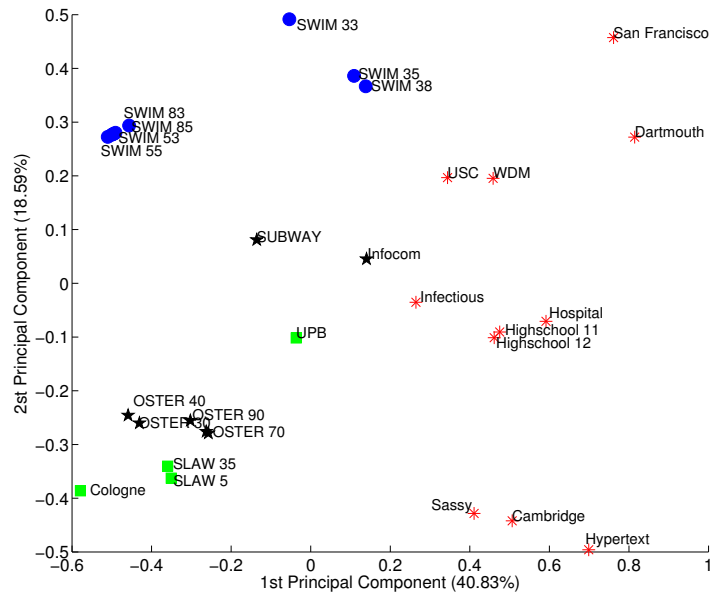
**Figure 5.4.**  $k$ -means ( $k = 2$ ) clustering of all evaluated datasets according to MOCHA using all available properties from all datasets



In Figure 5.5, we observe a new  $k$ -means clustering but that considers only *social* properties. As we can observe, the clustering changes. In Figure 5.5, all the SWIM datasets are clustered together and none of them is in the same cluster of Dartmouth or San Francisco, which remained clustered together.

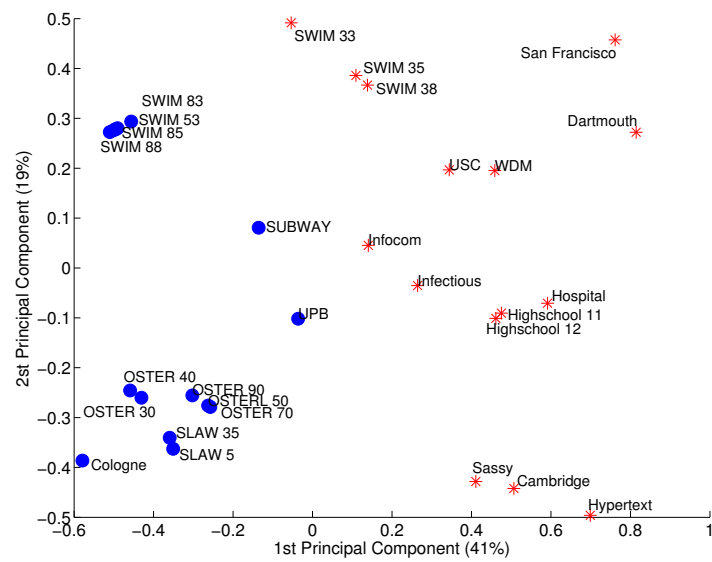
We also observe that all traces generated by Legion Studio (i.e., the OSTER traces) are grouped together. The same occurs with both SLAW scenarios. When considering only *social* properties, we observe a more logical clustering of the datasets. Almost all *campi* are clustered together and each generation tool, except SWIM, is

**Figure 5.5.**  $k$ -means ( $k = 4$ ) clustering of all evaluated datasets according to MOCHA using only social properties from all datasets



associated with a different set of real scenarios. These results indicate that, by considering only *social* properties, the division between synthetic and real datasets is very clear. If we run our  $k$ -means algorithm again using  $k = 2$ , we will obtain two clusters in which one of them is composed by 70% of all the real datasets. This observation is presented in Figure 5.6 and indicates how far away mobility generators are from generating realistic mobility data.

**Figure 5.6.**  $k$ -means ( $k = 2$ ) clustering of all evaluated datasets according to MOCHA using only social properties from all datasets







# Chapter 6

## Conclusions and Future Work

### 6.1 Conclusions

In this work, we presented MOCHA, a practical framework that performs comparison between mobility traces in a very simple and elegant way. By proposing a standard mathematical methodology to compare distributions and mobility properties, we create the possibility to quantify the differences among them and to better understand whether the lack of information can compromise the utilization of a specific dataset for a specific application.

By using MOCHA, we observe that the considered mobility properties affect the clustering results, causing the impression that two datasets can be related, when in fact they are very different. As we could see, there are some studies comparing mobility models by choosing an arbitrary set of mobility properties and concluding that they are equivalent or similar. MOCHA showed the importance of considering all the possible mobility properties to validate the similarity between mobility scenarios.

Our framework presents a solution for all the stages after the data collection until its analysis. By presenting a taxonomy and definition for the different types of mobility traces found in the literature, MOCHA's *parser* defines a methodology that enables the translation of any mobility trace to a common form. This form is based on a social encounter approach and can be analyzed to extract mobility properties. By extracting 11 different mobility properties, MOCHA covers the most used mobility properties present in the literature to validate and compare different scenarios. Notice that most of these properties were already used somehow in previous studies, but MOCHA is the first framework that actually put them all together to enable a fair and complete comparison. Finally, MOCHA's *classifier* analyzes the extracted properties individually and classifies them according to their statistical distributions. This methodology is

well known in the literature and is vastly used to analyze and understand the behavior of a specific parameter to enable a deeper comparison. MOCHA can also generate the related graphics of each scenario.

By implementing and validating MOCHA, we showed how apparently similar scenarios can represent completely different realities according to the nature of each trace. We also set an enhanced point of view to the human mobility research field. MOCHA is validated by some known premises found in the literature regarding mobility properties. However, MOCHA also presents new insights towards new the directions that can be taken to better understand the real differences between each mobility scenario. By using a quantitative approach, MOCHA shows how graphical comparison using statistical curves can be misleading and cannot be considered a reliable methodology to compare mobility properties. Besides the mobility properties extraction and classification, MOCHA also proposes a clustering methodology that can be used to determine similarities among different datasets and to benchmark synthetic models with real mobility scenarios.

## 6.2 Future Work

As future work, it is important to try to understand what is the real impact of each property classification in the whole scenario. MOCHA shows how mobility properties can be classified into three different categories but it lacks to explain what is the meaning of each property classification. For example, what is the difference between an exponentially bounded inter-contact time and a pareto bounded one? Does it mean that there are no real social ties in the dataset or is it just a difference caused by the type of scenario (e.g., vehicular versus campus)? One way to start exploiting these differences is to use network metrics to help the understanding of how a different statistical distribution might (or not) affect the network performance.

It is also important to start exploiting new mobility properties that can be found in all mobility traces to increase the relationship between datasets based on their properties. By understanding them, it would be interesting to develop a tool able to generate specific scenarios such as campus, vehicular, indoor and urban.

Besides, MOCHA can also improve in terms of its own algorithms, specifically the one used to infer venues in mobility traces. One important question that should be answered in the future is how to tune MOCHA parameters to extract the most accurate information of each trace (e.g., communication radius and number of venues).

Another interesting developing for this work is to be able to understand which pa-

parameters determine the difference between similar scenarios. Intuitively, we understand that different campi have different dynamics (the same applies for different conferences, cities and others). However, it will be useful to determine which properties or parameters are specifically responsible for these differences to be able to model and reproduce them.



# Bibliography

- Alshanyour, A. and Baroudi, U. (2008). Random and realistic mobility models impact on the performance of bypass-aodv routing protocol. In *Wireless Days, 2008. WD '08. 1st IFIP*, pages 1–5.
- Aschenbruck, N., Frank, M., Martini, P., and Tolle, J. (2004). Human mobility in manet disaster area simulation - a realistic approach. In *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, pages 668–675. ISSN 0742-1303.
- Aschenbruck, N., Munjal, A., and Camp, T. (2011). Trace-based mobility modeling for multi-hop wireless networks. *Computer Communications*, 34(6):704 – 714. ISSN 0140-3664.
- Bai, F., Sadagopan, N., and Helmy, A. (2003). Important: a framework to systematically analyze the impact of mobility on performance of routing protocols for ad-hoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 2, pages 825–835 vol.2. ISSN 0743-166X.
- Batabyal, S. and Bhaumik, P. (2014). Delay-overhead trade-offs in mobile opportunistic network using ttl based restricted flooding. In *Applications and Innovations in Mobile Computing (AIMoC), 2014*, pages 9–14.
- Bezerra, R. L., Campos, C. A. V., and Moraes, L. F. M. d. (2009). Uma proposta de tecnica para o ajuste de modelos de mobilidade em redes ad hoc e questionamentos sobre a adequacao dos parametros envolvidos com base em dados reais. In *Simposio Brasileiro de Redes de Computadores, 2009*.
- Bigwood, G., Rehunathan, D., Bateman, M., and Bhatti, S. (2011). CRAW-DAD data set st\_andrews/sassy (v. 2011-06-03). Downloaded from [http://crawdad.org/st\\_andrews/sassy/](http://crawdad.org/st_andrews/sassy/).

- Boldrini, C., Conti, M., and Passarella, A. (2007). Impact of social mobility on routing protocols for opportunistic networks. In *World of Wireless, Mobile and Multimedia Networks, 2007. WoWMoM 2007. IEEE International Symposium on a*, pages 1–6.
- Chaintreau, A., Hui, P., Crowcroft, J., Diot, C., Gass, R., and Scott, J. (2007). Impact of human mobility on opportunistic forwarding algorithms. *Mobile Computing, IEEE Transactions on*, 6(6):606–620. ISSN 1536-1233.
- Ciobanu, R. I. and Dobre, C. (2012). CRAWDAD data set upb/mobility2011 (v. 2012-06-18). Downloaded from <http://crawdad.org/upb/mobility2011/>.
- Clauset, A., Shalizi, C. R., and Newman, M. E. J. (2009). Power-Law Distributions in Empirical Data. *SIAM Review*, 51(4):661–703. ISSN 0036-1445.
- Dimatteo, S., Hui, P., Han, B., and Li, V. (2011). Cellular traffic offloading through wifi networks. In *Mobile Adhoc and Sensor Systems (MASS), 2011 IEEE 8th International Conference on*, pages 192–201. ISSN 2155-6806.
- Ekman, F., Keränen, A., Karvo, J., and Ott, J. (2008). Working day movement model. In *Proceedings of the 1st ACM SIGMOBILE Workshop on Mobility Models, MobilityModels '08*, pages 33–40, New York, NY, USA. ACM.
- Fischer, D., Herrmann, K., and Rothermel, K. (2010). Gesomo x2014; a general social mobility model for delay tolerant networks. In *Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on*, pages 99–108. ISSN 2155-6806.
- Foroozani, A., Gharib, M., Hemmatyar, A., and Movaghar, A. (2014). A novel human mobility model for manets based on real data. In *Computer Communication and Networks (ICCCN), 2014 23rd International Conference on*, pages 1–7.
- Fournet, J. and Barrat, A. (2014). Contact patterns among high school students. *PLoS ONE*, 9(9):e107878.
- Helgason, O., Kouyoumdjieva, S., and Karlsson, G. (2010). Does mobility matter? In *Wireless On-demand Network Systems and Services (WONS), 2010 Seventh International Conference on*, pages 9–16.
- Helgason, O., Kouyoumdjieva, S., and Karlsson, G. (2014a). Opportunistic communication and human mobility. *Mobile Computing, IEEE Transactions on*, 13(7):1597–1610. ISSN 1536-1233.

- Helgason, O., Kouyoumdjieva, S., and Karlsson, G. (2014b). Opportunistic communication and human mobility. *Mobile Computing, IEEE Transactions on*, 13(7):1597–1610. ISSN 1536-1233.
- Henderson, T., Kotz, D., and Abyzov, I. (2004). The changing usage of a mature campus-wide wireless network. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking, MobiCom '04*, pages 187–201, New York, NY, USA. ACM.
- Hilborn, R. and Mangel, M. (1997). *The Ecological Detective: Confronting Models with Data*. Monographs in population biology. Princeton University Press. ISBN 9780691034973.
- Isella, L., Stehle, J., Barrat, A., Cattuto, C., Pinton, J., and Van den Broeck, W. (2011). What’s in a crowd? analysis of face-to-face behavioral networks. *Journal of Theoretical Biology*, 271(1):166–180. ISSN 0022-5193.
- jen Hsu, W. and Helmy, A. (2008). CRAWDAD data set usc/mobilib (v. 2008-07-24). Downloaded from <http://crawdad.org/usc/mobilib/>.
- Karagiannis, T., Le Boudec, J.-Y., and Vojnovi?, M. (2010). Power law and exponential decay of intercontact times between mobile devices. *Mobile Computing, IEEE Transactions on*, 9(10):1377–1390. ISSN 1536-1233.
- Karamshuk, D., Boldrini, C., Conti, M., and Passarella, A. (2011). Human mobility models for opportunistic networks. *Communications Magazine, IEEE*, 49(12):157–165. ISSN 0163-6804.
- Karamshuk, D., Boldrini, C., Conti, M., and Passarella, A. (2014). Spot: Representing the social, spatial, and temporal dimensions of human mobility with a unifying framework. *Pervasive and Mobile Computing*, 11(0):19 – 40. ISSN 1574-1192.
- Kosta, S., Mei, A., and Stefa, J. (2014). Large-scale synthetic social mobile networks with swim. *Mobile Computing, IEEE Transactions on*, 13(1):116–129. ISSN 1536-1233.
- Kouyoumdjieva, S. T., Ólafur Ragnar Helgason, and Karlsson, G. (2014). CRAWDAD data set kth/walkers (v. 2014-05-05). Downloaded from <http://crawdad.org/kth/walkers/>.

- Lee, K., Hong, S., Kim, S. J., Rhee, I., and Chong, S. (2012). Slaw: Self-similar least-action human walk. *Networking, IEEE/ACM Transactions on*, 20(2):515–529. ISSN 1063-6692.
- Maupertuis, P. L. M. (1744). Accord des differentes lois de la nature qui avaient jusqu’ici paru incompatibles. *Memoires de l’Academie des Sciences*, page 417.
- Mccanne, S., Floyd, S., and Fall, K. (2007). ns2 (network simulator 2). <http://www-nrg.ee.lbl.gov/ns/>.
- Meghanathan, N. and Milton, L. (2009). A simulation based performance comparison study of stability-based routing, power-aware routing and load-balancing on-demand routing protocols for mobile ad hoc networks. In *Wireless On-Demand Network Systems and Services, 2009. WONS 2009. Sixth International Conference on*, pages 3–10.
- Munjal, A., Camp, T., and Navidi, W. (2010). Constructing rigorous manet simulation scenarios with realistic mobility. In *Wireless Conference (EW), 2010 European*, pages 817–824.
- Nair, J., Wierman, A., and Zwart, B. (2013). The fundamentals of heavy-tails: Properties, emergence, and identification. *SIGMETRICS Perform. Eval. Rev.*, 41(1):387–388. ISSN 0163-5999.
- Piorowski, M., Sarafijanovic-Djukic, N., and Grossglauser, M. (2009). CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from <http://crawdad.org/epfl/mobility/>.
- Sandulescu, G., Niruntasukrat, A., and Charnsripinyo, C. (2013). Resource-aware capacity evaluation for heterogeneous, disruption-tolerant networks. In *Proceedings of the 9th Asian Internet Engineering Conference, AINTEC ’13*, pages 57–64, New York, NY, USA. ACM.
- Scott, J., Gass, R., Crowcroft, J., Hui, P., Diot, C., and Chaintreau, A. (2006). CRAWDAD data set cambridge/haggle (v. 2006-01-31). Downloaded from <http://crawdad.org/cambridge/haggle/>.
- Shah, M. and Rathod, J. (2014). Efficient scheduling algorithm for query processing in opportunistic sensor network under human mobility model. In *Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference -*, pages 401–405.



- Song, C., Koren, T., Wang, P., and Barabasi, A.-L. (2010). Modelling the scaling properties of human mobility. *Nat Phys*, 6(10):818–823.
- Song, L. and Kotz, D. F. (2007). I. In *Proceedings of the Second ACM Workshop on Challenged Networks*, CHANTS '07, pages 35–42, New York, NY, USA. ACM.
- Stigler, S. M. (2007). The Epic Story of Maximum Likelihood. *Statistical Science*, 22(4):598–620. ISSN 0883-4237.
- Thakur, G. and Helmy, A. (2013). Cobra: A framework for the analysis of realistic mobility models. In *Computer Communications Workshops (INFOCOM WKSHPS), 2013 IEEE Conference on*, pages 145–150.
- Tibshirani, R., Walther, G., and Hastie, T. (2000). Estimating the number of clusters in a dataset via the gap statistic. 63:411–423.
- Treurniet, J. (2014). A taxonomy and survey of microscopic mobility models from the mobile networking domain. *ACM Comput. Surv.*, 47(1):14:1–14:32. ISSN 0360-0300.
- Uppoor, S. and Fiore, M. (2012). Mobicom 2011 poster: Vehicular mobility in large-scale urban environments? *SIGMOBILE Mob. Comput. Commun. Rev.*, 15(4):55–57. ISSN 1559-1662.
- Vanhems, P., Barrat, A., Cattuto, C., Pinton, J.-F., Khanafer, N., RÃ©gis, C., Kim, B.-a., Comte, B., and Voirin, N. (2013). Estimating potential infection transmission routes in hospital wards using wearable proximity sensors. *PLoS ONE*, 8(9):e73970.
- Vastardis, N. and Yang, K. (2014). An enhanced community-based mobility model for distributed mobile social networks. *Journal of Ambient Intelligence and Humanized Computing*, 5(1):65–75. ISSN 1868-5137.
- Vaz de Melo, P. O., Viana, A. C., Fiore, M., Jaffrès-Runser, K., Le Mouel, F., and Loureiro, A. A. (2013). Recast: Telling apart social and random relationships in dynamic networks. In *Proceedings of the 16th ACM International Conference on Modeling, Analysis & Simulation of Wireless and Mobile Systems*, MSWiM '13, pages 327–334, New York, NY, USA. ACM.
- Zyba, G., Voelker, G., Ioannidis, S., and Diot, C. (2011). Dissemination in opportunistic mobile ad-hoc networks: The power of the crowd. In *INFOCOM, 2011 Proceedings IEEE*, pages 1179–1187. ISSN 0743-166X.