COMPARISON OF DOMAIN-INDEPENDENT AND DOMAIN-SPECIFIC LOCATION PREDICTORS WITH CAMPUS-WIDE WI-FI MOBILITY DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

MÜCAHİT KARAKOÇ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

SEPTEMBER 2010

Approval of the thesis:

COMPARISON OF DOMAIN-INDEPENDENT AND DOMAIN-SPECIFIC LOCATION PREDICTORS WITH CAMPUS-WIDE WI-FI MOBILITY DATA

submitted by **MÜCAHİT KARAKOÇ** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Canan Özgen _____
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Adnan Yazıcı _____
Head of Department, **Computer Engineering**

Assoc. Prof. Dr. Ahmet Coşar _____
Supervisor, **Computer Engineering Dept., METU**

Dr. Murat Ali Bayır _____
Co-supervisor, **Ask.com, IAC**

Examining Committee Members:

Prof. Dr. Adnan Yazıcı
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. Ahmet Coşar
Computer Engineering Dept., METU _____

Prof. Dr. Faruk Polat
Computer Engineering Dept., METU _____

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering Dept., METU _____

Assoc. Prof. Dr. İbrahim Körpeoğlu
Computer Engineering Dept., Bilkent University _____

Date: **06.09.2010**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last Name: MÜCAHİT KARAKOÇ

Signature :

ABSTRACT

COMPARISON OF DOMAIN-INDEPENDENT AND DOMAIN-SPECIFIC LOCATION PREDICTORS WITH CAMPUS-WIDE WI-FI MOBILITY DATA

Karakoç, Mücahit

M.Sc., Department of Computer Engineering

Supervisor : Assoc. Prof. Dr. Ahmet Coşar

Co-Supervisor : Dr. Murat Ali Bayır

September 2010, 57 pages

In mobile computing systems, predicting the next location of a mobile wireless user has gained interest over the past decade. Location prediction may have a wide-range of application areas such as network load balancing, advertising and web page prefetching. In the literature, there exist many location predictors which are divided into two main classes: domain-independent and domain-specific. Song et al. compare the prediction accuracy of the domain-independent predictors from four major families, namely, Markov-based, compression-based, PPM and SPM predictors on Dartmouth's campus-wide Wi-Fi mobility data. As a result, the low-order Markov predictors are found as the best predictor. In another work, Bayir et al. propose a domain-specific location predictor (LPMP) as the application of a framework used for discovering mobile cell phone user profiles.

In this thesis, we evaluate LPMP and the best Markov predictor with Dartmouth's campus-wide Wi-Fi mobility data in terms of accuracy. We also propose a simple method which improves the accuracy of LPMP slightly in the location prediction part of LPMP. Our results show that the accuracy of the best Markov predictor is better than that of LPMP in total. However, interestingly, LPMP yields more accurate results than the best Markov predictor

does for the users with the low prediction accuracy.

Keywords: Location Prediction, Domain-Independent Location Predictors, Domain-Specific Location Predictors, WLAN, Wi-Fi

ÖZ

KAMPÜS-ÇAPLI KABLOSUZ AĞ(WI-FI) HAREKET VERİSİ ÜZERİNDE ALANDAN BAĞIMSIZ VE ALANA ÖZGÜ YER KESTİRİCİLERİNİN KARŞILAŞTIRILMASI

Karakoç, Mücahit
Yüksek Lisans, Bilgisayar Mühendisliği
Tez Yöneticisi : Doç. Dr. Ahmet Coşar
Ortak Tez Yöneticisi : Dr. Murat Ali Bayır
Eylül 2010, 57 sayfa

Gezgin bilgi işleme sistemlerinde, hareketli bir kullanıcının sonraki yerinin kestirimi geçen on yılda ilgi topladı. Yer kestirimi ağ yük dengeleme, reklamcılık ve örün belgesi önceden getirme gibi geniş çapta uygulama alanlarında yer bulabilir. Literatürde çok sayıda yer alan kestiriciler iki ana sınıfa ayrılırlar: alandan bağımsız ve alana özgü. Song ve ark. Markov-tabanlı, sıkıştırma-tabanlı, kısmî eşleştirmeyle kestirim (PPM) ve örneklenmiş örüntü eşleştirmesi (SPM) olarak adlandırılan dört ana aileye mensup alandan bağımsız kestiricilerin kestirim doğruluklarını Dartmouth'un kampüs-çaplı Wi-Fi hareket verisi üzerinde kıyaslar. Neticesinde, düşük dereceli Markov kestiricileri en iyi kestirici olarak bulunur. Diğer bir çalışmada ise Bayır ve ark. hareketli cep telefonu kullanıcı profillerini ortaya çıkarmak için kullanılan bir çerçeve yapının uygulaması olarak alana özgü bir yer kestirici (LPMP) önermektedirler.

Bu tezde, Dartmouth'un kampüs-çaplı Wi-Fi hareket verisini kullanarak LPMP ve en iyi Markov kestiricisini doğruluk açısından değerlendiriyoruz. Ek olarak, LPMP'nin doğruluğunu kısmen iyileştiren basit bir yöntemi LPMP'nin yer kestirimi kısmında öneriyoruz. Sonuçlarımız, en iyi Markov kestiricisinin doğruluğunun toplamda LPMP'ninkinden daha iyi olduğunu gösteriyor. Fakat, ilginç bir şekilde düşük kestirim doğruluğuna sahip kullanıcılarda LPMP

Markov kestiricisinden daha doğru sonuçlar üretmektedir.

Anahtar Kelimeler: Yer Kestirimi, Alandan Bağımsız Yer Kestiricileri, Alana Özgü Yer Kestiricileri, Kablosuz Ağ, Wi-Fi

To my family...

ACKNOWLEDGMENTS

I would like to thank my supervisor Assoc. Prof. Dr. Ahmet Coşar for his endless support and encouragement throughout my thesis.

I would like to express my grateful thanks to my co-supervisor Dr. Murat Ali Bayır for his generous help and valuable comments during thesis study.

I also thank my jury members Prof. Dr. Adnan Yazıcı, Prof. Dr. Faruk Polat, Prof. Dr. İsmail Hakkı Toroslu and Assoc. Prof. Dr. İbrahim Körpeoğlu for their time and useful comments.

I am grateful to my team leader Dr. M. Nedim Alpdemir from TÜBİTAK UEKAE / İLTAREN for his tolerance during thesis study.

My special thank goes to my friend Sercan Gök for his continuous help and guidance.

Lastly, but by no means least, I am very grateful to my family for their support throughout my life. I owe my deepest gratitude to my mother, Emine Taşcı, for her selfless devotion. Without her help and patience, I would not complete this study.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ALGORITHMS

ALGORITHMS

LIST OF ABBREVIATIONS

AP Access-Point

AW All-W Approach

FW Fallback-W Approach

LPMP Location Predictor Via Mobility Profiler Framework

MAC Media Access Control

MRA Median Running Accuracy

PPM Prediction by Partial Matching

SPM Sampled Pattern Matching

TS Time-Slice

CHAPTER 1

INTRODUCTION

Wireless networks allow a more flexible communication model than traditional wireline networks since the user is not limited to a fixed physical location [31]. As a result, mobile wireless communication has showed a significant increase in popularity over the past decade. Wireless communication devices are becoming increasingly ubiquitous and the number of devices people are willing to carry around is increasing rapidly [25].

The wireless devices are generating many huge mobility data all over the world in each second. Inevitably, those data have attracted the attention of the researchers working on the data mining field. Then, many works have focused on how mobility path information can be extracted from the mobility data.

Mobility path information can be used to predict the future location(s) of a user. This is called *location prediction*. The results of location prediction may be input to a wide-range of application areas such as network load balancing [21], advertising [20] and web page prefetching [35]. With a location predictor, the applications from the different areas can provide services or information based on the user's next location. For example, a student with a smart-phone takes a photo of his classmates after a class and wishes to print it. The printing application in his phone may suggest the printers near his current and next predicted location. While the student walks across the campus, his photo has already printed by the selected printer [30].

Obviously, accuracy which means predicting the user's next location correctly is an important issue in measuring the benefit of a location predictor. In literature, there are several location predictors which have been proposed to obtain better accuracy. Cheng et al. [10] divide the location predictors into two broad classes: domain-independent and domain-specific. The

domain-independent location predictors rely on just the past location movements (location history) of a user to predict his/her next location. They infer some statistical results using different methods such as Markov analysis and text-compression algorithms. Then, with the help of those results, the user's next location is predicted. In contrast, the domain-specific location predictors do not rely on just the location history. The history is partitioned based on the semantics of the location prediction domain. For example, if the user stays at a location a significant amount of time, that location may be a delimiter for the location history. The location coordinates and the geometry of user motion can be used as well as the time context in the example [30, 10].

Song et al. [30] compare the prediction accuracy of the several location predictors from four major families of domain-independent predictors on Dartmouth's campus-wide Wi-Fi mobility data [18]. The four major families are [30]:

- **Markov Family:** The *order-k* (or "$O(k)$") Markov predictor assumes that the user's next location can be predicted from the current *context* that is the most recent k symbols in the location history.
- **LZ Family:** These are the predictors based on incremental parsing algorithm by Ziv and Lempel [36] often used for text compression.
- **Prediction by Partial Matching (PPM):** A data compression scheme, often used in text compression [11], blends a set of different order context models, which are built as $O(k)$ Markov models, from 0 to k.
- **Sampled Pattern Matching (SPM):** Unlike $O(k)$ Markov predictors using a fixed value of the context length k, SPM predictors determine the context length by a fixed fraction of the longest context that has been previously seen.

Darmouth's campus-wide Wi-Fi mobility data [18] contains two-year record of the user movements with 6,202 users and 12,218,093 moves. The number of moves for different users has a wide-range with a median of 494 and a maximum of 188,479. Using this data, Song et al. [30] reached that the simple low-order Markov predictors worked as well or better than the more complex compression-based predictors. As a result, they identified $O(2)$ Markov $fallback$ predictor as the best for Dartmouth's data.

Bayir et al. [3] designed and implemented a framework, the **Mobility Profiler**, for discovering the mobility profiles of the mobile cell phone users starting from cell based raw location data. We will call that type of data as the *timed-location* history. Basically, the Mobility Profiler partitions the timed-location history into *mobility path*s and generates *mobility pattern*s from mobility paths taking the topology into consideration. Finally, it finds the patterns with the maximum length *supported* by a required number of the mobility paths. Those patterns are called as *maximal frequent patterns*. However, for the location prediction, we are not interested in finding the maximal frequent patterns as we will see. Therefore, we removed this and similar unnecessary parts from our implementation of Mobility Profiler. In addition, we implemented a simple method we propose in the location prediction phase. Consequently, we call our implementation as *Location Predictor Via Mobility Profiler Framework (LPMP)*. Since LPMP benefits the time-context -while partitioning the location history- in addition to the location history, we classify it as a domain-specific predictor.

As a part of this thesis, we implemented LPMP and modified the implementation of a ready-to-use Markov predictor slightly. We compared LPMP and the best Markov predictor in terms of the prediction accuracy. Due to the exponential nature of LPMP, we had to sample Darmouth Wi-Fi mobility data. We made our experiments on this sampled data. We experimented all parameters of LPMP in all ways. We examined how training affects the accuracy performance of the two predictors. We also applied a clustering method on data and then compared the predictors again. After the experiments, we found that the accuracy of the best Markov predictor is better than that of LPMP in total. However, interestingly, LPMP yields more accurate results than the best Markov predictor does for the users with the low prediction accuracy.

The main contributions of this thesis are listed as follows:

- A domain-independent predictor (Markov) and a domain-specific location predictor (LPMP) are compared on an empirical data in terms of the accuracy for the first time. We could not find such a work in the literature.
- We show that the best Markov predictor outperforms a domain-specific predictor in total. However, interestingly, LPMP yields better results for the users with low accuracy.
- We analyze all parameters of LPMP and how they affect the accuracy.

- In the location predicton part, we propose *all-w(AW)* approach which improves the accuracy of LPMP. In addition, in both *fallback-w(FW)* and AW approaches, we use the *empty pattern* concept which also improves the accuracy of LPMP.
- We show the relation between the accuracy and the trace length for LPMP. It has a relationship similar to the one in Markov predictors.
- We observe that LPMP is more successful for Wi-Fi users than for GSM users.

The thesis is organized as follows. In Chapter 2, we make the definitions of terms and metrics used in the thesis and present the domain-independent predictors which has been evaluated on the data which we use before. In this chapter, lastly, we explain which domain-independent predictor is the best with the reasons. In Chapter 3, we present the related work for this thesis. The works related with the location prediction will be in our focus.

In Chapter 4, we explain LPMP with the algorithms for its each phase. Furthermore, we propose a simple method in the location prediction phase of LPMP.

In Chapter 5, we evaluate LPMP and compare it with the best domain-independent predictor in terms of accuracy. We examine all of the parameters of LPMP in many ways. In addition, we study the effects of training and clustering on accuracy. Also we analyze the effect of the time based probability with different time-slices. Finally, in Chapter 6, we conclude the thesis and discuss about some possible future works.

CHAPTER 2

BACKGROUND

In this chapter, we summarize how O(2) Markov fallback predictor was found as the best domain-independent predictor. To do that, we define what we mean by location. We explain how empirical data set we used in our evaluation was collected and also how the domain-independent predictors from four major families work. We give the definitions of the metrics used in comparing the domain-independent predictors. In the last section, the evaluation which has been done during the process of determining the best domain-independent predictor is summarized. Before moving on to the sections, we should point out that this chapter is an extensive summary of the work of Song et al [30]. We adopt the notation and terminology from that work.

2.1 Location

In the context of the thesis, a *location* is an access point(AP) where the user's device is registered. All possible locations are listed in a finite alphabet A. Any location is represented as a symbol a drawn from that alphabet. The sequence of locations a user visited is called as his *location history* which is a string of symbols. If the history has n locations, $L_n = a_1 a_2 ... a_n$ for $1 \leq i \leq n$ where $a_i \in A$. The data we used is a sequence of location *changes* with $a_i \neq a_{i+1}$. Those location changes are also called as *move*s.

2.2 Data Collection

Dartmouth's campus-wide Wi-Fi mobility data [18] is a processed *movement data*. The raw data contained *syslog* messages which the access points transmitted when the client cards associated, re-associated, or disassociated. Each syslog message contained just the unique MAC address of the client card. As a result, only MAC addresses were known and saved. There might be some cases that a single card might be used by the multiple devices or the multiple people. Song et al. [30] ignore this fact intentionally and the term *user* refers to a wireless card, and vice versa, in the comparison of the domain-independent predictors. In our work, we will also make our evaluation based on this assumption.

Dartmouth's raw data was recorded from April 2001 through March 2003. The works [17, 15] explain the details of this process. Darmouth's movement data extracted from the raw data contains a series of locations with time information for each user's trace like in Table 2.1. As seen in table, it also introduces a special location *OFF* to represent the user's departure from the network somehow.

Table 2.1: A sample user trace

Time (in Unix timestamp)	*Location(AP)*
1008253217	AcadBldg12AP2
1008253716	AcadBldg25AP4
1022867758	AcadBldg20AP1
1022868237	OFF

Statistically, the median length, which is the number of the location in the sequence, of the traces is 494 and its maximum is 188,479. In order to analyze data in terms of the length, the traces were also grouped as *short* with 100 or fewer moves, *medium* with 101-1,000 moves and *long* with over 1,000 moves.

2.3 Domain-Independent Predictors

There are two types of the location predictors: domain-independent and domain-specific. While partitioning the location history, whereas the domain-independent predictors rely on just it , the domain-specific ones use domain information such as time, coordinates and ge-

ometry of the location [30, 10]. During the rest of this section, the four major families of the domain-independent predictors, namely, Order-k Markov, LZ-Based, PPM and SPM are explained. Those predictors are also *online predictors*, "which examine history so far, extract the current context, and predict the next location". After each location, the predictor updates it internal tables and predicts the next location with the help of this new state [30]. Our domain-specific predictor we will mention in Chapter 4 is an online predictor, too. In the evaluation of the predictor, we will also care and keep this property.

2.3.1 Markov Family

The predictors of the order-k (or "$O(k)$") Markov family predict the next location with the help of the current *context* which is the sequence of the k most recent symbols in the location history L if possible. Using the current context c of length k, the probability for the next location to be a is calculated with the equation below:

$$P_k(a) = \frac{N(ca, L)}{N(c, L)} \tag{2.1}$$

In Equation 2.1, $N(s', s)$ denotes the number of times the substring s' occurs in the string s. If c has never occurred before the current context, the equation evaluates to $0/1 = 0$ for all a, and $O(k)$ Markov predictor cannot predict any location. Otherwise, it predicts the location with the highest probability; that is, the location that most frequently followed c in the history. Note that $O(0)$ Markov predictor returns the location most frequently seen in L since the context is empty and $k = 0$.

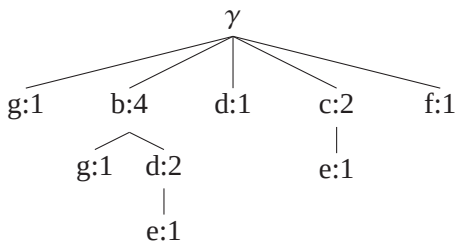
Example: We use $O(2)$ Markov predictor and the history is $L = abcdabdababcab$. The current context (last 2 locations of the history) ab has seen 5 times(including itself) in the history. The probabilites would be $1/5, 2/5, 1/5$ for a, c, d, respectively. Therefore, it predicts the location c with the highest probability.

2.3.2 LZ Family

Song et al. [30] state that LZ-based predictors, which are often used for text compression, seem promising since most good text compressors are good predictors [33] and LZ-based

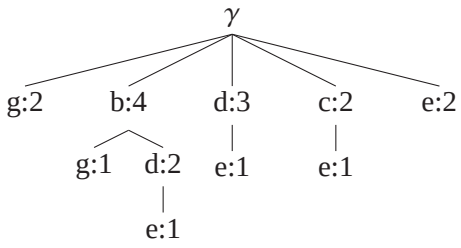
predictors are like the $O(k)$ Markov predictor except that k can grow to infinity [6].

LZ parsing algorithm partitions the input string s into distinct substrings $s_0, s_1, \ldots, s_m$ such that $s_0 = \gamma$ (empty string) and, for all $j > 0$, substring s_j without its last character is equal to some s_i, $0 \leq i < j$ and $s_0 s_1 \ldots s_m = s$. For example, $s = gbdcbgcedbdbde$ is parsed as $\gamma, g, b, d, c, bg, ce, bd, bde$. *LZ tree* whose each node represents one subtring s_i is built as in Figure 2.1.



$s = gbdcbgcefdbdbde$
$s_i = \gamma, g, b, d, c, bg, ce, bd, bde$

Figure 2.1: Example LZ parsing tree



$s = gbdcbgcedbdbde$
$s_i = \gamma, g, b, d, c, bg, e, ce, bd, de, bde$

Figure 2.2: Example LZP parsing tree

Song et al. [30] explain the parsing mechanism for LZ tree like that: ”If any child of the current node (initially, the root) matches the first symbol of s, remove that symbol from s and step down to that child, incrementing its counter; continue from that node, examining the next symbol from s. If the symbol did not match any child of the current node, then remove that symbol from s and add a new child to the current node, labeled with that symbol and *counter* = 1; resume parsing at the root with the now shorter string s”.

Song et al. [30] state that while several predictors based on the LZ parsing algorithm have been proposed in the past [1, 19, 13], they evaluate three of those predictors.

- **Basic LZ Predictors:** The probability for the next location to be a is calculated with the equation below:

$$P_k(a) = \frac{N^{LZ}(s_m a, L)}{N^{LZ}(s_m, L)} \tag{2.2}$$

In Equation 2.2, $N(s', s)$ denotes the number of times the substring s' occurs as a prefix among the substrings $s_0, \ldots, s_m$ of L. If there is no such substring starting with s_m except itself, LZ cannot make any prediction. Otherwise, it predicts the location a with the highest probability.

In an online implementation of LZ predictors, after parsing the current location through the LZ tree, the algorithm stops at a node in the tree. If exists, it predicts the location in the child with the highest counter, which means the highest frequency of past occurrence.

- **LZP (LZ + Prefix):** Since not every substring in L forms a node s_i, the cross boundaries of s_i are missed. In such cases, although enough information exists to make a prediction, an LZ predictor cannot make any prediction. To overcome this problem, Bhattacharya and Das [6] proposed the following modification. "When a new leaf is created for s_i, all the proper suffixes of s_i are also inserted in the tree. If a node representing a suffix does not exist, it is created, and the occurrence counter for every prefix of every suffix is incremented" [30]. Figure 2.2 shows the constructed LZP tree for $s = gbdcbgcefdbdbde$.

- **LeZi (LZ + Prefix + PPM):** In this version, the set S_m of the proper suffixes is constructed for a leaf string s_m and the LZP tree is updated. Then, for each such suffix, all the paths originating from the subtree rooted at the suffix are found. Finally, the PPM algorithm (see Subsection 2.3.3) is applied to those paths and finds the most probable location(s) based on each path's weight (number of occurrences) using the their predicted probabilities.

2.3.3 PPM

Like LZ-based predictors, Prediction by Partial Matching (PPM) is also used for text compression [11]. $Order - k$ (or $O(k)$) PPM uses all contexts (identical to *context* term in Markov

predictors) whose length is between 0 and k. It builds a fixed-order Markov model for each context. Those models are combined using the *escape* probabilities, which are the probabilities of encountering previously unseen symbols. Song et al. [30] use *Method C* [5] in their implementation. According to this method, the escape probability for *k*-symbol context is

$$E_k = \frac{N_e}{N} \tag{2.3}$$

where N_e is the number of escape events which is equal to the number of different symbols that have been seen in the context so far and N is the total events which is equal to the number of all symbols that have been seen in the context so far.

Then, for all $a \in A$, the $O(k)$ PPM probability is

$$P(x_{n+1} = a|L) = P_k(a) + \sum_{i=1}^{k} P_{k-1}(a)E_k \tag{2.4}$$

where P_k is the probability computed using the $O(k)$ Markov model (see Equation 2.1).

2.3.4 SPM

Sampled Pattern Matching (SPM) algorithm is a predictor proposed by Jacquet et al. [16]. SPM is similar to $O(k)$ Markov. However, this time, the context length k is not fixed. Instead, it is determined by a fixed fraction (α) of the longest context that has been previously seen.

SPM finds the longest suffix of L that has occurred previously. This suffix is called as maximal suffix (d). Only the fractional suffix (c) with the length-$\lceil \alpha \cdot |d| \rceil$ of the maximal suffix is used for prediction purpose. The next predicted character a is

$$argmax_{a \in A} N(ca, L) \tag{2.5}$$

where $N(s, L)$ is the number of times the string s occurred in the history L.

Example: The history is $L = abcdabdababcab$. As a result, the maximal suffix is ab. If we take $alfa = 0.5$, the fractional suffix becomes b. Since a, c, d follows b once, twice and once respectively. Therefore, SPM predicts the location c.

2.4 Metrics

Before the definitions of the metrics, we should explain how Song et al. [30] break ties when two or more locations have the same probability. They implemented three tie-break methods:

- **First added:** The first location added to the data structure is predicted; that is, first one seen in the history.
- **Most recently added:** The location that was most recently added to the data structure is predicted.
- **Most recent:** The location that was most recently seen is predicted.

Since the results showed that most of the users had less than about 10% of all predictions and the effects of the the choice of a tie-breaking method on the results were negligible, Song et al. [30] preferred to use *first added* method in their experiments. In Chapter 5, we will mention how we break the ties for LPMP, too.

2.4.1 Accuracy Metric

The predictors may return three possible values for the next location:

- correct location
- incorrect location
- no prediction

In our evaluation, we count *no prediction* case as an incorrect prediction as Song et al. [30] do. *Accuracy* is calculated with the equation:

$$accuracy = \frac{N_c}{N_a} \tag{2.6}$$

where N_c is the number of the correct predictions and N_a is the number of all moves. For $O(1)$ Markov predictor, an example of the accuracy calculation is depicted in Table 2.2. We

will use the righmost bold value in the table as *overall accuracy*. While saying that a user has some percentage accuracy, we will mean the overall accuracy.

Table 2.2: Example accuracy calculation for $O(1)$ Markov predictor

History	a	b	a	b	c	a	b
Prediction	NP	NP	NP	b	a	NP	b
Accuracy	0/1	0/2	0/3	1/4	1/5	1/6	**2/7**

2.4.2 Median Running Accuracy Metric

Song et al. [30] define the *median running accuracy (MRA)* as "at each step *i*, for each trace that has length at least *i*, we compute the average accuracy at each step by dividing the number of correct predictions so far by *i*. There are generally several traces that have length at least *i*; for each step *i*, we find the median accuracy among all such traces and call it the median running accuracy". This metric shows the relation between accuracy and trace length. For $O(1)$ Markov predictor, an example of the median running accuracy calculation is depicted in Table 2.3.

Table 2.3: Example median running accuracy calculation for $O(1)$ Markov predictor

Trace 1							
History	b	d	b	d			
Prediction	NP	NP	NP	d			
Accuracy	0/1	0/2	0/3	1/4			
Trace 2							
History	c	e	c	d	c		
Prediction	NP	NP	NP	NP	NP		
Accuracy	0/1	0/2	0/3	0/4	0/5		
Trace 3							
History	f	g	f	g	f	g	
Prediction	NP	NP	NP	g	f	g	
Accuracy	0/1	0/2	0/3	1/4	2/5	3/6	
Trace 4							
History	a	b	a	b	c	a	b
Prediction	NP	NP	NP	b	a	NP	b
Accuracy	0/1	0/2	0/3	1/4	1/5	1/6	2/7
Step	1	2	3	4	5	6	7
MRA	0	0	0	1/4	1/5	2/6	2/7

2.5 The Best Domain-Independent Predictor

Song et al. [30] compared the accuracy of $O(0)$ Markov with that of $O(1)$, $O(2)$, $O(3)$ and $O(4)$ Markov predictors. They found that the high-order $O(3)$ and $O(4)$ predictors were worse than $O(2)$. Since the context length increases, the number of samples decreases. This causes *no prediction* cases to increase. They reached this inference by using *conditional accuracy* that ignores the unpredicted moves. In that metric, the accuracy equals to the number of correct predictions divided by the number of predictions (not moves). Using that metric, $O(4)$ outpredicted the other Markov predictors. Due to this inference, they used $fallback$ mechanism which is simply based on the recursive use of the result of $O(k-1)$ predictor (with $k = 0$ as the base of recursion) when $O(k)$ fallback predictor encounters an unknown context. The fallback property improved the accuracy of the predictors. In the experiments, $O(2)$ Markov fallback predictor performed the best.

Assigning weights according to the *recency* of occurrence in the past was also experimented. The most recent seen transition is weighted by 1 whereas the others are weighted by 0. As a result, the original $O(2)$ frequency-weighted Markov predictor with the fallback had the best outcome.

Due to the thought that people also move in temporally regular patterns, *Time-aided Markov* predictor was developed. For that predictor, time of day was quantized in one-minute and one-hour buckets. The predictor's state was a pair: (location, time). However, this predictor could not outperform original $O(2)$ Markov fallback location predictor, too.

$O(2)$ Markov predictor was also compared with the versions of LZ mentioned in Subsection 2.3.2, PPM and SPM predictors. Only $O(2)$ PPM and SPM with $\alpha = 0.5$ had an accuracy negligibly better than $O(2)$ Markov fallback predictor. Therefore, Song et al. [30] determined $O(2)$ Markov with fallback as the best overall domain-independent predictor that was simple to implement, had relatively small table size, and had the best overall accuracy.

CHAPTER 3

RELATED WORK

A novel location predictor based Markov family of predictors has been proposed by Sun and Blough [32]. Although this predictor uses a future location list obtained from different available sources such as Microsoft Outlook, Lotus Notes and Google Calendar, it is a domain-independent location predictor since it does not partition the location history using any semantic interpretation. They used the same data [18] we used in their experiments and they reached that their location predictor can improve the prediction accuracy by 3% and 95% over the history Markov predictors and the improvement mainly depends on the user's mobility behavior and how much future knowledge is available.

In [35], a data mining approach similar to LPMP has been compared with *Mobility Prediction based on Transition Matrix* [28] and *Ignorant Prediction* [6]. Unlike LPMP, that approach calculates the support of a pattern by a path as follows:

$$Support(I, A) = \begin{cases} \frac{1}{1+totDist}, & \text{if the elements of } I \text{ are contained by the path } A \text{ in the same order} \\ 0, & \text{otherwise} \end{cases} \tag{3.1}$$

In Equation 3.1, *totDist* is the optimal(lowest) value for the total of the mismatches between A and I. Let $A =< 1, 2, 3, 4, 5, 4 >$ and $I =< 2, 4 >$. According to LPMP, the pattern I is not supported by the path A since A does not contain the elements of I consecutively. Therefore, the support of I by A becomes 0. However, the approach in [35] yields a partial support for the pattern I.

Table 3.1: An alignment example

A	1	2	3	4	5	4
I	-	2	-	4	-	-
I	-	2	-	-	-	4

As seen in Table 3.1, there are two possible alignments for I. *totDist* equals to 1 and 3 for the first and second alignment, respectively. Since the approach selects the first alignment which is the optimal one, A supports I with $1/(1 + 1) = 0.5$.

In [35], the support parameter was also evaluated in the experiments. As the support threshold increased, the accuracy decreased. This result is also valid for LPMP.

Chan et al. [8] compared five basic prediction algorithms based on individual mobility patterns:

- **Location Criterion:** Using the user's present location and the departure history of that location, it predicts the user's next move. The most frequently visited location is predicted as the user's next location.

- **Direction Criterion:** In addition to the Location Criterion, the user's direction information is used. Only the locations in the user's direction are taken into account for the departure history. The location with the highest departure rate is predicted as the user's next location.

- **Segment Criterion:** It extends the Direction Criterion further. All previous movements are partitioned into a number of segments and then stored. A segment starts and ends with a stationary cell in which the user stays for a sufficiently long time. The algorithm tries to match the segment currently under construction with the stored segments. If the present segment is matched with the initial portion of a stored segment, the location immediately after that initial portion is predicted as the user's next location.

- **Bayes' Rule:** It also extends the Direction Criterion so that all departure histories along the future direction of travel are considered. For a location which is two hops away from the present location, the Bayes' Rule [34] formula can be expressed as:

$$P(A_{i-1}A_iB_x|C_m) = \frac{P(A_{i-1}A_iB_x) \times P(C_m|A_{i-1}A_iB_x)}{\sum_{j=1}^{n} P(A_{i-1}A_iB_j) \times P(C_m|A_{i-1}A_iB_j)} \tag{3.2}$$

where A_{i-1} and A_i are the previous and present locations, B_x is the x^{th} possible next move, C_m is the most likely step for visit two hops away and n is the total number of possible next moves. After the calculations, the location with the highest probability is predicted as the user's next location.

- **Time Criterion:** It emphasizes the temporal mobility patterns and imposes the time of cell crossing into the Direction Criterion.

The algorithms above are criticized by [9] since they rely solely on the history of individual movement patterns and do not reflect the recent changes in the user behavior. Doss et al. [9] classify the prediction schemes which has been proposed to overcome that drawback broadly into two classes: the schemes employing individual user mobility information and the schemes employing group mobility patterns. They review many prediction schemes such as the Mobility Motion Prediction Algorithm [23], the Regular Path Recognition Method [12], the Shadow Cluster Scheme [22], the Hierarchical Position Prediction Scheme [24] and the Neural-Network Based Prediction Algorithm [27] from different approaches.

Gonzalez et al. [14] studied the real-time trajectory of 100,000 anonymous cell-phone users (selected randomly from more than 6 million users) whose position is tracked for a six-month period. According to Gonzalez et al., human trajectories show a high degree of temporal and spatial regularity, defined as the probability of finding the user in his most visited location during a certain hour [29], in contrast with what Levy flight and random walk models [7] propose. They ranked each location depending on the number of times a user was recorded in its vicinity. For instance, a location whose rank(L) is 3 refers the third-most-visited location for the selected user. Using those ranks, they observed $P(L) \sim 1/L$ where $P(L)$ is probability of finding a user at a location with a given rank L. Moreover, this outcome was independent of the number of locations visited by user. Therefore, this means that people spend most of their time in a few locations. Moving from that point, they reached a conclusion indicating that despite the diversity of their travel history, humans follow simple reproducible patterns regardless of time and distance.

In another work, Song et al. [29] focused on the limits of predictability, which refers to the probability of foreseeing a user's future whereabouts in the next hour based on his previous trajectory, in human mobility. They studied three months of the mobility patterns belonging to 50,000 anonymous cell phone users selected randomly from 10 million users. They measured the entropy of each user's trajectory and found that despite the significant differences in their travel patterns, most people had a 93% predictability regardless of how many kilometers they travel. In addition, they explored that the regularity (same as [14]) and predictability were not affected significantly by demographic factors such as age, gender and language groups.

CHAPTER 4

LOCATION PREDICTOR VIA MOBILITY PROFILER FRAMEWORK

In this chapter, we will explain how we converted Dartmouth's data [18] into the format of Location Predictor Via Mobility Profiler Framework (LPMP). In addition, we will explain how LPMP works [4] and how we implemented it. We note that the work of Bayir et al. [4] is the main source of this chapter.

4.1 Preliminaries

4.1.1 Data Conversion

We showed how each user's trace were recorded in Table 2.1. Information in the table is enough to be used by LPMP but its format is not suitable for LPMP. We converted data into the format as in Table 4.1.

Table 4.1: A sample user trace in the format of LPMP

Start Time	*End Time*	*Cell*
1008253217	1008253716	AcadBldg12AP2
1008253716	1022867758	AcadBldg25AP4
1022867758	1022868237	AcadBldg25AP4
1022868237	1022868237	OFF

The conversion was done as follows. We renamed *Time*, *Location* columns as *Start Time* and *Cell* respectively. We added a new column named *End Time*. We filled the value of that column with *Start Time* value of the next record. For the last record, we just put *Start Time*

value as *End Time* value. In fact, this value does not make any sense for the last record since the last cell is always an end-location as we will see in Subsection 4.2.1. We call each record as the *cell span record*.

4.1.2 Overview of Location Prediction Process

The location prediction process starts with the *path construction phase*. In this phase, the paths representing a user's travel from one end-location to another are constructed. After this phase, Bayir et al. [4] apply also *cell clustering* against the *ping-pong effect* [21] which we will mention in the clustering experiments (see Section 5.5. Although we will not cluster data in our experiments except the clustering ones, after the path construction phase we will also use *cluster* term in place of *cell* term (AP location).

The second phase is the topology construction. In fact, this phase has not any effects on the accuracy of LPMP. It is used just to make the *pattern discovery* phase more efficient. In the pattern discovery phase, the frequent mobility patterns are discovered. Then, in the *location prediction* phase, those patterns are used to predict the user's next location.

4.2 Location Predictor

In this section, we go into the details of the four phases of LPMP.

4.2.1 Path Construction

Before we mention how the phase works, we list the definitions [4] required for the phase:

Definition(Cell Duration Time): Cell duration time is the difference between end and start time for each cell span record L that shows how many seconds the user connected to the specified AP and calculated with:

$$L^k_{dur} = L^k_{end} - L^k_{start} \tag{4.1}$$

where L^k_{dur}, L^k_{end}, L^k_{start} are the cell duration time, the connection end time and the connection

start time for k^{th} cell span record, respectively.

Definition(Cell Transition Time): Cell transition time is the difference between the end time and the start time of two consecutive cell span records and calculated with:

$$L^k_{tra} = L^{k+1}_{start} - L^k_{end} \tag{4.2}$$

where L^k_{dur}, L^k_{end}, L^k_{start} are the k^{th} cell duration time, the connection end time for k^{th} cell span record and the connection start time for $(k+1)^{th}$ cell span record, respectively.

Definition(Observed End-Location): A cell span record whose duration time L^k_{dur} is greater than the predefined threshold $\delta_{duration}$ is called as an observed end-location record:

$$L^k_{dur} > \delta_{duration} \tag{4.3}$$

Definition(Hidden End-Location): A location which is between two consecutive cell span record k^{th} and $(k+1)^{th}$ and the user stayed longer than a predefined upper bound $\delta_{transition}$ in is called as an hidden end-location:

$$L^k_{tra} > \delta_{transition} \tag{4.4}$$

In our evaluation of LPMP, we do not use this type of end-location since the special location OFF (see Section 2.2) is introduced in our data and thus, the transition time is always zero. However, for the sake of the completeness, we mention and use it in this chapter.

Definition(Mobility Path): An ordered sequence of the cells that a user visited during her travel from one end-location to another is called as a mobility path $C = [C_1, C_2, C_3, \ldots, C_n]$. A mobility path must satisfy the following two rules:

- **End Location Rule:** $\forall C_k \in C, L^k_{dur} > \delta_{duration} \Rightarrow k = 1 \; or \; k = |C|$
- **Transition Time Rule:** $\forall C_k, C_{k+1} \in C \Rightarrow L^{k+1}_{start} - L^k_{end} \leq \delta_{transition}$

1: $\delta_{dur} \leftarrow$ Duration time threshold
2: $\delta_{tra} \leftarrow$ Transition time threshold
3: $L \leftarrow$ Cell Span Record set {Sorted by time; Cell Span Record structure: ($start$, end, $cell$)}
4: $tempPath \leftarrow \emptyset$ {Cell Span Record set}
5: $f_{set} \leftarrow \emptyset$ {Output; final Path set}
6: **for all** Cell span record L_i in L **do**
7: $dur_i \leftarrow end_i - start_i$
8: **if** $dur_i \leq \delta_{dur}$ **then**
9: **if** $tempPath \neq \emptyset$ **then**
10: **if** $start_i - GetEndTime(tempPath_{last}) \leq \delta_{tra}$ **then**
11: $tempPath \leftarrow tempPath \cup [L_i]$
12: **else**
13: $f_{set} \leftarrow f_{set} \cup tempPath$
14: $tempPath \leftarrow [L_i]$
15: **end if**
16: **else**
17: $tempPath \leftarrow [L_i]$
18: **end if**
19: **else**
20: **if** $tempPath \neq \emptyset$ **then**
21: **if** $start_i - GetEndTime(tempPath_{last}) \leq \delta_{tra}$ **then**
22: $tempPath \leftarrow tempPath \cup [L_i]$
23: **end if**
24: $f_{set} \leftarrow f_{set} \cup tempPath$
25: **end if**
26: $tempPath \leftarrow [L_i]$
27: **end if**
28: **end for**
29: **if** $tempPath \neq \emptyset$ **then**
30: $S \leftarrow S \cup tempPath$
31: **end if**

Algorithm 4.1: Mobility Path Construction

In order to show how Algorithm 4.1 works, we run the algorithm on data in Table 4.2 [4] with $\delta_{duration} = 7$ and $\delta_{transition} = 5$. In the table, T_{start}, T_{end}, T_{dur} and T_{tra} are start time, end time, duration time and transition time, respectively.

The algoritm creates an initial path containing only the first cell $[C_1]$. Then, since $T_{dur} > \delta_{duration}$ for $move = 4$, the algoritm terminates the current path $[C_1, C_2, C_3, C_5]$.

Since the end-location $[C_5]$ is an observed end-location, it becomes the initial cell of the new path. This time, since $T_{tra} > \delta_{transition}$ for $move = 7$, the algorithm terminates the current path $[C_5, C_3, C_1]$ not appending the current cell $[C_2]$. The inequality $T_{tra} > \delta_{transition}$ states that the user enters a hidden location after cell C_1. Thus, C_2 cannot be appended to the previous path and a new path $[C_2]$ is initalized. After all records are exhausted, the algorithm stops and returns the mobility paths in Table 4.3.

Table 4.2: An example cell span data set

move	T_{start}	T_{end}	T_{dur}	T_{tra}	*cell*
1	0	4	4	-1	C_1
2	6	9	3	2	C_2
3	9	13	4	0	C_3
4	15	23	8	2	C_5
5	23	27	4	0	C_3
6	27	30	3	0	C_1
7	41	45	4	11	C_2
8	49	50	1	4	C_3
9	56	58	2	6	C_1
10	58	61	3	0	C_3
11	62	66	4	1	C_4

Table 4.3: Reconstructed path set

PathId	*Path*
1	$[C_1, C_2, C_3, C_5]$
2	$[C_5, C_3, C_1]$
3	$[C_2, C_3]$
4	$[C_1, C_3, C_4]$

4.2.2 Topology Construction

This phase does not affect the accuracy of LPMP but may expedite the pattern discovery phase with the exponential time complexity by eliminating majority of candidate path sequences. One scan through the mobility paths is enough to construct the topology graph. During this scan, an edge between the cell cluster pairs C_k and C_{k+1} is created if both of them exist in any path in consecutive positions. Algorithm 4.2 contains a pseudocode explaining this phase.

```
1: S ← Path Set {in terms of clusters}
2: Link ← ∅ {Output; topology matrix}
3: for all Path S_i in S do
4:     for all Cluster C_k and C_{k+1} in S_i do
5:         Link[C_k][C_{k+1}] ← true;
6:     end for
7: end for
```

Algorithm 4.2: Topology Construction

4.2.3 Pattern Discovery

In this phase, a modified version of AprioriAll [2] algorithm, which is called as Sequential Apriori algorithm, is used to discover the frequent mobility patterns from the mobility paths. According to this algorithm, if one's items of the two sequences are found inside the other in a consecutive order, there exists support relation between the two sequences. For example, the sequence $< 1, 2, 3 >$ does not support $< 1, 3 >$ since 3 does not follow 1 consecutively in $< 1, 2, 3 >$. There is a 2 in the middle. However, $< 1, 3, 2 >$ supports $< 1, 3 >$. Therefore, a path S supports a pattern P if and only if P is a substring of S. All the paths supporting a pattern are called as its *support set*.

Since our focus is on the location prediction, we do not need to determine which pattern is maximal. Therefore, we removed the parts related with maximal patterns in the algorithm written by Bayir et al. [4]. The final state of the algorithm is depicted in Algorithm 4.3.

The algorithm works as follows:

```
1: δ_sup ← Minimum support frequency
2: S ← Paths of clusters
3: Link ← Topology matrix
4: C ← Cluster set
5: P ← ∅ {Output; Set of the frequent patterns}
6: L_1 ← ∅ {Set of the frequent length-1 patterns}
7: for i = 1 to |C| do
8:     if Support([C_i], S) ≥ δ_sup then
9:         L_1 ← L_1 ∪ [C_i]
10:    end if
11: end for
12: k ← 1
13: loop
14:    if L_k = ∅ then
15:        Break the loop
16:    else
17:        L_{k+1} ← ∅
18:        for all Pattern I_i in L_k do
19:            for all Cluster C_j in L_1 do
20:                if Link[LastCluster(I_i), C_j] = true then
21:                    T ← I_i • C_j {Append C_j to I_i}
22:                    if Support(T, S) ≥ δ_sup then
23:                        L_{k+1} ← L_{k+1} ∪ [T]
24:                        P ← P ∪ [T]
25:                    end if
26:                end if
27:            end for
28:        end for
29:    end if
30:    k ← k + 1
31: end loop
```

Algorithm 4.3: Sequential Apriori

- In the beginning, the clusters with sufficient support form a set of the frequent (supported) length-1 patterns.

- If the last cluster of a length-k pattern is incident to the cell cluster of the length-1 pattern, length-1 cell cluster is appended to length-k pattern and thus, length-(k + 1) candidate pattern is generated.

- If the support of the length-(k+1) pattern is equal to or greater than the required support, it becomes a supported (frequent) pattern.

- At some value k, if no new supported pattern is generated, the iteration halts.

The function $Support(I : Pattern, S)$ determines whether the pattern I has sufficient support from all the mobility paths S generated in the path construction phase. The support is calculated as:

$$Support(I) = \frac{|S_i| \forall I \text{ is substring of } S_i|}{|S|} \tag{4.5}$$

Being $\delta_{support} = 0.25$, an example execution of Algorithm 4.3 is presented in Table 4.4. In the table, the patterns in bold are the frequent patterns for each iteration. The other ones are eliminated due to their insufficient support. After 4th iteration, since no new frequent patterns can be generated, the iteration halts.

Table 4.4: Patterns generated at each iteration

$Step$	$Pattern : Support$
1	$\mathbf{< C_1 >: 0.75}, < C_2 >: \mathbf{0.50}, < C_3 >: \mathbf{1.00}, < C_4 >: \mathbf{0.25}, < C_5 >: \mathbf{0.50}$
2	$\mathbf{< C_1, C_2 >: 0.25}, \mathbf{< C_1, C_3 >: 0.25}, \mathbf{< C_2, C_3 >: 0.50}, \mathbf{< C_3, C_1 >: 0.25}$, $\mathbf{< C_3, C_4 >: 0.25}, \mathbf{< C_3, C_5 >: 0.25}, \mathbf{< C_5, C_3 >: 0.25}$
3	$\mathbf{< C_1, C_2, C_3 >: 0.25}, \mathbf{< C_1, C_3, C_4 >: 0.25}, \mathbf{< C_2, C_3, C_5 >: 0.25}$, $\mathbf{< C_5, C_3, C_1 >: 0.25}, < C_1, C_3, C_1 >: 0.0, < C_1, C_3, C_5 >: 0.0$, $< C_2, C_3, C_1 >: 0.0, < C_2, C_3, C_4 >: 0.0, < C_3, C_1, C_2 >: 0.0$, $< C_3, C_1, C_3 >: 0.0, < C_3, C_5, C_3 >: 0.0, < C_5, C_3, C_2 >: 0.0$, $< C_5, C_3, C_4 >: 0.0$
4	$\mathbf{< C_1, C_2, C_3, C_5 >: 0.25}, < C_1, C_2, C_3, C_1 >: 0.0, < C_1, C_2, C_3, C_4 >: 0.0$, $< C_2, C_3, C_5, C_3 >: 0.0, < C_5, C_3, C_1, C_2 >: 0.0, < C_5, C_3, C_1, C_3 >: 0.0$
5	$< C_1, C_2, C_3, C_5, C_3 >: 0.0$

4.2.4 Location Prediction

After the frequent mobility patterns are generated, the confidence of the last elements in each frequent pattern is used for the prediction purpose. The confidence of a location is calculated as [3]:

$$Conf(P, x) = \frac{Support(P \bullet x)}{Support(P)} \tag{4.6}$$

where P is a frequent pattern which equals to the mobility history window w and $P \bullet x$ (here the $\bullet$ is the concatenation operator) is another frequent pattern with length $|w| + 1$. The prediction algorithm works as follows. If the pattern which equals to the mobility history w is found, the frequent patterns with length $|w| + 1$ whose prefixes with length $|w|$ match with w. The last elements ($|w| + 1$-th element) of matched patterns are collected in a candidate set including their confidence values. The elements are sorted by the confidence values. The top-m elements are put into the prediction set (where m is the size of the prediction set). During the location prediction, if the user's next location equals to one of the m locations in the prediction set, LPMP is counted as successful. Otherwise, it is counted as unsuccessful. Algorithm 4.4 shows a pseudocode for the location prediction.

Bayir et al. [4] look for the longest pattern matching with w to predict the user's next location in their work. However, they also put an upper limit for $|w|$. Indeed, this approach is the same as the fallback mechanism in the Markov predictors mentioned in Section 2.5. For example, starting $|w| = 5$, if LPMP cannot find the patterns with length-5 and length-6 whose prefixes with length-$|w|$ match with w, it does the same for $|w| = 4$ and so on. This is exactly the same what the Markov fallback mechanism does. There is only one difference between them, that is, no context (length-0 context) case. In order to remove this difference, we propose the *empty* pattern term. We suppose that all the mobility paths support the empty pattern. As a result, the support of the empty pattern always is 1. If $|w| = 1$ and LPMP cannot make a prediction, then the location in the most supported length-1 pattern is predicted as the user's next location. Hence, we call the whole process as *fallback-w (FW)* approach.

Since FW approach may miss some locations with high confidence in the shorter patterns due to the priority of the longer patterns in the prediction process, it sometimes may cause LPMP to make incorrect predictions. In order to cope with such situations, we propose another

approach which we call as *all-w (AW)*. In this approach, we use all the patterns from the shortest length to the longest minus 1 length. For example, let the length of the shortest patterns and the longest patterns be 2 and 5. LPMP tries to match each frequent pattern which has a length from 2 to 4(=5-1) with w of the suitable length. In addition, to make a prediction, LPMP finds the patterns, which has a length from 3 to 5, whose prefixes are w. If those conditions are satisfied, ignoring the pattern length, the location with the high confidence is predicted as the user's next location. If LPMP cannot make a prediction using the patterns, like in FW approach, LPMP uses the empty pattern to predict the user's next location.

```
1:  P ← Set of the frequent patterns
2:  w ← Current mobility history
3:  m ← Size of prediction set
4:  F ← ∅ {Output; Final prediction set}
5:  CandidateSet ← ∅ {Candidate prediction set}
6:  if w ∈ P then
7:     for all Pattern P_i in P do
8:        if P_i starts with w then
9:           if |P_i| = |w| + 1 then
10:             CandidateSet ← CandidateSet ∪ P_i[|w| + 1]
11:          end if
12:       end if
13:    end for
14:    Sort(CandidateSet) {Sort with respect to confidence values}
15:    F ← CandidateSet[1...m] {Select top m elements}
16: end if
```

Algorithm 4.4: Location Prediction

In our evaluation, we compare FW and AW approaches and also take the prediction set size as 1. Furthermore, we evaluate how the *time slice based probability* affects the accuracy of LPMP in Section 5.4. Time slice based probability is calculated as the number of instances of the pattern observed in the specific time slices over all instances. While calculating the score of each location to be used in the prediction process, it is multiplied with the confidence value of the possible locations.

CHAPTER 5

EVALUATION

In this chapter, we will evaluate the accuracy of LPMP comparing to the best Markov predictor. We will try to achieve better results than those of the best Markov predictor in terms of accuracy. In the first section, we will describe the test environment on which we made the evaluation. In the second section, we will examine how the end location thresholds ($\delta_{duration}$ and $\delta_{transition}$), the frequency support threshold ($\delta_{support}$) and the mobility history window (w) parameters affect the accuracy of LPMP. Next, we will use the half of data to train the two predictors. After training, we will reevaluate the accuracy of the predictors. Then, we will examine the time-slice based probability. Lastly, we will apply a simple clustering method to data and determine whether the accuracy of LPMP relative to that of the best Markov predictor improves or not. In the last section, we will make an overall evaluation of the results.

5.1 Test Environment

In the beginning, we decided to use the whole of data [18] stated in Chapter 2. However, our LPMP experiments took long hours -even days- to generate the results due to the exponential nature of Algorithm 4.3. We made some enhancements in its implementation but still could not obtain the endurable running times. For the user X, Figure 5.1 illustrates how the running time for the accuracy calculation by LPMP exhibits a quadratic growth. The curve $4.76 \times 10^{-5} n^2$ fits the running time results where n is the number of the steps. On the other hand, the running time for the accuracy calculation by Markov predictor grows linearly like in Figure 5.2. This time, the curve $5.71 \times 10^{-2} n$ fits the results.

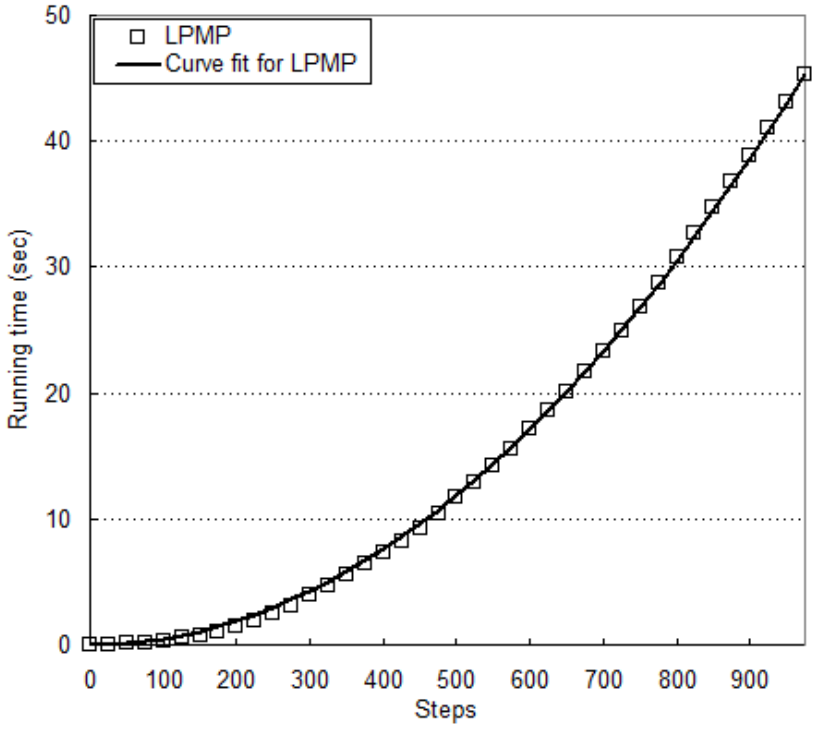


Figure 5.1: Running time for the accuracy calculation by LPMP with $\delta_{duration} = 10min$, $\delta_{support} = 0.001$ and AW method (Curve function = $4.76 \times 10^{-5} n^2$)

To cope with the quadratic growth of the accuracy calculation by LPMP, we had to sample the real data. To do that, we removed all *long* traces which cause LPMP experiments to take long. Next, we sorted *short* and *medium* traces in descending order in terms of the number of the moves. Then, we selected every 10th trace of *medium* traces and every 25th trace of *short* traces. The number of the traces being 6,202 before the sampling has decreased to 289. Therefore, we have sampled the real data at about 5% in terms of the number of the traces. In addition, the number of the moves has decreased from 12,218,093 to 101,195. To determine how successful our sampling has done, we compare the accuracy and the median running accuracy of the real and sampled data in Figure 5.3-5.6.

Figure 5.3 depicts how successfully the accuracy of O(2) Markov fallback predictor for the sampled data overlaps with the one for the real data. To measure the quality of the overlap of the accuracy graphs, we calculate mean (μ) and standard deviation(σ) of the accuracy graphs. We should note that those values in this section do not belong to the accuracy values. They belong to the discrete values forming the accuracy graphs. In Figure 5.3, we obtain $\mu_{sampled} =$

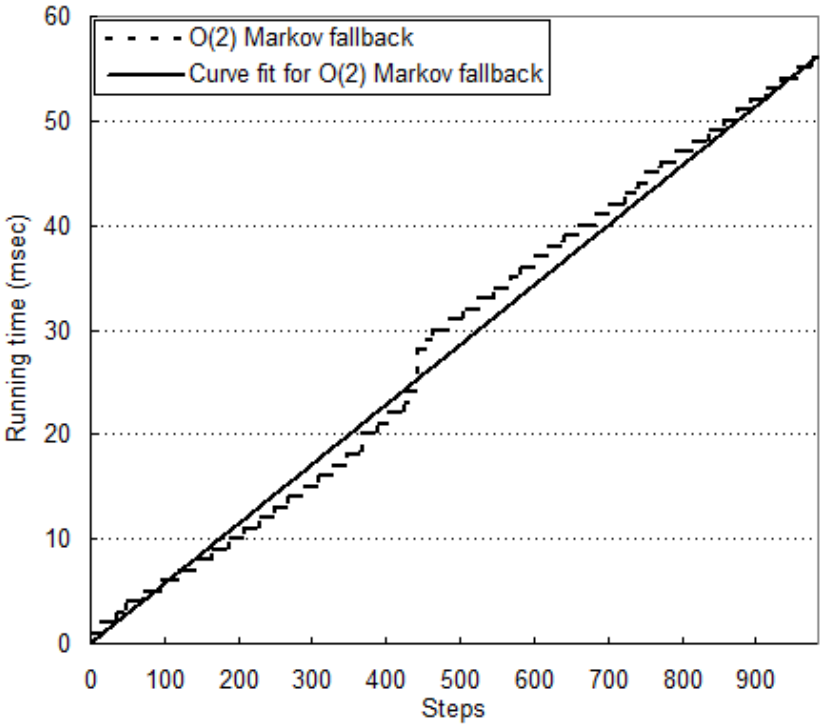


Figure 5.2: Running time for the accuracy calculation by Markov predictor (Curve function $= 5.71 \times 10^{-2} n$)

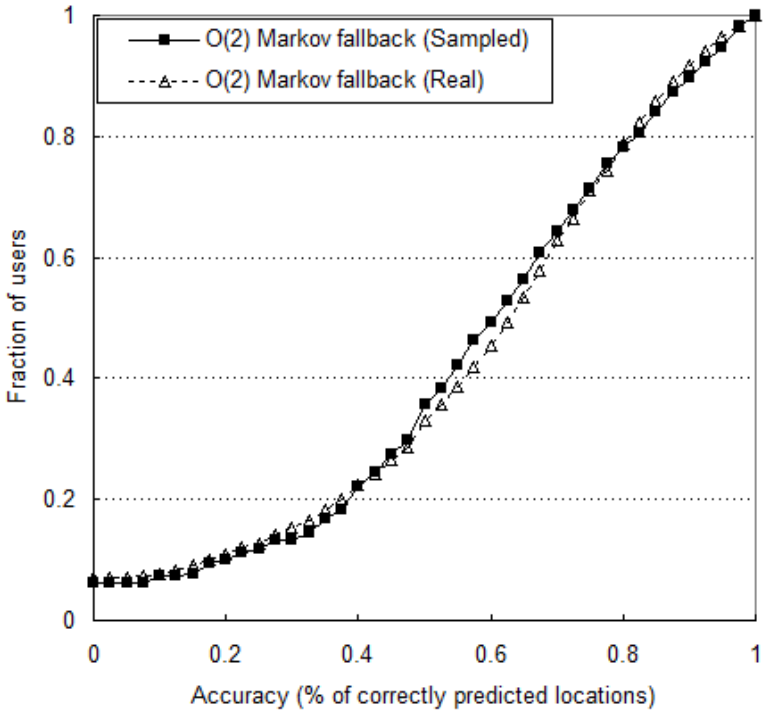


Figure 5.3: O(2) Markov fallback predictors for sampled and real data

0.4232 and $\sigma_{sampled}$ = 0.3241 for the sampled data and μ_{real} = 0.4224 and σ_{real} = 0.3224 for the real data. The error rates for $\mu_{sampled}$ and $\sigma_{sampled}$ are 0.2% and 0.5% respectively.

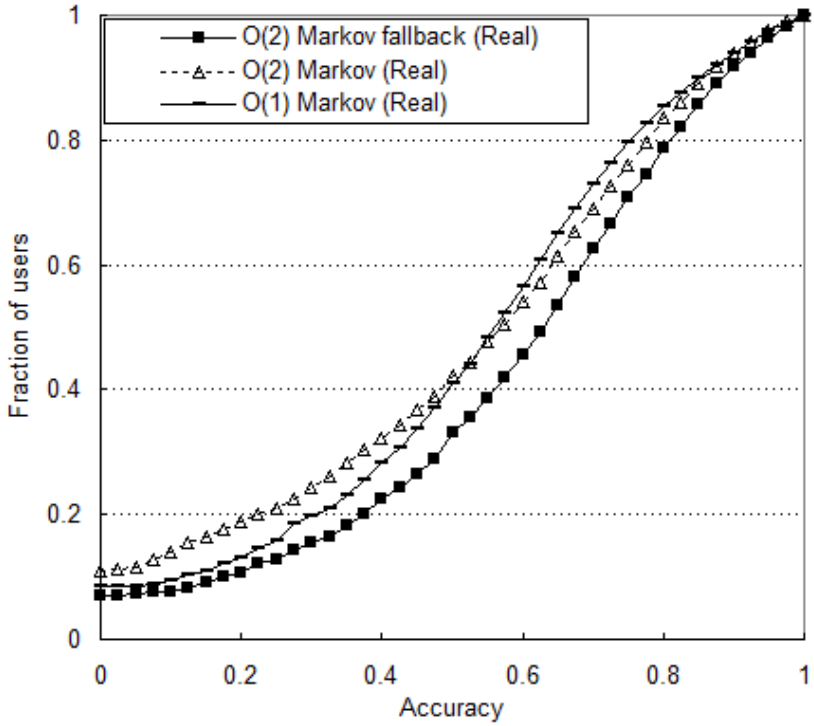


Figure 5.4: Markov predictors for real data

The accuracies of three different Markov predictors relative to each other for the real and the sampled data are illustrated in Figure 5.4 and 5.5. In addition, μ and σ values of the Markov predictors are shown in Table 5.1 and 5.2. As seen in the tables, the error rates are very low and negligible. This means that the accuracy values for the sampled data fit the ones for the real data well and the sampled data can be used in our experiments to make the comparisons between LPMP and the best domain-independent predictor (O(2) Markov fallback).

Table 5.1: mean(μ) values of Markov predictors for sampled and real data

Markov Predictor	$\mu_{sampled}$	μ_{real}	*Error Rate*(%)
O(1)	0.4695	0.4752	1.2
O(2)	0.4922	0.4882	0.8
O(2) fallback	0.4232	0.4224	0.2

Although the accuracy evaluation of the real and the sampled data shows that our sampling

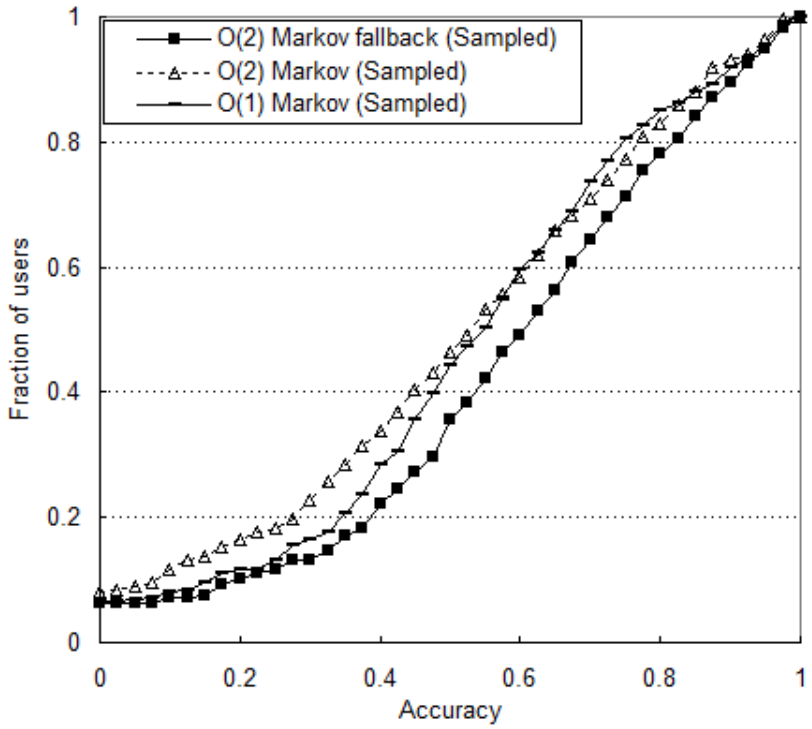


Figure 5.5: Markov predictors for sampled data

Table 5.2: standard deviation(σ) values of Markov predictors for sampled and real data

Markov Predictor	$\sigma_{sampled}$	σ_{real}	*Error Rate*(%)
O(1)	0.3332	0.3283	1.5
O(2)	0.3122	0.3029	3.1
O(2) fallback	0.3241	0.3224	0.5

is sufficiently successful, we have also evaluated MRA of $O(2)$ Markov fallback predictor for the real and the sampled data to ensure the success in our sampling.

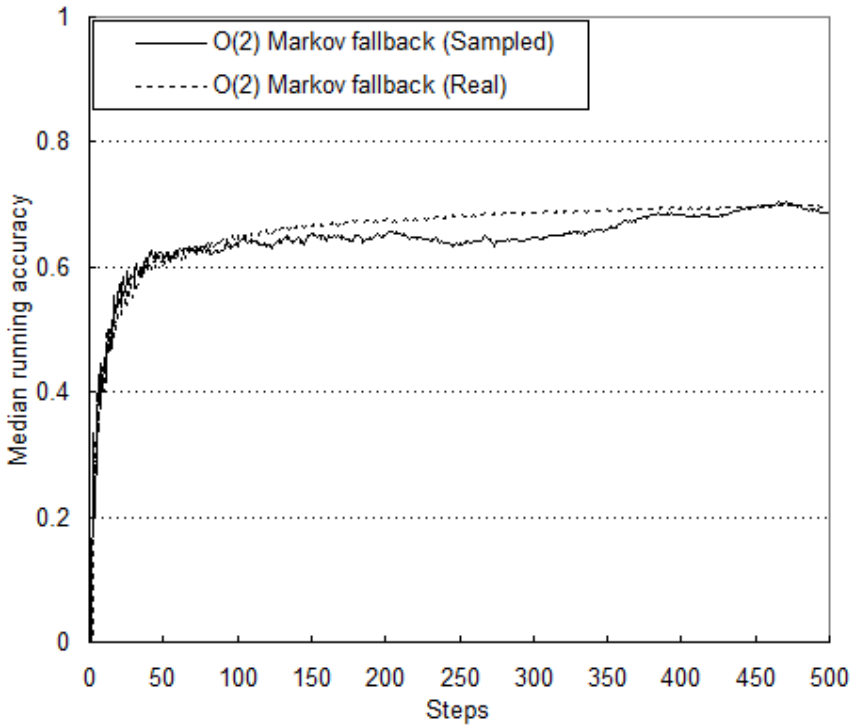


Figure 5.6: Median running accuracy of O(2) Markov fallback predictors for sampled and real data

In Figure 5.6, MRA for only 500 steps is depicted since *median* loses its statistical benefit after 500 steps due to the decrease in the number of the traces with at least 500 steps. Hence, for 500 steps, we obtain $\mu_{sampled} = 0.6428$ and $\sigma_{sampled} = 0.0667$ for the sampled data and $\mu_{real} = 0.6589$ and $\sigma_{real} = 0.0749$ for the real data. The error rates for $\mu_{sampled}$ and $\sigma_{sampled}$ are 2.4% and 10.9% respectively. Since we removed *long* traces completely during the sampling, *median* works in favor of the shorter traces. That is why the error rate of $\sigma_{sampled}$ is relatively high. Therefore, we determine that the error rates for MRA are still acceptable to run our experiments on the sampled data.

5.2 Parameter Experiments

As seen in Algorithm 4.1, the mobility path construction requires L, $\delta_{duration}$ and $\delta_{transition}$ as input. As mentioned in Section 2.2, the special location OFF is introduced in Dartmouth

Wi-Fi mobility data [18] to represent the user's departure from the network. Because of that reason, there are not any time gaps between the consecutive locations. In other words, in our data, the transition time between the consecutive locations is always zero. Therefore, we do not analyze $\delta_{transition}$ and we assume that $\delta_{transition}$ is zero. To be more clear, we should say that $\delta_{transition}$ will not have any effects on our experiments.

In order to start our experiments, we need to determine $\delta_{duration}$ threshold. To do that, we define an experimental duration set which contains 37 5-minute time values from 0 minute to 180 minutes. Because of the layout constraint, just the time values from 0 minute to 60 minutes are shown in Figure 5.7. Like in illustrating the accuracy performance of a predictor, we use cumulative mass function to analyze $\delta_{duration}$. For example, using Figure 5.7, we can say that 40% of the total moves have the duration equal to or less than 15 minutes.

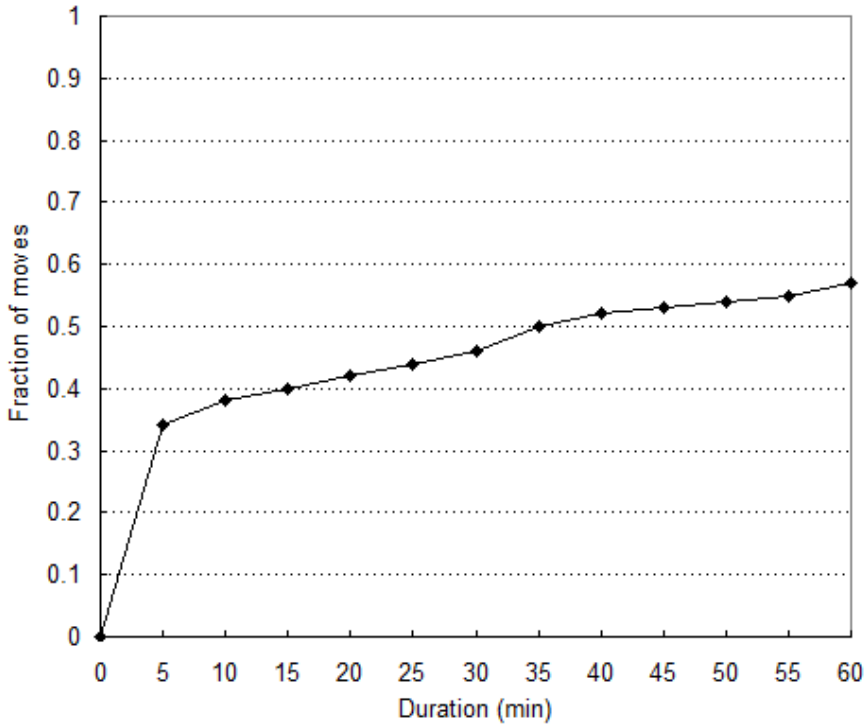


Figure 5.7: Duration threshold analysis

The key point of selecting $\delta_{duration}$ is to differentiate the static (end) locations from the non-static (intermediate) locations. In order to do that, we look for the first sharp decrease between two different time values while going through x axes in $-x$ direction. Until 35th minute, the decrease in the cumulative mass function stays around 1-2%. On the other hand, there exists

4% difference between 35th minute and 30th minute. In other words, 35th minute is the *knee* point of the graph. Therefore, we decided to accept $\delta_{duration}$ as 35 minutes.

The two predictors which we compare are *online* predictors. In fact, the analysis of $\delta_{duration}$ violates this online approach. However, we could also find that value by trial-and-error. In order to start our experiments, we needed an initial value. Consequently, just to accelerate the process of finding a reasonable $\delta_{duration}$ value, we made that analysis. In this point of view, the online approach is not violated. Besides, later in this section, we will also evaluate how reasonable our $\delta_{duration}$ choice is.

The next parameter we have to determine is $\delta_{support}$ in Algorithm 4.3. We do not make any analysis to find an initial value for this parameter unlike $\delta_{duration}$. We start with $\delta_{support} = 0.05$. According to that value, for example, if a user has 100 mobility paths in his/her location history, only the patterns which exist at least 5 mobility paths are taken into consideration in the location prediction process. The other patterns are eliminated before Algorithm 4.4. Later in this section, we will also try the different values for $\delta_{support}$ to improve the accuracy of LPMP.

As mentioned in Subsection 4.2.4, we have the two different approaches for $|w|$: *fallback-w(FW)* and *all-w(AW)*. In our first experiments, we will use FW approach. Then, it will be compared with AW approach. We select $|w| = 2$ for our first experiment.

Before we interpret our first experiment, we need to decide how we will break the ties. If the two or more patterns with the same length have the same confidence, we need to decide how we will choose one of them. At this point, we propose two alternatives: choosing the first generated or the last generated pattern. Here is what we analyze in our first experiment.

As seen in Figure 5.8, the first generated pattern alternative is slightly better than the other one. We can see the statistical proof of that in Table 5.3. Therefore, since $\mu_{first} > \mu_{last}$, we will use the first generated alternative for the next experiments. In the figure, we also show how the usage of the *empty* pattern improves the accuracy of LPMP. Obviously, LPMP which does not use the empty pattern to predict the next location returns the worst result.

Before passing to the next experiment, we should clarify why we do not use *median* accuracy like Song et al. [30] while comparing the predictors. We can use *median* while saying that

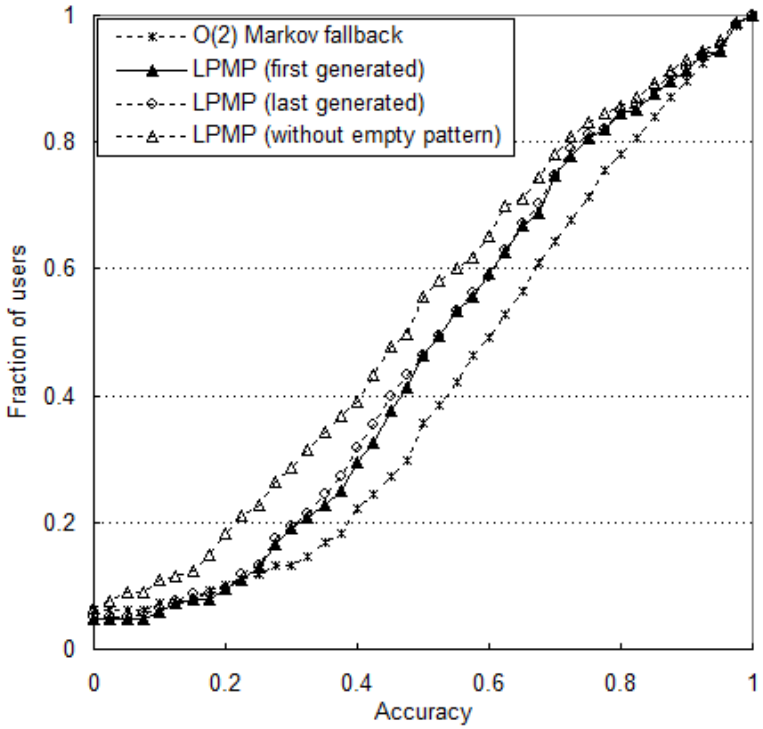


Figure 5.8: First vs. Last generated pattern analysis where $\delta_{duration} = 35min$, $\delta_{support} = 0.05$, $FW = 2$

Table 5.3: mean(μ) accuracy values for Figure 5.8

Predictor	μ
O(2) Markov fallback	0.5797
LPMP (first generated)	0.5315
LPMP (last generated)	0.5302
LPMP (without empty pattern)	0.4732

the predictor has at least *x*% accuracy for the half of the users. This is an indicator of the prediction performance of a predictor. However, use of *median* tool sometimes may take us to incorrect inferences. For instance, as we see in Figure 5.8, the predictor which prefers the first generated patterns has a slightly better graph. We can catch this difference by using *mean* tool over the accuracies. In contrast, if we use median, we miss that issue because *median*s of the predictor which prefers the last generated patterns is greater than that of the other one. Therefore, we use *mean* tool to compare the prediction accuracy of the predictors in our experiments.

Because *O*(2) Markov fallback is the best predictor among the Markov predictors, we started our experiments with $FW = 2$ for LPMP. However, we have to check how acceptable our choice is. The results of our next experiment are depicted in Figure 5.9.

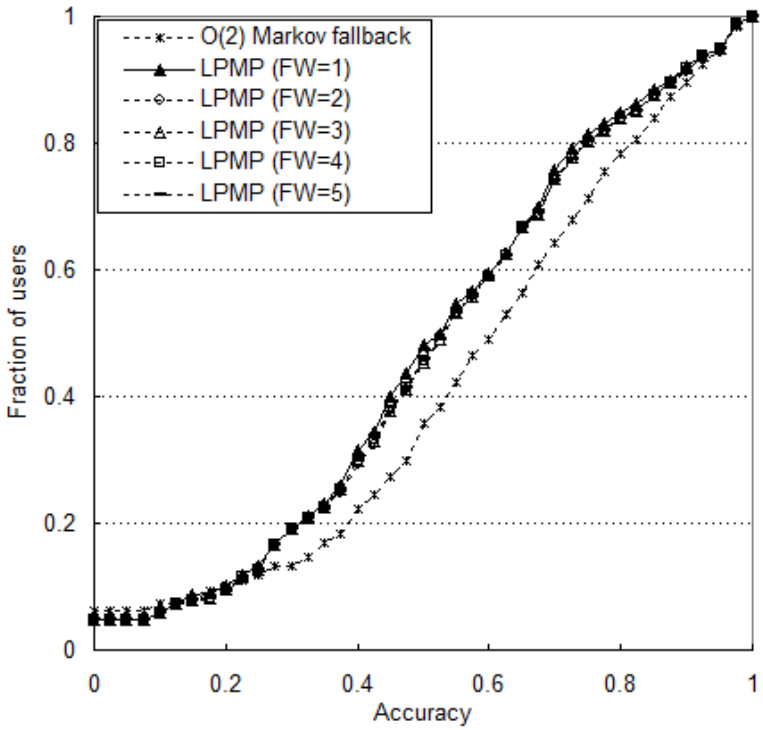


Figure 5.9: FW value analysis where $\delta_{duration} = 35min$, $\delta_{support} = 0.05$

As seen in Figure 5.9, the predictors with different FW values except $FW = 1$ almost overlap each other. We can see this fact numerically in Table 5.4. Indeed, this fact is expected since the number of *"no prediction"* cases increases after some length of *context*. As the length of the context (or pattern) increases, the number of the samples which are used to make a

prediction decreases. As a result, for example, if LPMP cannot make a prediction using the patterns whose length is 5, it tries the patterns whose length is 4 and this goes recursively until LPMP makes a prediction. Therefore, in *fallback* mechanism, after some length of context, the predictors overlap each other. In Figure 5.10, we can see the same behaviour of Markov fallback predictors. As of the contexts whose length is 2, both LPMP and Markov fallback predictors start to converge. However, as seen in Table 5.9, LPMP with FW=4 is the best LPMP predictor because of its highest mean. Therefore, we will use LPMP with FW=4 in the next experiment.

Table 5.4: mean(μ) accuracy values for Figure 5.9

LPMP Predictor	μ
FW=1	0.5245
FW=2	0.5315
FW=3	0.5315
FW=4	0.5316
FW=5	0.5313

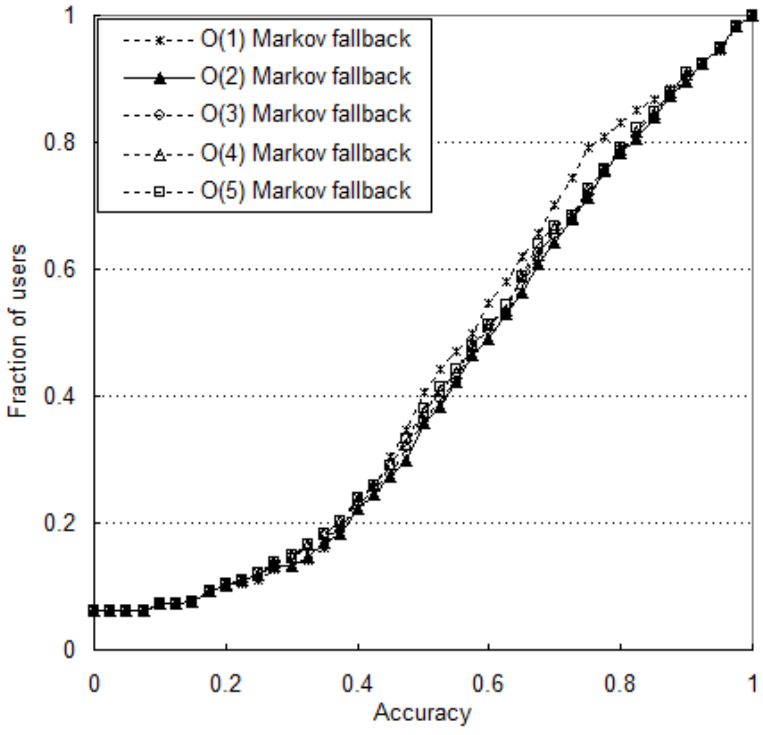


Figure 5.10: Markov fallback predictors

In FW approach, we may miss the correct predictions with high confidence values but short

pattern-length. To overcome this issue, in Subsection 4.2.4, we propose AW approach. However, in AW approach, similar to the comparison of the first and the last generated pattern, also there is an issue which requires that we have to choose an alternative. If two patterns with the different lengths have the same confidence, we have to decide which one is more suitable. Consequently, in our next experiment, we analyze those issues. In Figure 5.11, whereas LPMPs have similar graphs, LPMP with AW-long pattern seems negligibly better than the other two LPMPs. In order to determine which one is more accurate, we put the mean values of the predictors into Table 5.5. According to the table, since LPMP with AW-Long pattern has the highest mean value, we count LPMP with AW-Long pattern as the best LPMP so far. In the next experiments, we will use AW approach and not mention the issue of the long pattern anymore. Thus, wherever we use AW approach, we also mean the use of the long pattern as default.

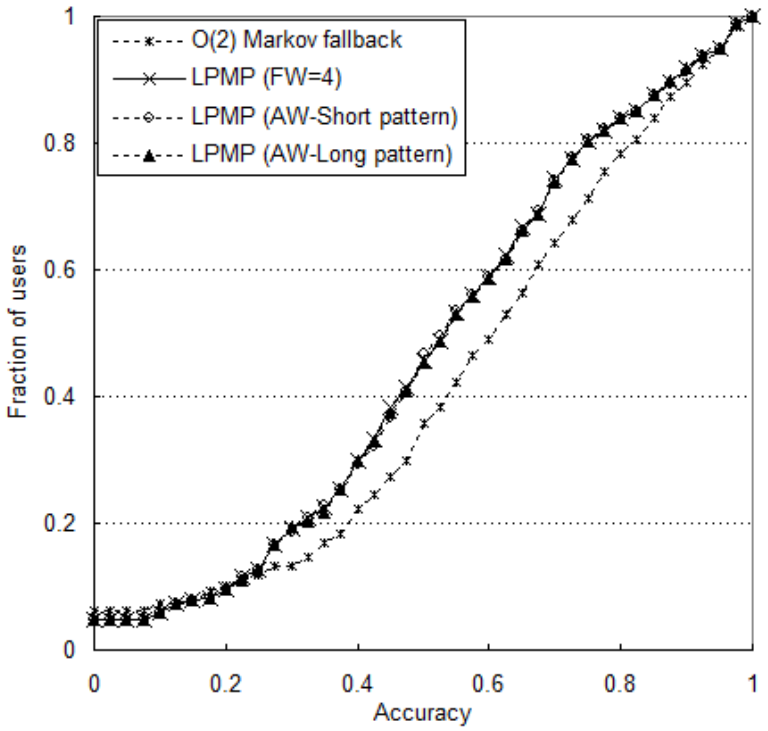


Figure 5.11: FW vs. AW analysis where $\delta_{duration} = 35min$, $\delta_{support} = 0.05$

In our next experiment, we will test the different values for $\delta_{support}$. If the tested data is regular in terms of the mobility patterns and contains very rare (noisy) locations, a higher $\delta_{support}$ which eliminates the noisy patterns may return higher accuracy. However, a higher

Table 5.5: mean(μ) accuracy values for Figure 5.11

LPMP Predictor	μ
FW=4	0.5316
AW-Short pattern	0.5320
AW-Long pattern	0.5326

$\delta_{support}$ may also mean eliminating the beneficial patterns for the correct predictions. For example, let four patterns (*bcd*, *bc*, *abce*, *abc*) be with the supports 0.3, 0.5, 0.07 and 0.1. We suppose that the 3 recent locations are *abc* and naturally the 2 recent locations are *bc*. In this condition, when the predictor calculates the confidence of *d* and *e*, it gets 0.6 and 0.7 respectively and predicts *e* for the next location. However, *bcd* has a higher support than that of *abce*. If we used a higher $\delta_{support}$ such as 0.1, the predictors would return *d* as the next location. To sum up, depending the attributes of data, there may be a trade-off. In our experiment, we finish decreasing $\delta_{support}$ at 0.001 since our sampled data contains at most 1000 moves for any user. Decreasing $\delta_{support}$ after 0.001 does not make any sense. In Figure 5.12, obviously LPMP with $\delta_{support} = 0.001$ is the best among LPMPs. We can see this fact statistically in Table 5.12. In addition, the execution times of the predictors are listed in the table since the support threshold affects the execution time significantly and sometimes we may sacrifice the accuracy for a better execution time.

Table 5.6: mean(μ) accuracy values for Figure 5.12

LPMP Predictor	μ	*Execution time (min)*
$\delta_{support} = 0.1$	0.5179	8.03
$\delta_{support} = 0.05$	0.5326	30.49
$\delta_{support} = 0.01$	0.5598	50.80
$\delta_{support} = 0.001$	0.5666	93.76

Now, we come to the analysis of the parameter $\delta_{duration}$ that makes LPMP a domain-specific predictor. In the beginning of this section, we analyzed it briefly. That analysis tries to differentiate the end locations from the intermediate locations. The determined $\delta_{duration}$ value may work well with the aim of the Mobility Profiler. However, we need to test whether this value is *good* also for the location prediction.

Similar to $\delta_{support}$ analysis, there are two sides of $\delta_{duration}$ analysis. While $\delta_{duration}$ approaches to zero, LPMP starts to lose the dependency on the domain and it becomes a domain-

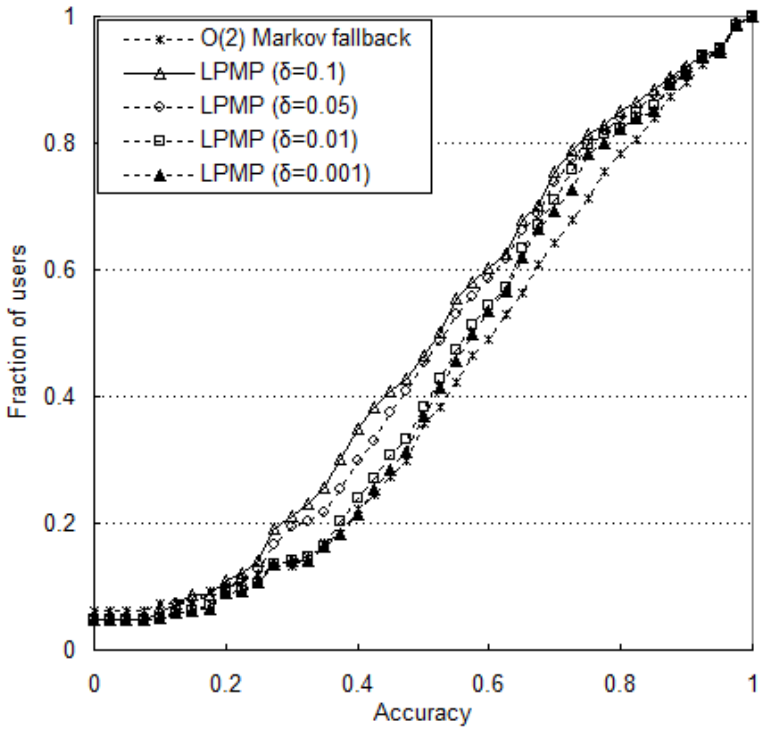


Figure 5.12: $\delta_{support}$ analysis where $\delta_{duration} = 35min$ and the prediction approach is AW

independent predictor. The behaviour of LPMP with $\delta_{duration} \leq 0$ proves this fact. It shows a prediction performance very close to O(1) Markov fallback predictor. This is due to fact that LPMP with $\delta_{duration} \leq 0$ and $O(1)$ Markov fallback predictor partition the location history similarly. LPMP partitions the location history $L = abba$ as the mobility paths *ab*, *bb*, *ba*. Then, it generates the patterns *a*, *b*, *ab*, *bb*, *ba* and predicts the fifth location using the patterns *ab* and *a*. The same is valid for $O(1)$ Markov fallback predictor. It sees the current context as *a* and looks for the location that most frequently followed the current context. In other words, it searches a pattern like *ax* where x is the most frequent location followed *a*. That is why the statistics ($\mu_{LPMP} = 0.5559$, $\sigma_{LPMP} = 0.2458$ and $\mu_{Markov} = 0.5585$, $\sigma_{Markov} = 0.2487$) of those predictors are very close to each other. On the other hand, while $\delta_{duration}$ approaches to infinity, the number of the paths decreases and the path lengths increase. Due to this fact, the pattern supports converge to each other. At infinity, all frequent patterns have the same support that is 1. Therefore, according to our last settings, LPMP always predicts the location with the first-generated longest pattern. For most of the time, this means the incorrect prediction. In conclusion, due to this trade-off, we look for an equilibrium point for $\delta_{duration}$.

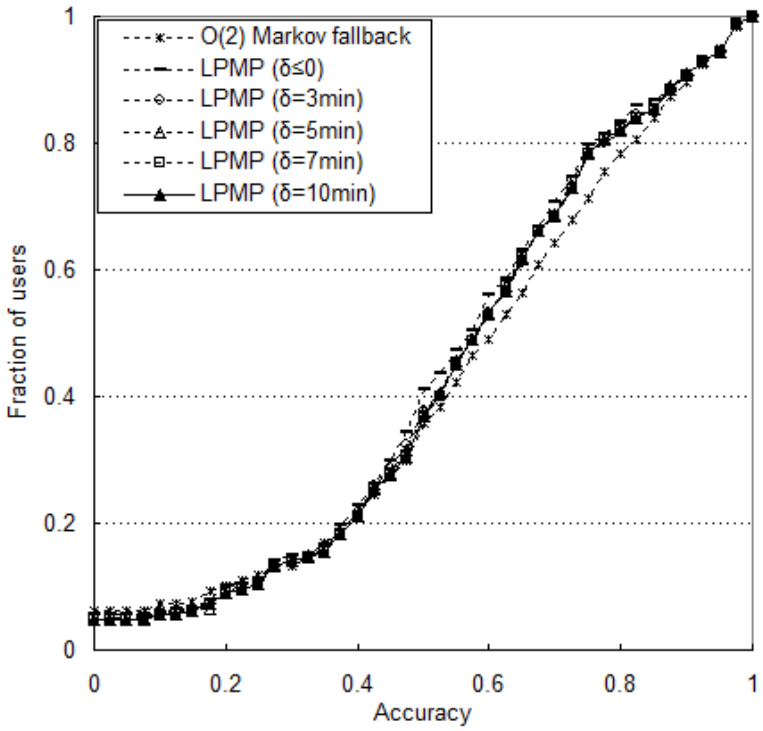


Figure 5.13: $\delta_{duration}$ analysis up to 10min where $\delta_{support} = 0.001$ and the prediction approach is AW

In Figure 5.13 and 5.14, for the different $\delta_{duration}$ values, the prediction performance of LPMPs is depicted. Since the graphs are almost overlapping each other, we cannot notice which one is better easily. Table 5.7 helps us to determine the best LPMP. According to the table, until $\delta_{duration} = 10min$, the accuracy increases and after $10min$, the accuracy decreases. Thus, LPMP with $\delta_{duration} = 10min$ is the best LPMP so far.

Table 5.7: mean(μ) accuracy values for Figure 5.13 and 5.14

LPMP Predictor	μ
$\delta_{duration} \leq 0$	0.5559
$\delta_{duration} = 3$	0.5651
$\delta_{duration} = 5$	0.5670
$\delta_{duration} = 7$	0.5678
$\delta_{duration} = 10$	0.5683
$\delta_{duration} = 20$	0.5682
$\delta_{duration} = 35$	0.5666
$\delta_{duration} = 60$	0.5654
$\delta_{duration} = 90$	0.5644

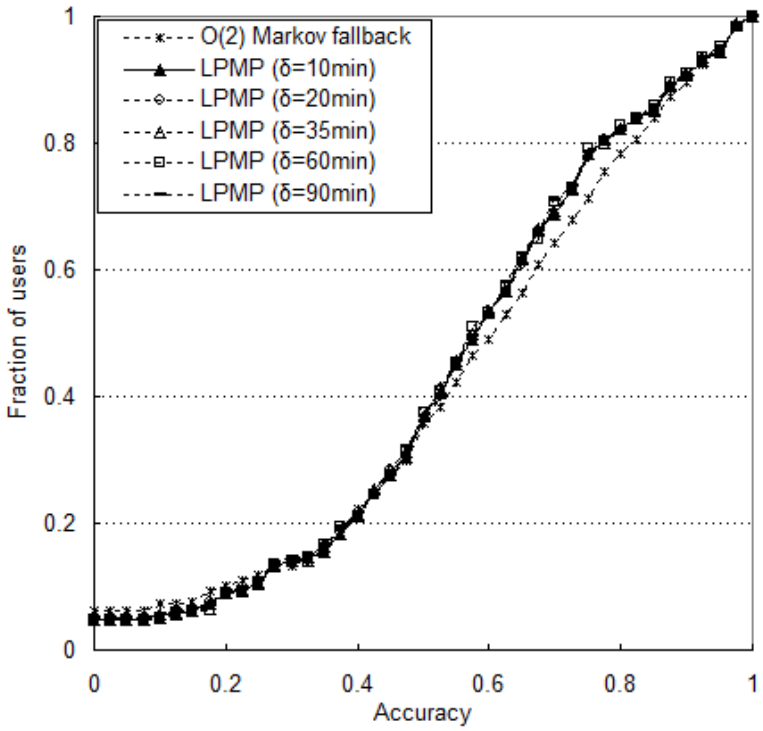


Figure 5.14: $\delta_{duration}$ analysis starting from 10min where $\delta_{support} = 0.001$ and the prediction approach is AW

In this section, we improved the accuracy of LPMP analyzing the parameters of LPMP. As illustrated in Figure 5.15, until *accuracy* = 0.4, LPMP shows better performance. After that point, the best Markov predictor shows better performance. In total, the Markov predictor with $\mu = 0.5797$ is better than LPMP with $\mu = 0.5683$. In the next sections, we will use the parameter values we obtained in this section.

5.3 Training Experiment

In training experiment, we use the half of the user location histories to train the best LPMP and Markov predictors. For example, a user X has such a location history $L = abbacdabdabca$ whose length is 12. The predictors use $12/2 = 6$ locations to train themselves. Then, the accuracy calculation starts. In fact, training exists in all of the experiments. Here, we mean the offline attribute of this training. In Figure 5.16, as expected, the prediction performances of the two predictors are better comparing with the results of online training experiments. In order to explore which predictor is the best learner, we put the statistical values into Table 5.8.

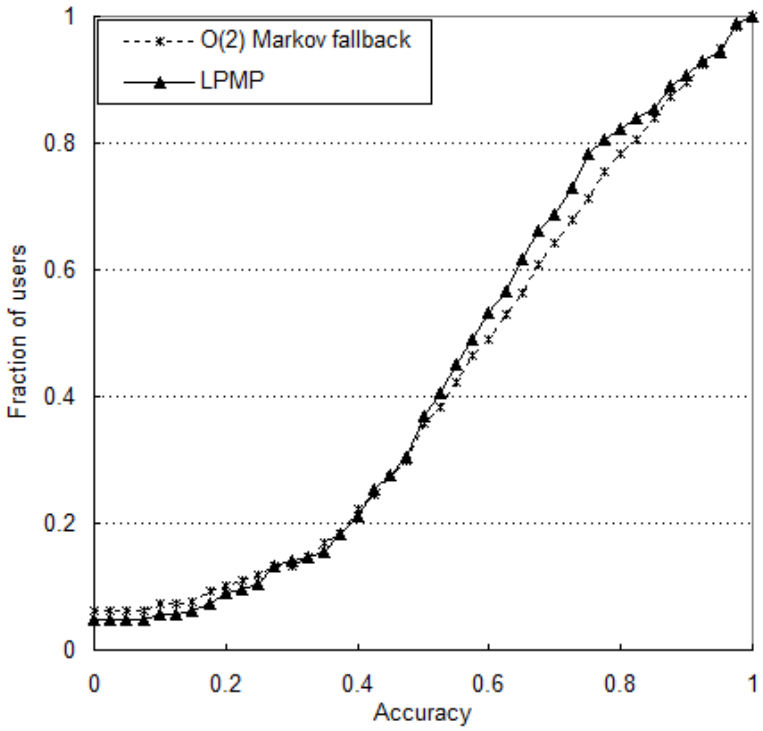


Figure 5.15: Comparison of LPMP ($\delta_{duration}$=10min, $\delta_{support}$ = 0.001 and AW method) and O(2) Markov fallback predictor

Since the error rate after training is higher, we count the Markov predictor as the best learner.

Table 5.8: mean(μ) accuracy values for Figure 5.15 and 5.16

	μ_{Markov}	μ_{LPMP}	*Error Rate(%)*
Before training	0.5797	0.5683	2
After training	0.6019	0.5877	2.4

5.4 Time-Slice Based Probability Experiments

As mentioned in Subsection 4.2.4, the prediction can be done also using the time-slices. We applied the time-slice based probability over the confidence values. Unlike the results of Bayir et al. [4] in Figure 5.17, the time-slice based probability never improved our LPMP predictor as seen in Figure 5.18 and Table 5.9. As the length of the time-slice became shorter, the accuracy decreased. We dropped some graphs from the figure to make the other graphs

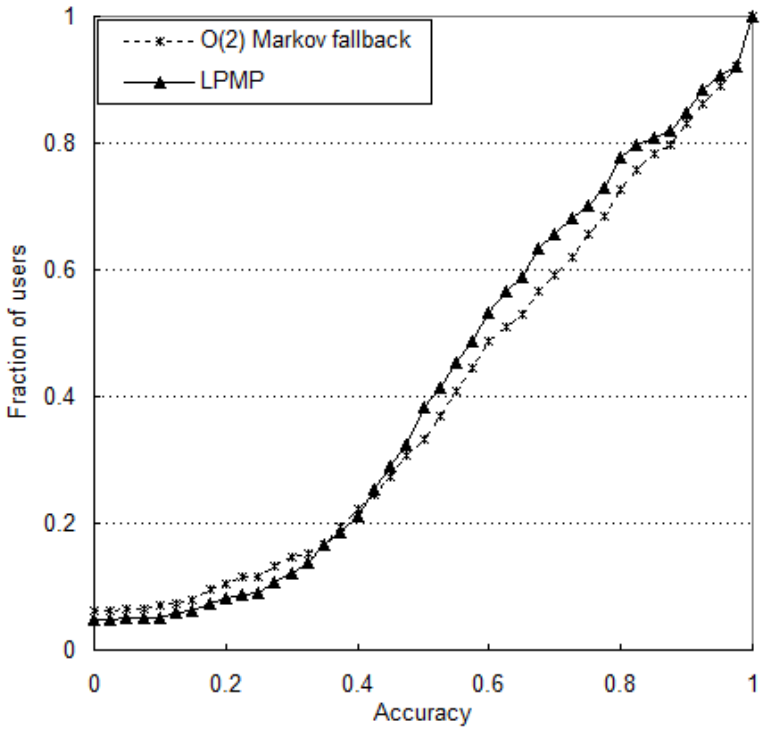


Figure 5.16: Comparison of LPMP ($\delta_{duration}$=10min, $\delta_{support}$ = 0.001 and AW method) and O(2) Markov fallback predictor after training

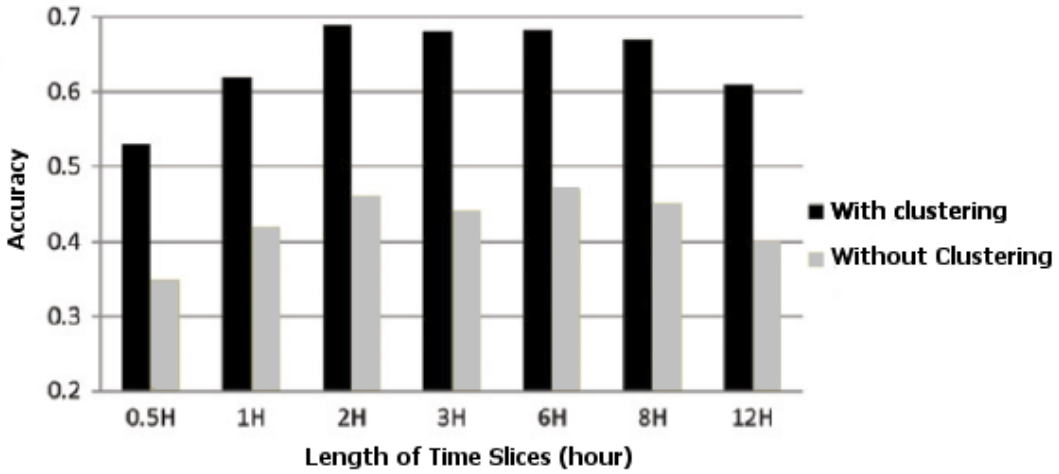


Figure 5.17: Location prediction for user *X* (adopted from [4])

more noticeable. The μ values for the dropped graphs can also be seen in the table.

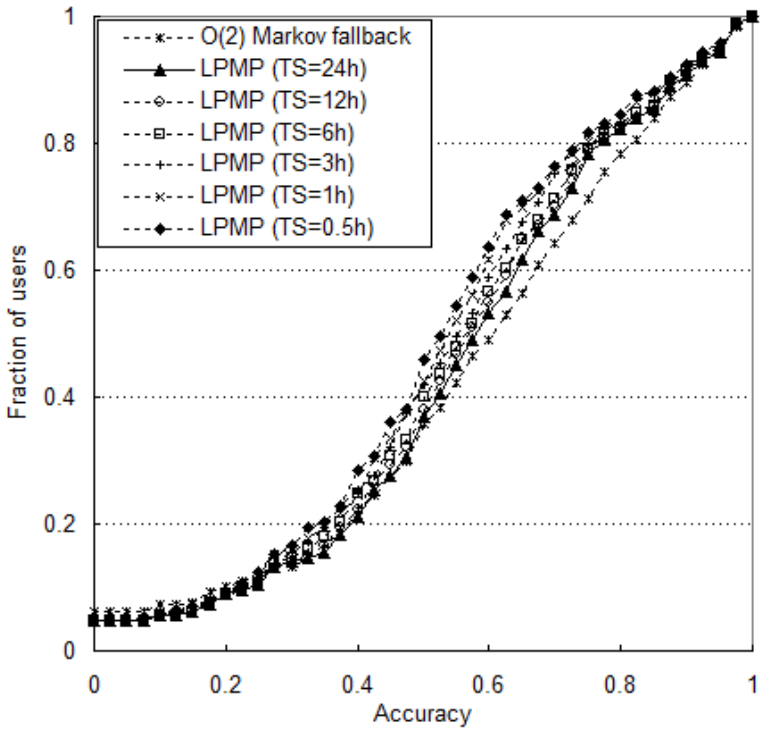


Figure 5.18: Comparison of LPMP ($\delta_{duration} = 10min$, $\delta_{support} = 0.001$ and AW method) with the different time-slices and O(2) Markov fallback predictor

Table 5.9: mean(μ) accuracy values for Figure 5.18

LPMP Predictor	μ
TS=24h	0.5683
TS=12h	0.5618
TS=8h	0.5564
TS=6h	0.5543
TS=4h	0.5501
TS=3h	0.5458
TS=2h	0.5407
TS=1h	0.5361
TS=0.5h	0.5276
TS=0.25h	0.5236

Song et al. [30] proposed a predictor named *time-aided* Markov predictor similar to our time-slice based probability approach. They quantized time of day in one-minute and one-hour buckets. That predictor behaved similarly. As the bucket size decreased, the accuracy decreased. They give two reasons causing the low accuracy of time-aided Markov predictor.

Those reasons are also valid for LPMPs applying time-slice based probability [30]:

- The location information in Darmouth Wi-Fi mobility data [18] is not real human movements and instead, it contains the wireless devices' assocations with access points. Although human movements usually follow patterns according to time, the wireless devices may change their associated APs when they find a stronger signal.
- As the bucket size decreases, the number of samples required to make a prediction decreases. This causes incorrect predictions.

5.5 Clustering Experiments

Although the owner of a wireless device is not mobile, the device may be associated with different APs in a short period of time. These transitions are defined as the ping-pong effect [21]. The ping-pong effect may decrease the prediction performance of the predictors and some clustering on the mobility data may be beneficial to the accuracy of the predictors.

Since Bayir et al. [4] observe that clustering improves prediction performance(30%) (see Figure 5.17, we decide to cluster our data and then compare the two predictors over the clustered data. Instead of clustering method mentioned in the work of Bayir et al. [4], we apply a simpler method [26] to cluster the data. In this method, we divide time into a certain length time-segments and determine the AP which is seen more than the other APs are in each time-segment. That AP is called as the *dominant* AP for the corresponding time-segment. To cluster data, the APs in each time-segment are replaced by the dominant AP.

Table 5.10: mean(μ) accuracy values for Figure 5.15 and 5.20

	μ_{Markov}	μ_{LPMP}	*Error Rate(%)*
Before clustering	0.5797	0.5683	2
After clustering	0.5618	0.5493	2.2

In order to determine the length of the time-segments, again we make a duration analysis. However, this time, we look for the first sharp increase between two different time values while going through x axes in $+x$ direction. In Figure 5.19, obviously $1min$ is the value we

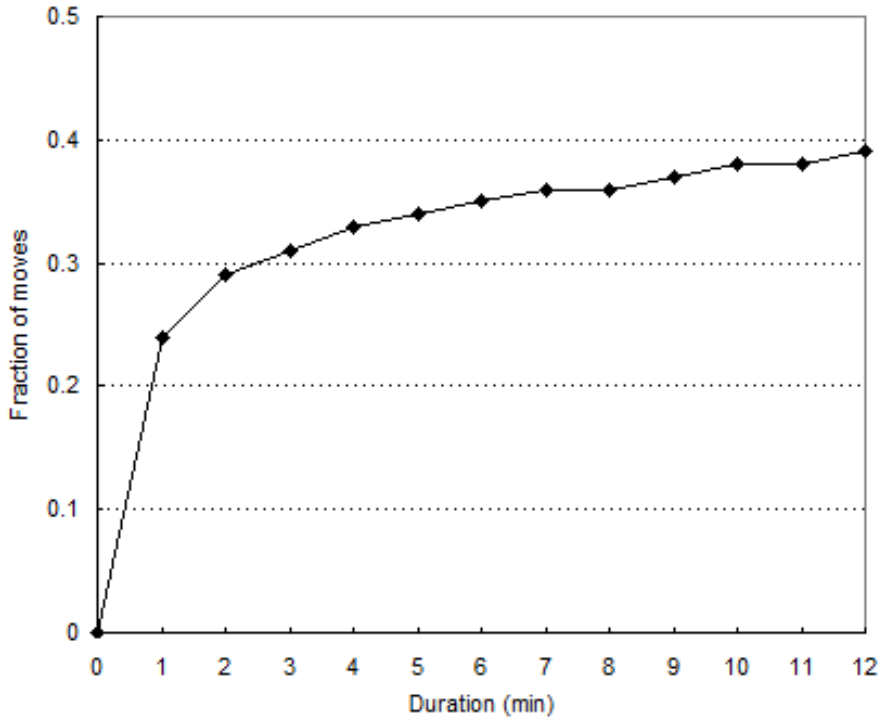


Figure 5.19: Ping-pong effect analysis

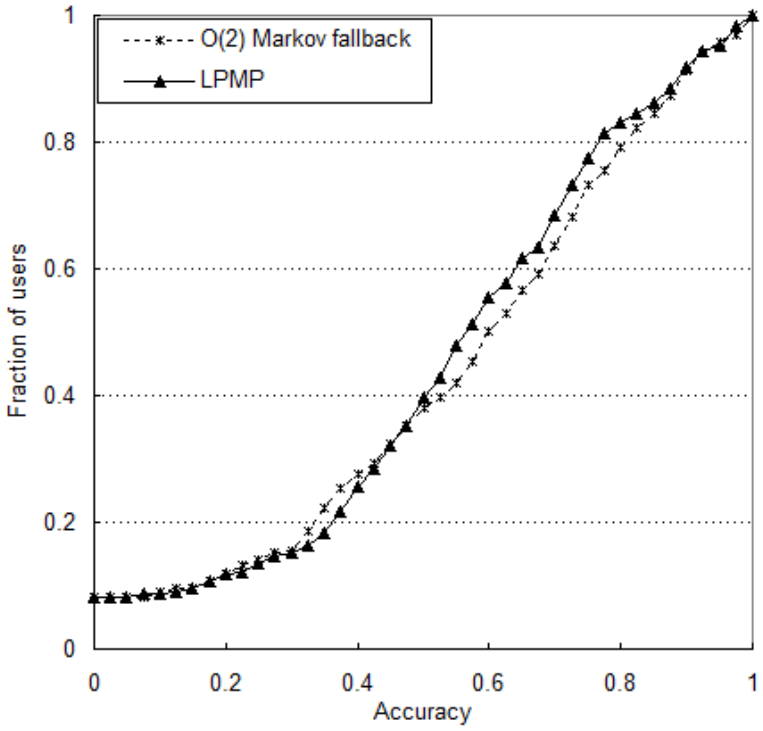


Figure 5.20: Comparison of LPMP ($\delta_{duration} = 10min$, $\delta_{support} = 0.001$ and AW method) and O(2) Markov fallback predictor on the clustered data

look for. This means that there may be some ping-pong transitions if the user stays at an AP less than 1*min*. According to this result, we cluster the mobility data and we run our LPMP on the clustered data. Figure 5.20 depicts the accuracy graphs of the predictors. As seen in Table 5.10, after clustering, the accuracy decreases for the predictors. This is due to fact that the number of the user moves decreases because of the clustering. As the number of the user moves decreases, the accuracy of both LPMP and Markov predictor decreases. Figure 5.21 shows this fact. That's why the accuracy of the predictors decreases. To reveal this fact more clearly, we also run our experiments on the *medium* traces with more than 100 moves for both unclustered and clustered data. Figure 5.22 and 5.23 show the resulting accuracy graphs for the unclustered and the clustered medium traces respectively. When we look at the mean values in Table 5.11, we notice that the mean values before and after the clustering are very close. Probably, as the number of the users with a medium trace increases, the clustered data would give the better results. Another fact we should remark is that the decrease in the error rate after clustering. This decrease may depend on two reasons. First, the clustering decreased the total moves from 101,195 to 73,351. The decrease in the number of the total moves may cause the decrease in the error rate. Second, LPMP may be more sensitive to the ping-pong effect than the Markov predictor. Since the clustering made data more regular, LPMP showed the better performance.

Table 5.11: mean(μ) accuracy values for Figure 5.22 and 5.23

	μ_{Markov}	μ_{LPMP}	*Error Rate(%)*
Before clustering	0.6345	0.6216	2
After clustering	0.6344	0.6218	2

5.6 Overall Evaluation

First of all, we defined the test environment. We explained that we had to sample Dartmouth's campus-wide Wi-Fi mobility data due to the exponential nature of 4.3. We validated our sampled data against the real data.

In Section 5.2, in order to start the experiments, we assigned the initial values to the parameters. After the duration analysis, we decided $\delta_{duration} = 35min$. We specified that there was

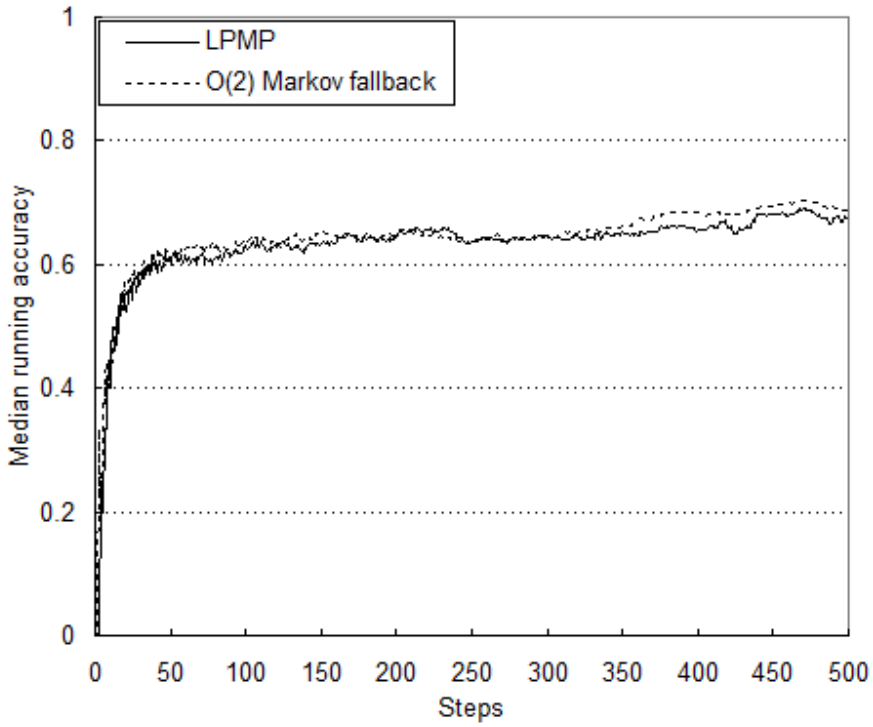


Figure 5.21: Relation between accuracy and trace length ($\delta_{duration} = 10min$, $\delta_{support} = 0.001$, *AW* approach)

no transition time between two consecutive locations due to the special location OFF. As a result, the value of $\delta_{transition}$ was insignificant and $\delta_{transition}$ had no effects on our experiments. Then, we assigned $\delta_{transition}$ as 0.05. For the predictions, we decided to use FW approach with $|w| = 2$ in the first experiment.

In the first experiment, we determined how we would break the ties if two or more patterns with the same length had the same confidence. We reached that the predictor choosing the first generated patterns yielded better result than that of the predictor choosing the last generated ones. Furthermore, we showed the positive effect of the empty pattern on the accuracy of LPMP. In addition, we explained why we measured the accuracy performance of the predictors with *mean* tool. We indicated that *median* might take us to the false inferences.

We tested FW approach with different $|w|$. We found that the predictors converged after $|w| = 1$ and the predictor with $|w| = 4$ produced the best result. Therefore, we updated LPMP with this new value. In addition, we showed the similar behaviour for the Markov predictors.

FW and AW approaches were compared. In AW approach, we needed to decide how we

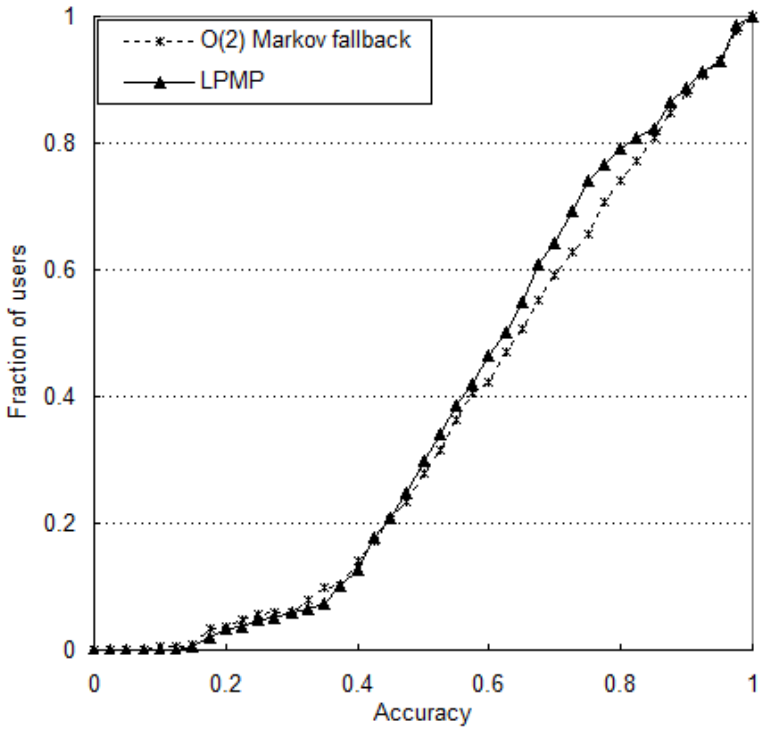


Figure 5.22: Comparison of LPMP ($\delta_{duration} = 10min$, $\delta_{support} = 0.001$ and AW method) and O(2) Markov fallback predictor on the unclustered medium data

would break the ties if two or more patterns with the different lengths had the same confidence. After the experiment, LPMP using AW-Long pattern approach was the best. Thus, we determined to use this approach for the next experiments.

We examined $\delta_{support}$ with the different values. We explained the trade-off in the selection of $\delta_{support}$ value. The experiment gave the best results for $\delta_{support} = 0.001$. Consequently, we updated $\delta_{support}$ parameter of LPMP.

We analyzed $\delta_{duration}$ with the different values less and greater than our initial value which was $35min$. We also mentioned the trade-off issue similar to $\delta_{support}$ case in the selection of $\delta_{duration}$. LPMP with $\delta_{duration} = 10min$ yielded the best results. Hence, we updated $\delta_{duration}$ parameter of LPMP.

In Section 5.2, we broke the ties choosing the first generated patterns for the same length patterns with the same confidence and the long patterns for the different length patterns with the same confidence. Based on these tie-breakers, we found that LPMP worked best with $\delta_{duration} = 10min$, $\delta_{support} = 0.001$ and AW approach. However, this was not enough to

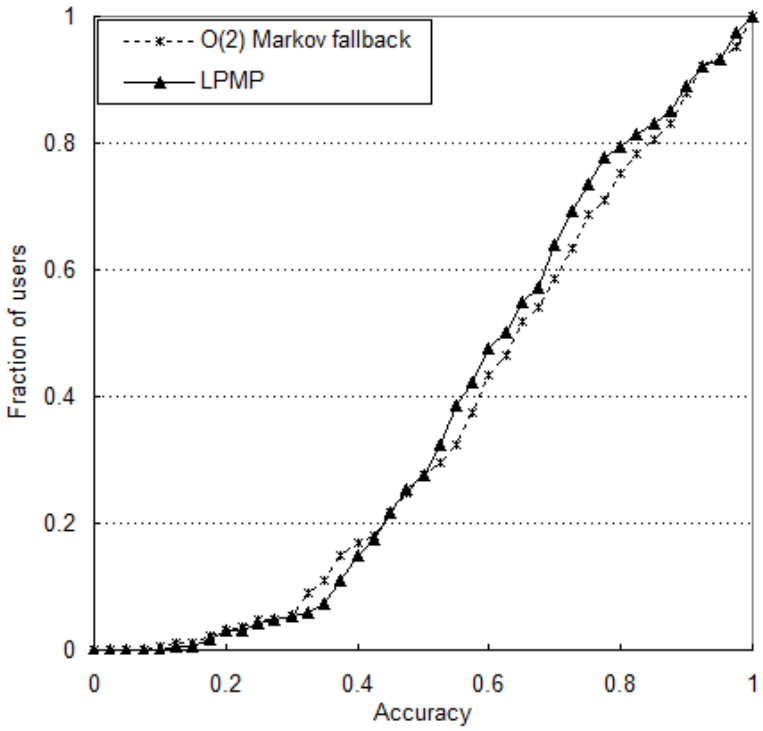


Figure 5.23: Comparison of LPMP ($\delta_{duration} = 10min$, $\delta_{support} = 0.001$ and AW method) and O(2) Markov fallback predictor on the clustered medium data

outperform the Markov predictor.

In Section 5.3, we trained the predictors with the half of data before the accuracy calculation. Naturally, both predictors showed better performance. However, LPMP still did not outperform the Markov predictor.

In Section 5.4, LPMP used the time-slice based probability while making the predictions. However, it showed worse performance. We gave two reasons for that.

In Section 5.5, we clustered our data and run the predictors on that clustered data. For all traces, the predictors yielded worse results. However, this was due to the decrease in the trace lengths after clustering. After we showed the relation between the accuracy and the trace length, we run the predictors on just medium traces. The predictors produced better results. However, LPMP still did not outperform the Markov predictor.

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 Conclusion

In this thesis, we compared two predictors in terms of the accuracy. One of them is $O(2)$ Markov fallback predictor which is a domain-independent predictor and the other one is LPMP which is a domain-specific predictor. In order to make this comparison, we employed a ready-to-use implementation of Markov predictor with some minor modifications. In addition, we implemented LPMP from scratch. To implement it, we removed unnecessary parts of Mobility Profiler Framework [4] and added the *empty* pattern concept and AW approach in the location prediction part.

Due to the quadratic growth of the accuracy calculation by LPMP, we sampled the data at 5% percentage to be able to complete our experiments. After sampling, we discovered that a first generated pattern might return better result when a tie occurred due to some patterns with the same length and the same confidence.

We analyzed FW and AW approaches. We found that FW approach yielded the best result if $|w| = 4$. However, AW approach outperformed LPMPs with different FW values. Furthermore, we discovered that the long patterns with the same confidence, which also some short patterns have, returned better result in AW approach when a tie occurred.

We revealed that $\delta_{support}$ and $\delta_{duration}$ had a trade-off issue. Our experiments confirmed that that issue was valid for $\delta_{duration}$. From $0min$ to some time($10min$), the accuracy of LPMP increased. After that time, the accuracy started to decrease. In contrast, we could not see a trade-off issue for $\delta_{support}$ in our experiments. As the value of $\delta_{support}$ decreased, the accuracy of LPMP increased.

We tested also offline training, time-slice base probability and clustering. As expected, training and clustering improved the accuracy of LPMP. However, unlike in the work of Bayir et al. [4], time-slice based probability worsened the accuracy. The probable reasons for that was presented.

To conclude, although we examined LPMP in many ways and improved its accuracy, in total, LPMP could not outperformed $O(2)$ Markov fallback predictor which is the best domain-independent predictor.

6.2 Future Work

In this thesis, we improved the accuracy of LPMP by changing only one parameter and keep the others fixed. However, there may be some balance among the parameters. For example, we tested $\delta_{duration} = 35min$ and $\delta_{duration} = 10min$ while $\delta_{support} = 0.001$ and found that $\delta_{duration} = 10min$ had higher accuracy. However, for some $\delta_{support}$ values, $\delta_{duration} = 35min$ may yield better result unexpectedly. Therefore, the relations among the parameters need to be studied.

We could not analyze $\delta_{transition}$ since the data does not contain the transition time between the locations. $\delta_{transition}$ can be analyzed by removing OFF location from the data or counting OFF location as transition without modifying the data. The different behavior to OFF and non-OFF locations may bring better accuracy to LPMP.

The prediction set size was taken as 1 in the thesis. For both Markov predictor and LPMP, the different reasonable prediction set sizes can be tested. LPMP may predict the secondary locations more accurately so that LPMP may yield better result than Markov predictor.

We applied a simple clustering method on the data. The other clustering methods may be applied on the data to see how the accuracy will be affected.

Although we cannot see such an indication, LPMP may return better result for *long* traces. In order to run LPMP on long traces, the efficiency of LPMP should be improved. Therefore, as a future work, it may be focused on developing a more efficient version of LPMP.

REFERENCES

[1] Mobility-based predictive call admission control and bandwidth reservation in wireless cellular networks. *Computer Networks*, 38(5):577 – 589, 2002.

[2] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *ICDE '95: Proceedings of the Eleventh International Conference on Data Engineering*, pages 3–14, Washington, DC, USA, 1995. IEEE Computer Society.

[3] Murat Ali Bayir. *Enabling Location Aware Smartphone Applications Via Mobility Profiling*. PhD thesis, State University of New York at Buffalo, May 2010.

[4] Murat Ali Bayir, Murat Demirbas, and Nathan Eagle. Mobility profiler: A framework for discovering mobility profiles of cell phone users. *Pervasive and Mobile Computing*, In Press, Corrected Proof:–, 2010.

[5] Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text compression*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1990.

[6] Amiya Bhattacharya and Sajal K. Das. Lezi-update: an information-theoretic approach to track mobile users in pcs networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 1–12, New York, NY, USA, 1999. ACM.

[7] D. Brockmann, L. Hufnagel, and T. Geisel. The scaling laws of human travel. *Nature*, 439(7075):462–465, January 2006.

[8] J. Chan, S. Zhou, and A. Seneviratne. A qos adaptive mobility prediction scheme for wireless networks. 1998.

[9] R. Chellappa Doss, A. Jennings, and N. Shenoy. A review on current work in mobility prediction for wireless networks. In *Proceedings of 3rd Asian International Mobile Computing Conference*, Kasetsart University, 2004.

[10] Christine Cheng, Ravi Jain, and Eric van den Berg. Location prediction algorithms for mobile wireless systems. pages 245–263, 2003.

[11] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on*, 32(4):396 – 402, apr 1984.

[12] F. Erbas, J. Steuer, D. Eggesieker, K. Kyamakya, and K. Jobinann. A regular path recognition method and prediction of user movements in wireless networks. volume 4, pages 2672 –2676 vol.4, 2001.

[13] M. Feder, N. Merhav, and M. Gutman. Universal prediction of individual sequences. *Information Theory, IEEE Transactions on*, 38(4):1258 –1270, jul. 1992.

[14] Marta C. Gonzalez, Cesar A. Hidalgo, and Albert-Laszlo Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

[15] Tristan Henderson, David Kotz, and Ilya Abyzov. The changing usage of a mature campus-wide wireless network. *Computer Networks*, 52(14):2690 – 2712, 2008.

[16] P. Jacquet, W. Szpankowski, and I. Apostol. A universal predictor based on pattern matching. *Information Theory, IEEE Transactions on*, 48(6):1462 –1472, jun. 2002.

[17] David Kotz and Kobby Essien. Analysis of a campus-wide wireless network. *Wirel. Netw.*, 11(1-2):115–133, 2005.

[18] David Kotz, Tristan Henderson, Ilya Abyzov, and Jihwang Yeo. CRAWDAD trace dartmouth/campus/movement/infocom04 (v. 2004-08-05). Downloaded from http://crawdad.cs.dartmouth.edu/dartmouth/campus/movement/infocom04, August 2004.

[19] P. Krishnan and Jeffrey Scott Vitter. Optimal prediction for prefetching in the worst case. In *SODA '94: Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*, pages 392–401, Philadelphia, PA, USA, 1994. Society for Industrial and Applied Mathematics.

[20] John Krumm. Where will they turn: predicting turn proportions at intersections. *Personal and Ubiquitous Computing*, pages 1–9, 2009. 10.1007/s00779-009-0248-1.

[21] Jong-Kwon Lee and Jennifer C. Hou. Modeling steady-state and transient behaviors of user mobility: formulation, analysis, and application. In *MobiHoc '06: Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing*, pages 85–96, New York, NY, USA, 2006. ACM.

[22] David A. Levine, Ian F. Akyildiz, and Mahmoud Naghshineh. The shadow cluster concept for resource allocation and call admission in atm-based wireless networks. In *MobiCom '95: Proceedings of the 1st annual international conference on Mobile computing and networking*, pages 142–150, New York, NY, USA, 1995. ACM.

[23] George Liu and Gerald Maguire, Jr. A class of mobile motion prediction algorithms for wireless mobile computing and communication. *Mob. Netw. Appl.*, 1(2):113–121, 1996.

[24] Tong Liu, P. Bahl, and I. Chlamtac. Mobility modeling, location tracking, and trajectory prediction in wireless atm networks. *Selected Areas in Communications, IEEE Journal on*, 16(6):922 –936, aug. 1998.

[25] Natalia Marmasse and Chris Schmandt. A user-centered location model. *Personal Ubiquitous Comput.*, 6(5-6):318–321, 2002.

[26] Min Y. Mun, Deborah Estrin, Jeff Burke, and Mark Hansen. Parsimonious mobility classification using gsm and wifi traces. In *Proceedings of the Fifth Workshop on Embedded Networked Sensors (HotEmNets 2008)*, Charlottesville, Virginia, United States, June 2008.

[27] T. Poon and E. Chan. Traffic management in wireless atm network using a hierarchical neural-network based prediction algorithm. In *Proceedings of International Conference on Computers and their Applications*, New Orleans, USA, March 2000.

[28] S. Rajagopal, N. Srinivasan, R.B. Narayan, and X.B.C. Petit. Gps based predictive resource allocation in cellular networks. pages 229 – 234, 2002.

[29] Chaoming Song, Zehui Qu, Nicholas Blumm, and Albert-Laszlo Barabasi. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, February 2010.

[30] L. Song, D. Kotz, R. Jain, and X. He. Evaluating next-cell predictors with extensive wi-fi mobility data. *IEEE Transactions On Mobile Computing*, 5(12):1633–1649, December 2006.

[31] W. Su, S.-J. Lee, and M. Gerla. Mobility prediction in wireless networks. In *MILCOM 2000. 21st Century Military Communications Conference Proceedings*, volume 1, pages 491 –495 vol.1, 2000.

[32] Michael H. Sun and Douglas M. Blough. Mobility prediction using future knowledge. In *MSWiM '07: Proceedings of the 10th ACM Symposium on Modeling, analysis, and simulation of wireless and mobile systems*, pages 235–239, New York, NY, USA, 2007. ACM.

[33] Jeffrey Scott Vitter and P. Krishnan. Optimal prefetching via data compression. *J. ACM*, 43(5):771–793, 1996.

[34] Wayne L. Winston. *Operations research : applications and algorithms*. Belmont : Duxbury Press, 1991.

[35] Gökhan Yavas, Dimitrios Katsaros, Özgür Ulusoy, and Yannis Manolopoulos. A data mining approach for location prediction in mobile environments. *Data & Knowledge Engineering*, 54(2):121 – 146, 2005.

[36] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *Information Theory, IEEE Transactions on*, 24(5):530 – 536, sep 1978.