Handling Relations in a Ubiquitous Computing Environments

By

Min Lin, and Xifeng Wen

Thesis submitted in Partial Fulfillment of the

Requirements for the Degree Master of Technology in

Information and Communication Technology

Faculty of Engineering and Science

University of Agder

Grimstad

May 2009

Preface

This thesis is submitted in partially fulfillment of the requirement for the degree of Master in ICT at Faculty of Engineering and Science at University of Agder, Grimstad, Norway. The Master project documented in this thesis is a part of Ericsson-UiA cooperative research on handling relations between people in the ubiquitous computing environments.

First of all, we would like to thank Dean's Prof. Frank Reichert for introducing us to the fascinating field of mobile services. We benefit a lot from his valuable academic and industrial experience and insightful comments on our project work. Then, we want to address special thanks to Assoc. Prof. Ole Christoffer Granmo for his excellent tutoring. It is him who pulled us out of the painful *"trial-and-error"* period in the research. His encouragements are always the most important driving force for us to accomplish this project successfully. We also wish to thank Research Fellow Anis Yazidi for his thorough discussions with us over the last few months. Some core papers provided by him are very helpful for us to understand the theoretical background of our research problem. He has also exerted efforts to help us to accomplish the prototype development. In addition, Prof. Frank Reichert and Research Fellow Anis Yazidi have also provided some important comments in order to help us to improve the report writing.

Finally, we want to say thanks to Sissel Andreassen, Stein Bergsmark and all other teachers in the ICT Department. Your dedications make us have a beautiful and fruitful time studying in Norway.

Min Lin, and Xifeng Wen
Grimstad
May 2009

Abstract

Context-awareness has been recognized as one of the most important topics in the ubiquitous computing environment. Context-awareness constitutes the basis of self-adaption which has been considered important in many mobile services. Time and location are two fundamental aspects of the context. In our thesis, we investigated a context-awareness problem in which anomalous contexts (times and locations are not conforming to the existing patterns) should be detected in order to serve for a “friend Reminder” mobile service.

We proposed the *Learning Automata* based solutions which are able to work in a pure online manner in this thesis. This is different from most of the previous research work. Our extensive experiments demonstrated the performances of the proposed solutions in different noisy environments. We also brought out new challenges in this regard for the convenience of future researchers.

A prototype of mobile service has been developed for the purpose of demonstration. The prototype can work in an ad hoc network environment. Our peer to peer (P2P) architecture can avoid the dependency on the advanced infrastructural technologies which cannot be found in many real-life scenarios. Since privacy control is of high importance to mobile services, some security schemes have also been integrated into our prototype development.

Table of Contents

List of Figures

1. Introduction

"Ubiquitous computing names the third wave in computing, just now beginning. First were mainframes, each shared by lots of people. Now we are in the personal computing era, person and machine staring uneasily at each other across the desktop. Next comes ubiquitous computing, or the age of calm technology, when technology recedes into the background of our lives."

"The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it."

-- *Mark Weiser (1952-1999), the chief technology officer at Xerox Palo Alto Research Center.*

1.1 Background and Motivation

It has been about 20 years since the term *"Ubiquitous Computing"* was first proposed by the computer scientist Mark Weiser working at Xerox Palo Alto Research Center. As the name implies, "Ubiquitous" means "existing everywhere". The essence of ubiquitous computing lies in its "people-centric" characteristics. That means, computing could be seen as some kind of resources and services, which can be conveniently obtained by people at anytime and anywhere. Under such a circumstance, people will not be restricted to get information processing capacity by sitting in front of their computers, which is a common "machine-centric" case in the desktop computing era.

The people-centric feature makes handling relations between people in the ubiquitous computing environment important. Applications in a ubiquitous computing environment should be able to adapt to its current operational environment autonomously without artificial configuration / interference so that people are enabled to enjoy the seamless and humanistic information services.

Our research problem has been identified as a part of the Ericsson-UiA cooperative research on handling relations between people in the ubiquitous computing environments. A self-adaptive application should be developed for a Wi-Fi based ubiquitous computing environment in our research. In view of this, there are two basic aspects needed to be understood before we can describe our specific research problem in section 1.2. In the first place, it is important to understand the ubiquitous computing environment in a generic sense. In particular, for the sake of clarity and coherence, it'd better to view the ubiquitous computing from an evolutionary point of view. This can be shown in the section 1.1.1. Moreover, considering the self-adaptability requires the applications to be aware of the context, the background knowledge of context and context-awareness will be given in section 1.1.2. Then, we presented the problem description in section 1.2. In this chapter, section 1.3 presents a short definition of this thesis. Section 1.4 clarifies the key assumptions and limitations in our research. Section 1.5 presents our

contributions to the knowledge. Section 1.6 will let the readers have an overview of the organization of this thesis.

1.1.1 Evolutionary History of Computing

Ubiquitous computing can be seen as a significant step in the evolutionary history of computing. The driving force behind the evolution is the rapid development of the hardware and infrastructural technologies. For example, the advancement of wireless technologies makes it possible for many mobile devices to join in a distributed computing environment. In view of this, it is necessary to consider new problems introduced by mobility, QoS (Quality of Service) of wireless network and resource constraints in the mobile computing research area [1].

When coming to ubiquitous computing/pervasive computing, *"the research agenda of pervasive computing subsumes that of mobile computing, but goes much f*urther" [1]. Since ubiquitous computing inherits many common characteristics of its predecessors – distributed computing and mobile computing, many techniques developed in the prior stages would be applicable in the ubiquitous computing environment. However, ubiquitous computing does introduce a few unique challenges, which cannot be addressed by the traditional techniques, to the research communities.

A general evolutionary process of computing can be shown in the following figure (Figure 1 [1]):

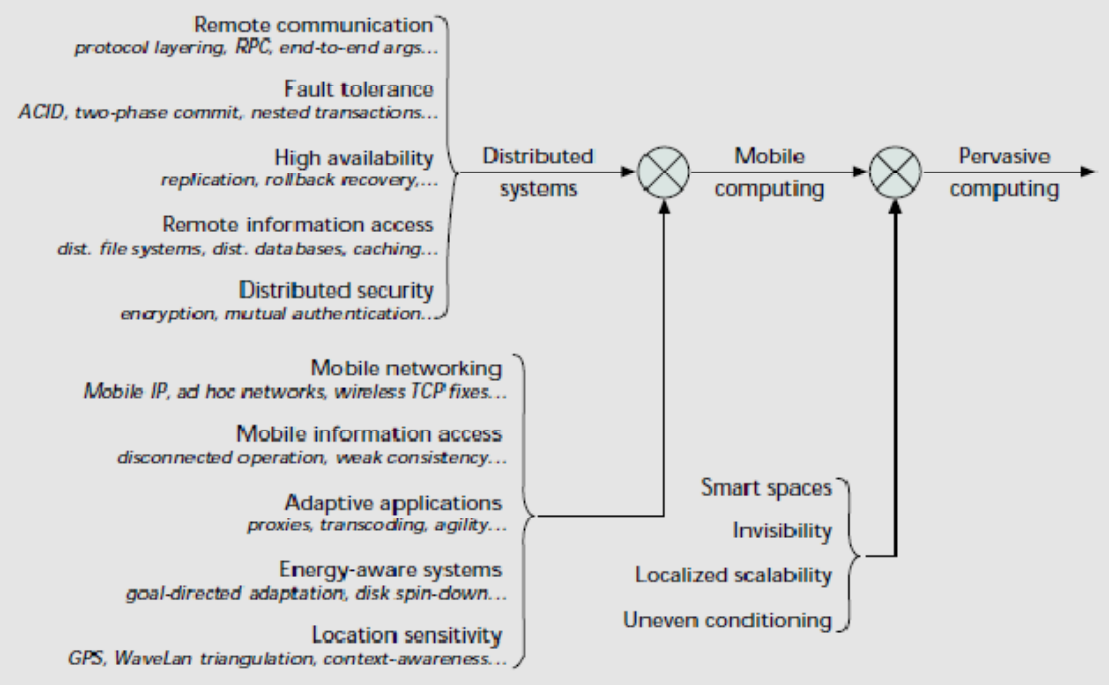


Figure 1 The evolutionary history of computing [1]

.It is important to note that many problems, though they are addressed in the prior stages (logically, not chronologically), will have new complexities introduced when entering into a new phase. And moreover, "*Research effort on some aspect of pervasive computing began relatively early. For example, work on smart spaces began in the early 1990's and proceeded relatively*

independently of work in mobile computing." [1] Context-awareness is one of such research topics, which has been attracting much attention for a relatively long time in the ubiquitous computing community.

1.1.2. Context and Context Awareness

Ubiquitous computing is naturally context aware computing.

It is B. Schilit et al. who first introduced the concepts of "context" and "context-awareness" in 1994 [2]. However, over the past several years, researchers with different research focuses still held different opinions about the definitions of context [3] [6]. G. K. Mostefaoui et al. summarized four different definitions of context in [3]. The authors of [3] also pointed out the formalism are missing from all these definitions. Among the existing definitions, the one proposed by A. k. Dey in [4] seems to be more comprehensive than others:

Definition 1
"Context is any information that can be used to characterize the situation of an entity. An entity is a person, or object that is considered relevant to the interaction between a user and an application, including the user and the application themselves." [4]

From an application point of view, a recent definition of context made by Edwin J. Y. Wei and Alvin T. S. Chan from Hong Kong Polytechnic University place emphasis on the "application-specific" properties of the context [5]:

Definition 2
"Context is application specific, and context of an application is any external information which can be utilized to adapt the data, behavior or structure of this application." [5]

From this definition, it's obvious that the context must be the application-specific information but outside the application. *"Context of one specific application may make no sense to others"* [5], According to Edwin J. Y. Wei and Alvin T. S. Chan.

After the definition of context was introduced, A.K. Dey then proposed a definition of "context-aware" in [4] as the follows:

Definition 3:
"A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task." [4]

For the purpose of better understanding, the following sample scenario (Figure 1) will help to illustrate context aware service in a ubiquitous computing environment.

A student from Grimstad is going to participate in a job interview which will be held in Oslo. When

he gets off the bus in the Oslo bus terminal, his mobile phone will show the weather conditions of the following few days in the Oslo city. The traffic information (e.g. routes, traffic jam) and even the most convenient means of public transportations to the interview office can also be displayed in his handset. These services are generally called context-aware. Here, for the sake of easy understanding, only location information would be counted into the context in this case (Location Based Service). Sometimes, the real-world context usually contains much more diverse information sources. (Figure 2)

From the sample scenario, it can be figured out that the applications in a ubiquitous computing environment should be capable of adapting to their contexts (operational environments). Moreover, all the adaptive processes should be "transparent" to the end users. That means, users in a ubiquitous computing environment will benefit from context-aware services without actually caring about the underlying dynamics in the context.

This leaves to the researchers a series of questions, e.g. 1) how can the context information be acquired from the real-world environment? 2) How can the context information be modeled, stored and managed? 3) How can we make use of the context information in order to provide the adaptive services? Here, only to mention a few of them should be sufficient to illustrate the important and exciting research area we are now working on. The potential outcomes from the research work related to this area would make people get closer to enjoy the "freedom of computing". The next section (Section 1.2) will present the specific research problem, which can be related to the above question 3), in this thesis.

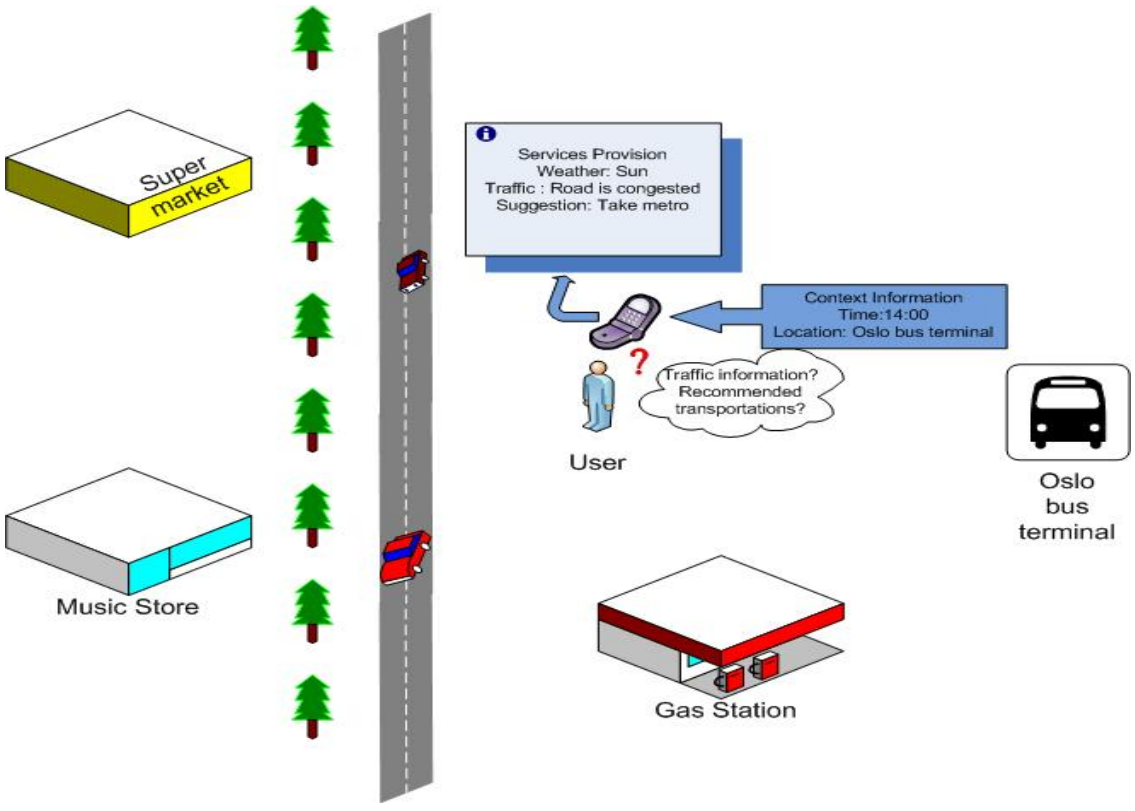


Figure 2 A sample scenario

1.2 Problem Description

Since this project is a part of Ericsson-UiA cooperative Research on handling relations between people in the ubiquitous computing environments, our research problem is being contained in several selected scenarios.

Two roles are involved in these scenarios—Alice and Bob. Alice and Bob are classmates in a study program in the same university. They also live in the same student apartment. They basically meet each other in the classrooms and the student apartment every day. If their mobile phones can be involved in a ubiquitous environment, then, in which context, more precisely, when and where, the alerts should be activated by their mobile phones to remind him/her the presence of the other party? This is a context-awareness problem which can be shown in the following figure (Figure 3).

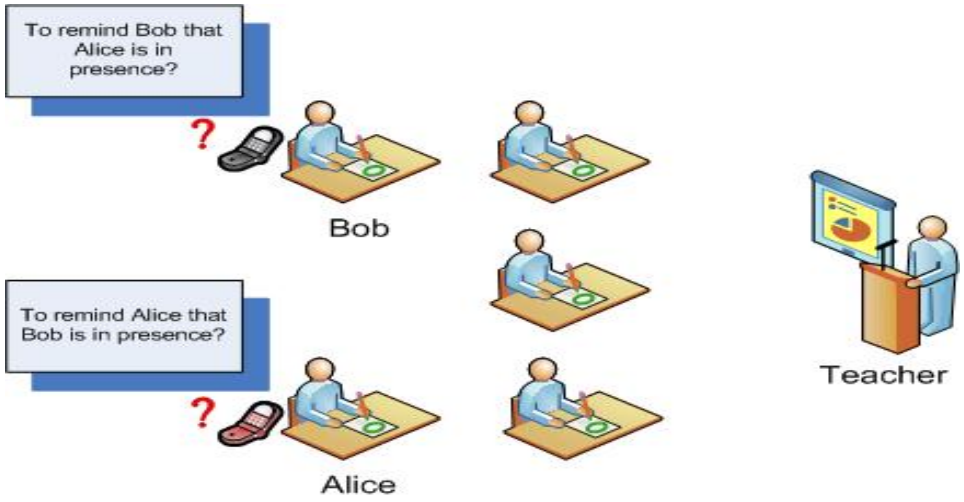


Figure 3 A context-awareness problem

Obviously, if an alert will be activated each time the presence of the other party, both Alice and Bob's daily life will be immersed in a vast sea of numerous "meaningless" alerts. It would be a nuisance since it is apparently unnecessary to remind each other when they meet in the classroom or the apartment every day (*problem scenario 1* in Figure 4).

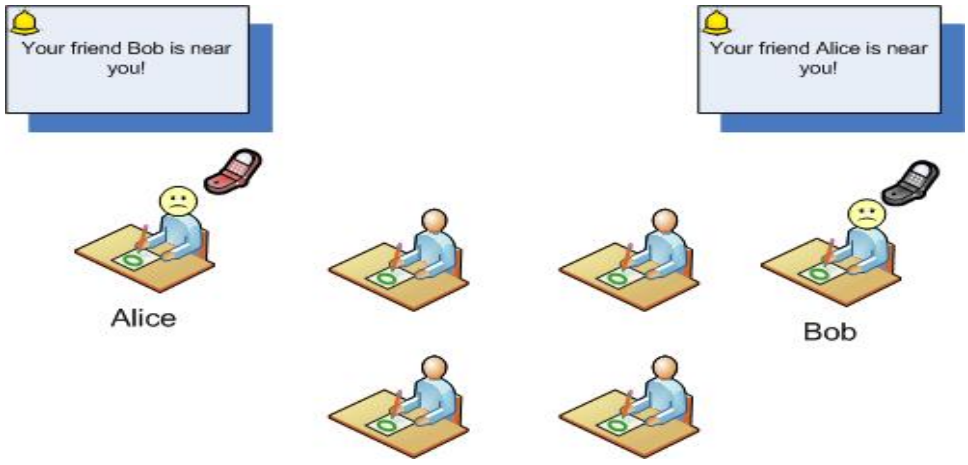


Figure 4 Problem scenario 1

However, assume that one of them, without loss of generality, Alice, went abroad (e.g. attended an academic conference) for probably two weeks. When she comes back to the student apartment or campus, Bob's mobile phone should trigger an alert of Alice's presence since they haven't met for a period. Thus, Bob knows Alice's back without any prior knowledge of her itinerary (*Problem scenario 2* in Figure 5)

Figure 5 Problem Scenario 2

In another scenario, Bob came back from a trip and arrived in the Oslo airport. He was going to do some shopping in the shops at airport. At the same time, Alice was sitting in the concourse and waiting for her flight to Paris. Both of them didn't know anything about each other's travel plan. They even never thought one day they might meet in the same airport. However, in such a situation, the mobile phones held by them should alert them that the other party is located in the same airport right now. Therefore, they may meet and have a little chat (*problem scenario 3* in Figure 6)

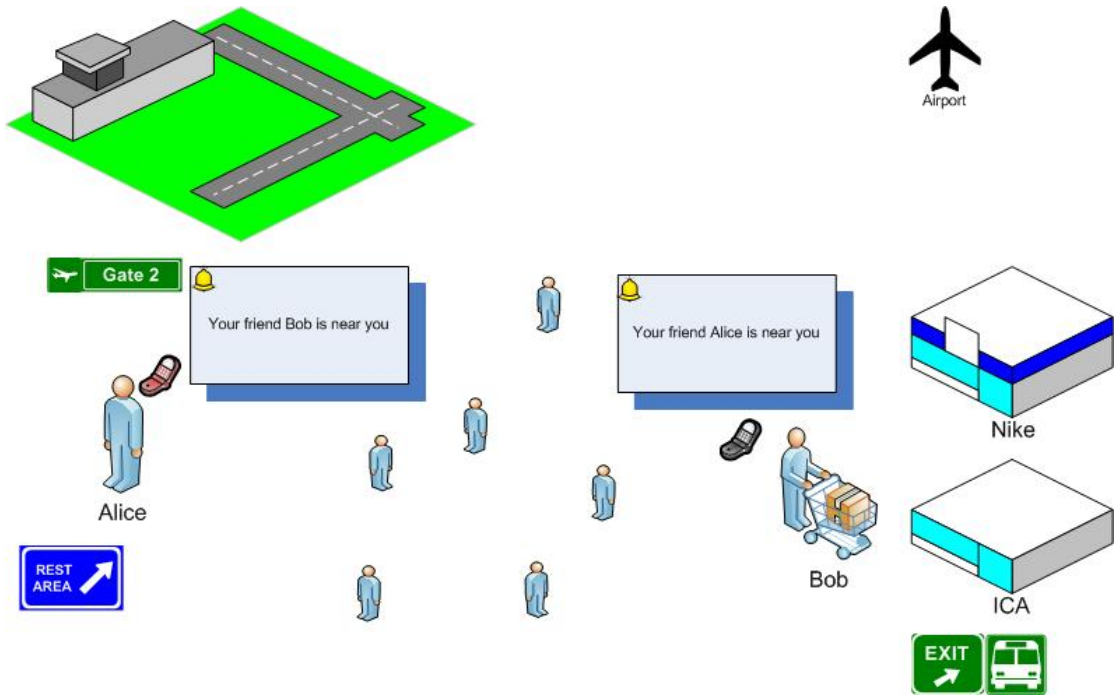


Figure 6 Problem scenario 3

The similar scenarios can be that they suddenly appear in the same hospitals, supermarkets or shopping malls without any prior knowledge of each other's presence. That means, if the mobile phones couldn't trigger an alert in such a case, they probably never have a chance to know the other party is also located in the same place.

Briefly speaking, the project addresses on a basic context-awareness issue in the ubiquitous computing environment. It asks us to find solutions in order to make mobiles phones give the right response (here, alert or not) based on the current situation (context).

In our scenarios, for the sake of simplification, only time and location would be taken in to account when dealing with context information. Actually as we know, the real-world context could be very complex information with many other factors involved. However, the collection of varies information generally needs advanced sensor technologies. The high requirements on infrastructure are usually not that realistic in many real-world places based on the current technical level. In contrast, only the time and location information, which can be considered as the most fundamental aspects of the context, can be easily acquired by applying the existing technologies. That makes our potential solution have a generic sense and wide application.

To avoid any further confusion, it is necessary to clarify that how to acquire context information is not within our problem boundary. However, it doesn't prevent us making use of the existing means to acquire information of times and locations. In addition, a simple data structure will be used to represent the context information in our case since the context modeling is also not our research focus in this thesis. However, Finding solutions to detect the anomalous contexts (times and locations are not conforming to the existing patterns) which could trigger the self-adaption should be the main emphasis of our research.

The prototype is also important to be implemented on mobile phones in order to form a solid demonstration that our solution can be integrated into the practical mobile services.

Certainly, since we are developing applications in the ubiquitous computing environment, the privacy protection should be a crucial matter. We will design the most suitable security schemes for our prototype. In such a case, some existing security techniques or mechanisms would be adapted to the architecture of our prototype.

1.3 Thesis Definition

We define the thesis as the following statement:

The Master project is concerned with a context-awareness issue in the ubiquitous computing environment. Due to the limitations in the present infrastructural technologies, we considered time and location as two fundamental aspects of the context in our research. We proposed solutions to detect anomalous contexts (times and locations are not conforming to the existing patterns) in terms of co-presence in order to provide self-adaptive mobile service. Based on the literature review, we found most of the existing machine learning approaches requires that the

datasets should be prepared beforehand. This prerequisite is not feasible for mobile services in the ubiquitous computing environment. In view of this, also to pursue a better user experience in the mobile services, we are going to propose learning automata based solutions, which can work in a pure online mode with low computational complexity, to such a context-awareness problem. Since we are dealing with context-awareness application development in a Wi-Fi based ubiquitous computing environment, a prototype should be developed in order to demonstrate our overall solution for the related mobile services, not just the core algorithms. For the prototype part, we proposed our implementation which will realize presence-sharing, localization and context-awareness in the form of ad-hoc based social network. In addition, since privacy protection is critical in the ubiquitous computing environment, the design of security schemes will be also a part of our prototype development.

1.4 Key Assumptions and Limitations

The key assumptions and limitations of our research problem can be stated from two aspects:

1) For the algorithm part, there is no relevant datasets for our machine learning solutions. Usually, machine learning models needs to be trained by the pre-selected datasets in order to adapt themselves to the application environment. In other words, most of the traditional solutions for the similar *Anomaly Detection* problems can only work in an offline manner (testing after training). However, in our case, it is unrealistic to expect that there would be a relevant datasets for the training purpose. That means, the solutions proposed by us in this thesis must be able to work in an online manner (testing while learning). Online can also promise a better user experience since users will not have to wait for a period (gathering datasets for machine learning) before the application become well-functioning.

   Here, in order to avoid any unnecessary confusion, we clarify the differences among the terms of “online”, “real-time” and “offline” in the following fashion:

   “Online” refers to the situation of handling data instance when it arrives. “Offline” refer to the situation of mining an available offline dataset. “Real-time” is concerned with the time taken to handling data instances. Generally speaking, it is possible to handling data instances in an offline but real-time manner if the mining results can be obtained in real-time. However, for an online approach, non-real-time is totally pointless. That means, the term “Online” has already given the guarantee to the “real-time” property of an approach. [1]

   Additionally, an *incremental approach* is not necessarily online. Generally speaking, an *incremental approach* refers to the approach of first offline mining datasets and then online handling upcoming additions. In this sense, an incremental approach is essentially a mixed approach for which the prepared datasets are still needed.

[1] These views came from the email discussions with Varun Chandola, the first author of literature [7] “Anomaly detection-a survey”.

This limitation (no datasets at all) decides that we have to adopt a pure online approach in the proposed solutions (Learning automata based).

2) For the prototype part, we can't expect more advanced infrastructures other than a simple Wi-Fi based environment for the purpose of testing our prototypes. This situation can be formulated as the following statement:

If we want to build a prototype that is going to be tested in real life in Grimstad, we cannot assume any other hardware than the two or three Wi-Fi-based phones that are planned included in the prototype. E.g., if two people happen to be in the same restaurant in Grimstad, we may assume that their phones will be able to detect their presence using Wi-Fi, however, we may not as easily assume that any other supporting infrastructure will be present for this purpose. We can of course build a limited testing environment where any conceivable supporting infrastructure can be configured, however, that will make the prototype much less useful in practice (until such infrastructure is available everywhere our prototype is supposed to work).

That means, the prototype should be built within the common Wi-Fi framework. Since we cannot expect any other advanced infrastructural technologies, it is really a big challenge for us to design the architectures and security schemes of the prototype within a relatively narrow technical framework.

1.5 Our Contributions

To the best of our knowledge, most of the prior research work in the *Anomaly Detection* field only addressed the models operating in an offline manner [7]. With regard to the context-awareness issue in ubiquitous environments as described in section 1.2, in order to pursue a better user experience, our solutions would illustrate a possible online approach which can be fit into mobile services. Considering the resource-constraint nature of mobile applications, the proposed solutions are also light weight. That means, the computational complexity of the proposed solutions can be acceptable in resource-constraint mobile platforms. Our effort to find such a solution supporting both online and light weight features for mobile services may arouse interests in the relevant research communities.

The real-world testing of algorithms in a ubiquitous computing environment could be a very expensive task. In view of this, computerized simulation should be a good alternative in order to empirically show how well the algorithms can work. In this thesis, we presented extensive empirical results to demonstrate the performance of the proposed solutions in different noisy environments. We take Boolean values signifying meeting or not at the current day as input to the proposed algorithms and then get alerts reminding a friend is nearby as output. We have defined number of false alerts (accuracy) as the performance criteria for the evaluation work. Our empirical results demonstrated that the performance of the proposed solutions is generally scenario-dependent. That means, it's hard to identify a "silver bullet" for all the cases—a "good enough" performance is depending on the actual application environment and also the practical requirements. However, since the required performance level for a practical application should be decided by the domain experts or industrial communities, without loss of generality we presented how the accuracy of the proposed solutions may vary with different factors in our research. Hence these empirical results will help the industrial communities to make the suitable design plans for a certain application environment. Additionally, from these empirical results, it is also straightforward for the readers to understand the important properties of our solutions.

The prototype developed by us would be a solid demonstration that our solution can be integrated into the mobile service. Different mobile services may put different requirements on the architectures and security schemes. The schemes proposed by us and the relevant discussions in the thesis could be found beneficial for the fellow researchers.

In particular, please note that our prototype was built up by making use of the existing infrastructural technologies. Therefore, our prototype may have good usability in practice.

Programming on a mobile platform could be a very different experience from programming in a desktop computer. The source code of the prototype attached to this thesis could be a good starting point for the subsequent development work.

To sum up, though it is a simplified research problem in the context-awareness computing domain, the significances of the research outcomes can be recognized in a generic level.

1.6 Thesis Outline

The structure of this thesis can be listed as the follows:

Chapter 1 is an introduction part of the Master thesis in which our research problem has been formulated in a scenario-based fashion.

Chapter 2 is a survey of the current theories and techniques regarding our research problem.

Chapter 3 describes the approach adopted for our research problem. The proposed learning automata based solutions are presented in this chapter.

Chapter 4 presents the extensive empirical results in order to demonstrate the performance of the proposed solutions.

Chapter 5 presents the process of prototype development.

Chapter 6 presents the conclusion of our research work in the thesis. Some challenges and problems are also summarized in this chapter for future studies.

2. State of the Art

2.1. An Overview of Anomaly Detection

Anomalies can be seen as a concept opposite to the normal behaviors or patterns. The basic idea is that it is assumed the normal data instances are always sourced from the normal behaviors. However, in different application domains, the definition of "normal behaviors" could be very different. For example, in an Intrusion Detection System (IDS), the normal behaviors refer to those operations or network traffics which do not violate the predefined security policies. In a damage discovery system, the normal behaviors could be defined as the structures or the functional components which are conforming to the safety requirements or the original design specifications. Despite there are many different notions of the normal behaviors in different application domains, an unambiguous definition of the normal behaviors is required as the prerequisite of recognizing anomalies in the given datasets.

Definition of Anomalies:

"Anomalies are patterns in data that do not conform to a well defined notion of normal behavior" [7].

From the above definition, it can be figured out that *Anomaly Detection* is actually a Pattern Recognition problem in nature. The purpose of *Anomaly Detection* is to find those anomalous patterns in the datasets. However, it is important to note that anomalies should not be those data instances of no interest to us. This is the most important distinction between "noises" and "anomalies". Since the noises will hinder the *Pattern Recognition* process in the data mining, analysts usually try to remove the noises before they can start mining the patterns. Anomalies in a dataset, though they are out of the normal data region, are always attracting much attention from us. In general, we are very interested in the reasons which caused these anomalies in order to discover the potential problems or risks in our systems and thus prevent the possible occurrences of more serious cases in the future. However, in a broad sense, if we can distinguish anomalous instances from those normal ones, it is possible to provide self-adaptive services. From this point of view, *Anomaly Detection* could be possibly applicable to the context-awareness research in the ubiquitous computing environment.

Then, some people may confuse *Anomaly Detection* and *Novelty Detection*. The distinction between these two superficially similar terms is obvious. *Anomaly Detection* aims to find out those anomalous patterns in contrast to the normal patterns. However, *Novelty Detection* aims to find out those previously unknown patterns which are *"incorporated into the normal model after detected"* [7]. The occurrence region of the anomalies and novelties are also different: the former should be outside the normal regions with the latter located within the normal regions. That's also why the "outliers" and "anomalies" can be used as interchangeable terms.

An *Anomaly Detection* technique involves several basic aspects. This can be shown in the

following figure (Figure 7):

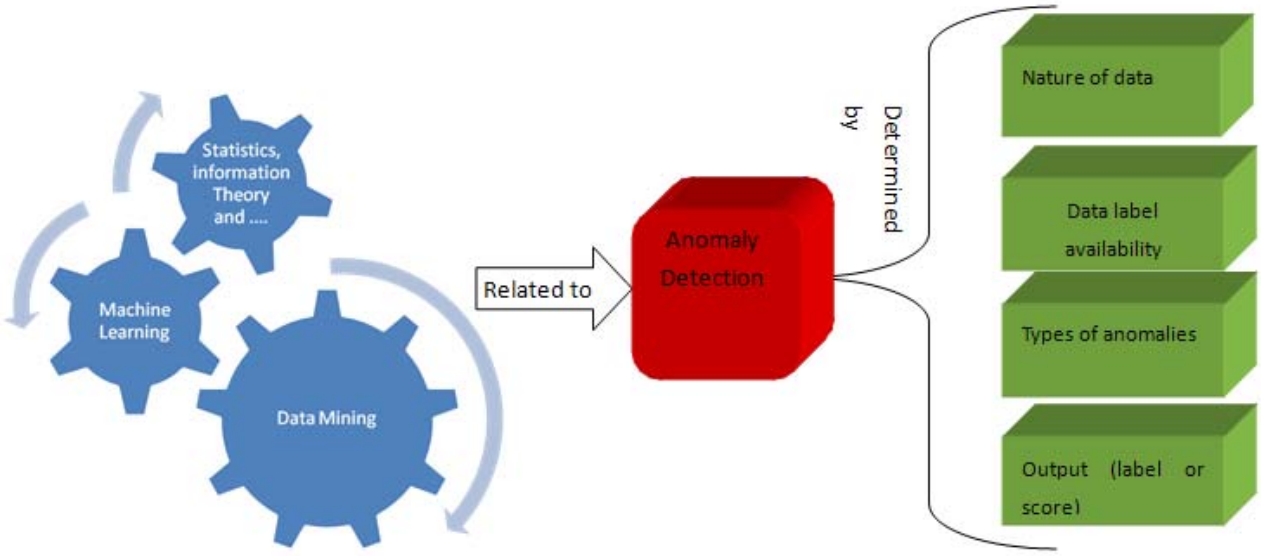


Figure 7 Aspects of Anomaly Detection

Anomalies can be grouped into different types. Assume that a dataset can be represented in a coordinate system with some normal regions. However, there are still some data instances falling outside the normal regions. This kind of "outliers" could be individual points by themselves or forming a small and sparse region. Since they have different features from the normal data instances, we called these data instances "point anomalies". Point anomalies are the simplest type of anomalies.

However, in many cases, the point anomalies type is far from sufficient to describe our research or practical problems. The insufficiency of point anomalies lies in the fact that they treat the features or attributes of data instances without any conditional constraints. A common case in our real life or industry is that the data instances are usually associated with a particular context. A typical example introduced in the literature [7] shows the basic principles behind contextual anomalies. As we know, 0 ℃ can be absolutely normal during the winter time. However, with the same temperature value in the summer, it is definitely an anomaly. Since the data instances with the same value will be treated differently in different contexts, we call them contextual anomalies. *"Contextual anomalies have been most commonly explored in time-series data"* [7] since temporal properties can be seen as the most common contextual attributes.

The last type of anomalies is the collective anomalies. According to its name, collective anomalies are a group of data instances which constitutes anomalous behaviors. Each or even several of these data instances cannot be recognized as anomalies by themselves. In other words, anomalous behaviors are represented by a particular combination of data instances, but each data instance in the combination can be a normal operation by itself. Collective anomalies could be another common form of anomalies found in the temporal data mining area.

Though the contextual anomalies and collective anomalies are much more complicated than the point anomalies in nature, they are usually being transformed to point anomalies for the convenience of processing ("*data reduction"*, [7]).

For a given *Anomaly Detection* problem, the nature of data should be amongst the most important factors to determine which techniques could be applied. The input data is usually complex in nature. For example, a data instance may consist of several attributes with each attribute of different types. In many prior cases, the relationships among data instances were more or less ignored. However, for the temporal data or the temporal-spatial data, the connections between a data instance and its neighbors obviously exist. In such a case, *Anomaly Detection* techniques should be adjusted in order to explore the internal relationships in the datasets. Another example to show why the nature of data is so important is that we can't apply stochastic models to solve the *Anomaly Detection* problems if we are not able to identify the probabilistic distribution hidden in the given datasets.

It has been mentioned that the *Anomaly Detection* problem can be transformed to a *Pattern Recognition* problem. Generally speaking, a *Pattern Recognition* problem can be addressed with two involved phases—training and testing. Training is a pattern discovery process or a modeling process of the probabilistic distributions in a dataset if the stochastic approaches are going to be applied. At this stage, training datasets should be collected as an experimental prerequisite. Since the datasets from different application domains have different natures, it is usually not that straightforward to identify a suitable machine learning algorithm. In particular, for a stochastic model, training the model to get better parameter estimations sometimes could be a really difficult task. The training phase falls into the following three categories:

1) Supervised learning: This mode requires the domain expert labeled both the normal data instances and anomalous data instances in the given dataset.

2) Semi-supervised learning: This mode requires the normal data instances or the anomalous data instances in the given dataset to be labeled (but not both of them). Since domain experts are usually very familiar with the expected normal behaviors in a system, only labeling the normal data instances could be much easier than labeling both the normal and anomalous ones. In view of this, the semi-supervised mode could be widely fit into many practical problems.

3) Unsupervised learning: This mode doesn't require training data and data labels. However, it should be adopted based on the assumption that *"normal instances are far more frequent than anomalies in the test data"* [7]. Many nearest-neighbors, clustering and stochastic approaches bear the unsupervised mode. However, it is usually the case that the performance of a supervised technique (e.g. accuracy) can be significantly improved if it is being operated in the semi-supervised setting.

Please note that the terms of "online" and "offline" should be distinguished from the terms of "supervised", "semi-supervised" and "unsupervised". All the three modes specified above can

only work in an offline mode since all of them need full datasets. The supervised and semi-supervised modes need datasets for both training and testing. The unsupervised mode needs datasets for testing only. That means, they are not able to support working in an incremental manner, or an online mode.

There are many techniques which can be applied into *Anomaly Detection* field. In order to present a comprehensive comparison of these techniques, a table is listed as the follows based on the categorical descriptions in the survey [7]:

Category	Typical Representatives	Key Assumption(s)	Computational Complexity	Advantages	Disadvantages
Classification Based	Neural Network; Bayesian Network; Support Vector Machine;	Classifier can be learnt in the given feature space	Training speed depends on the algorithms used; Generally fast testing speed;	Distinguish different normal classes; Fast testing;	Hard to find labels for multi-classes; Sometimes lack of anomaly score;
Nearest Neighbor Based	K^{th} Nearest Neighbor; Relative Density;	Normal data->dense neighborhoods, anomalies ->sparse neighborhoods	Generally, $O(N^2)$ required	Unsupervised nature; Semi-supervised mode will bring better performance; Easily fit to different data types;	Unsupervised mode result in missing anomalies; High false positive rate even for semi-supervised mode; High computational complexity in the testing phase; Challenging distance measures;
Clustering Based	Organizing Map(SOM); K-Means Clustering; Cluster Based Local Outlier Factor(CBLOF)	Belong to the clusters or not; Close to the centroid of clusters or not; Belong to the large/dense clusters or not;	Training speed depends on the algorithms used; Generally fast testing speed.	Unsupervised operate mode; Easily fit to complex data types; Fast testing speed;	Algorithm dependent performance; Not naturally optimized for anomaly detection

Statistical Based (Parametric and Non Parametric)	Gaussian Model Based; Regression Model Based; Histogram Based; Kernel Function Based;	In a specific statistical model, normal data->high probability area, anomalies->low probability area;	Depend on the statistical models; Generally HMM linear per iteration; Kernel based quadratic time required;	Statistically justifiable if the assumption holds; Anomaly score associated with confidence; May operate in unsupervised mode if the distribution estimation robust enough;	Hard to find a suitable probabilistic distribution; Nontrivial hypothesis tests; Histogram can't address the relationships between data attributes;
Information Theoretic Based	Kolomogorov Complexity; Entropy;	Anomalies->Irregularities in the information content	Basically exponential time complexity; Approximation require only linear time;	Unsupervised operate mode; No assumptions of probabilistic distribution;	Information theoretic measure dependent; Hard to get the size of data substructure; Hard to assign anomaly score;
Spectral Based	Principal Component Analysis (PCA); Compact Matrix Decomposition (CMD);	Normal data and anomalies can be differentiated in a lower dimensional subspace;	Depends on the concrete techniques; Standard PCA linear (data size) but quadratic (number of dimensions)	Dimension reduction, also can prepare a subspace for other techniques; Unsupervised operate mode;	Must make sure the normal data and anomalies be distinguished in the lower dimensional space; High computational complexity;

Table 1 Comparison of Anomaly Detection techniques

As to contextual and collective anomalies, *data reduction* [7] is usually used to transform a complex problem to a simpler case of point anomalies so that many existing techniques can be applied. However, though both contextual and collective anomalies can better reflect the complex nature in many application domains, they are also faced with a few difficult challenges. For example, it's usually ambiguous to identify the contextual attributes when anomalies are being considered "contextual".

An interesting problem in the collective anomalies detection area should be concerned with sequential anomalies. Time-series data, or more generally, data instances with the temporal aspect can be found very common in the real world. In most of the cases, we are focusing on the anomalous sequences in a sequence set, anomalous subsequences in a global sequence or even the anomalous frequency of a pattern in a given sequence [7].

If we consider *Anomaly Detection* in a set of many sequences, there would be some situations which could be very challenging to be tackled. For example, all the sequences in the set may not have equal length and also not well-aligned. Usually, the Markov models have been widely applied to modeling the temporal data. In general, the most popular models in the Markovian family include *Markov Chain* and *Hidden Markov Model (HMM)*. The latter one are actually the extension of the former one with hidden states added. In a sense, *HMM* could be dual stochastic processes: one is for the hidden states transitions; the other is for the transitions between the hidden states and the observable states. Temporal data *Anomaly Detection* is closely related to the research on temporal data mining. Readers who feel interested in this regard can find relevant publications of different focuses in this field according to the two bibliographies [14, 15].

After the discussion of different categories of *Anomaly Detection* techniques, it is obvious that each technique have its advantages and disadvantages. No such a techniques can be suitable for all the problem formulations. Selection of techniques should be determined by the nature of the problem and datasets. Also, the application domain often put computational complexity requirements on the applied techniques. Generally speaking, a technique with an efficient testing phase is usually associated with a time-consuming training phase. Every advantage has its disadvantage—the universal law of nature can be also found in the *Anomaly Detection* area.

Based on the above discussions, unfortunately, we couldn't identify a technique which can be directly applied to solve our research problem in a proper way. The underlying reason is that all these techniques are either offline or computationally complex. Additionally, they are not able to address the nature of our research problem very well. In the later sections of this chapter, the reader will see that our research problem actually poses two unique challenges on the traditional *Anomaly Detection* research. Section 2.2 and section 2.3 will discuss these two challenges and survey the related state-of-the-art, respectively.

2.2. Mining Periodic Patterns

With regard to our problem, the common Anomaly Detection techniques are faced with two big challenges:

1) It is very difficult to identify a technique which can handle the anomalies outside the patterns of periodic nature. Though temporal data *Anomaly Detection* has been discussed in the latest survey report [7], it is unconcerned about how to deal with temporal events/transactions which repeat themselves with a certain time period.

   Periodic patterns have been recognized as one of the two fundamental problems in the temporal data mining research [13]. Periodicity of events/transactions is a distinct feature with several nice properties in the temporal databases. In order to represent *Periodic Patterns* in the context of temporal sequences, first of all, we introduced the definitions of *event*, *event sequence* and *point sequence (*S. Ma et al. formulated in [8]):

   Definition of event, event sequence and point sequence [8]:

   An event is a pair (a,t), *a refers to the type of event (e.g. different items in terms of transactions),* t *refers to the discrete time point associated with the event. Then an event sequence* S *can be defined as an ordered set of event,* $S=\{(a_1,t_1),(a_2,t_2),(a_3,t_3),...,(a_n,t_n)\}$, a_i *represents different event types,* $a_i \in D$, D *is a finite set of event types,* $t_1,t_2,t_3,...,t_n$ *represents a linear discrete clock. Since* S *is ordered, it is required to make sure that* $t_i \le t_j (i \le j,(i,j) \in [1,n])$ *.if the event types were omitted from the event sequence* S, *we get a pure time point sequence* $\{t_1,t_2,t_3,...,t_n\}$ *which corresponds to a series of discrete points in a time axis.*

   In terms of *event sequences*, the conventional time-series analysis was only concerned with finding the *association rules* in a series (e.g. effect comes after causes). And also, the frequent event *episodes* in a time-series have also been studied by prior researchers.

   The periodic nature of patterns was initially recognized as cyclic association rules in [9]. However, the "perfect" *periodic patterns* discussed in [9] are very hard to be found in the real-world. Then researchers turned to consider the *periodic patterns* which appear in some temporal ranges but disappear in other temporal ranges. These contextual *Periodic Patterns* are called "*partially periodic pattern*" (Han et al. in [10]). With regard to the *partially periodic patterns*, the time axis can be divided into several non-overlapping segments—"*on-segment"* and "*off-segment"* [8]. According to their names, *on-segment* refers to the set of time points

in which the *periodic patterns* will appear as what it looks like. *Off-segment* refers to the set of time points in which the *periodic patterns* will not show up.

In order to gain better understanding of *on-segments* and *off-segments*, let's consider the following scenario:

***On-segment* and *off-segment* scenario**:

Consider a semester consisting of 20 weeks. For the first 12 weeks, Alice goes to the downtown restaurant for dinner together with her boy friend every Saturday (weekly pattern). However, after 12 weeks, Alice's boy friend goes abroad to attend an international conference for about 2 weeks. Then considering the absence of her boyfriend, Alice suspends the plan of going to restaurant temporarily in these 2 weeks. As her boy friend coming back, Alice continues to meet him in the downtown restaurant every Saturday as before. 4 weeks later, both Alice and her boy friend need to prepare for the final examinations, then the Saturday's dinner date would be cancelled again.

In the above scenario, it can be seen that the *weekly pattern* occurs for the first 12 weeks, disappear for 2 weeks, and then re-occur for 4 weeks followed by cancellation in the last 2 weeks. In such a case, we consider the first 12 weeks and the 4 weeks in which Alice and her boy friend meet every Saturday as two *on-segments.* Then the other two periods—2 weeks in the middle and the last 2 weeks can be seen as *off-segments*. Please note, in the *off-segments*, take the last 2 weeks in the above scenario for instances, Alice and her boy friend can still meet each other but without following the *weekly pattern* as before. That means, the exact *weekly pattern* will only be found in *on-segments*.

Then we can give out the definition of *on-segments* and *off-segments* as the follows:

Definition of on-segments and off-segments:

On-segments refer to sub-segments of time axis in which a given periodic pattern will appear in a persistent manner. Off-segments refer to sub-segments of time axis in which a given periodic might not be found. There is no time point subsuming to both the on-segments and the off-segments.

Then it's safe for us to introduce the definition of *periodic point sequence* and *partially periodic point sequence* based on the description in [8]:

Definition of periodic point sequence and partially periodic point sequence [8]:

A point $t_i (1 \le i \le m)$ *in the point sequence* $S_p = \{t_1, t_2, t_3, ..., t_m\}$ *always occurs every* p *time points with no larger than* δ *time points' deviation (time tolerance). Then we can call*

the point sequence S_p *as a periodic point sequence. If the periodic nature of points only exists in the on-segments, then we call* S_p *partially periodic.*

The periodic pattern and partially periodic pattern can be shown as the following figure (Figure 8):

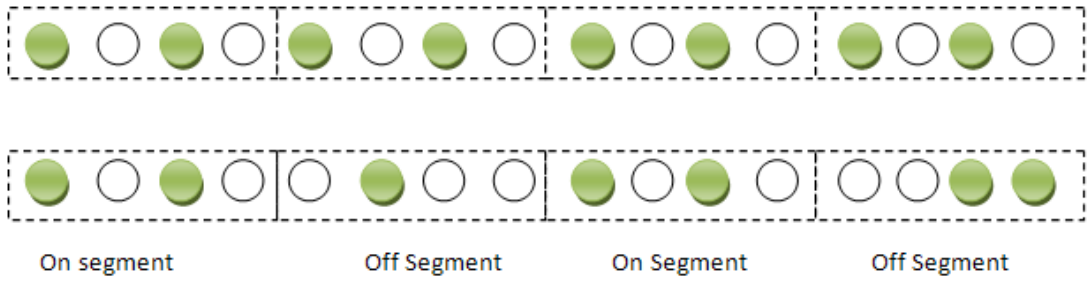


Figure 8 a) periodic pattern b) partially periodic pattern (p =2, δ =0)

Though *periodic patterns* discovery has received a lot of research concerns in the last decade, most of the prior research was only concerned with the offline operation mode. The evaluation of different algorithms always needs a prepared dataset. One example is that G.L. Chen et al. [11] from Dartmouth College use a 60-day system log of client traces (approx. 2259 traces in total) for their evaluation. Their goal was to discover the WLAN users' behaviors in their campus. The algorithm they used can be found in [8], in which the authors first adopted *Chi-squared test* to identify the length of periods, and then compare the association-first and period-first algorithms. In their case, the synthetic data has been used for performance evaluation. However, though the dataset is synthesized artificially, the operation mode was still offline.

As we know, the above discussion is unconcerned with in which form the point sequences can be represented. The time points can be seconds, minutes, hours, days and any other granularities. However, in our real life, the most common way of representing time domain should be calendar-based. In paper [8], A. k. Mahanta et al. are trying to discover the *periodic patterns* in the datasets with calendar date properties associated. The basic idea in [12] is to make use of the hierarchy in the calendar dates and *fuzzy* interval operation "*superimposition*" to extract *periodic* and *partially periodic patterns* in different levels. The authors test their algorithm in a real-world retail *Market-basket* dataset. Though their algorithm is so-called "*incremental in nature*" [12], this advantage was built on the fact that there already exist many previously discovered *periodic* and *partially periodic patterns*. However, this cannot be achieved without pre-processing a large dataset.

To the best of our knowledge, there is no pure on-line approach to handle *periodic patterns*. All the related work need a dataset prepared for the offline processing, though some of them don't require the users to specify the periods [8, 12].

In our research problem, an online approach is of great significance. As we know, the application in a new mobile phone must be able to work from the "zero day". It is unrealistic for users to wait for a long time and afford a large number of false alerts to make the application come to well-functioning. That means the demanded adaptability in our case should not require any previously collected data for training purpose. This brings us the second challenge.

2) Most of the existing *periodic patterns* discovery algorithms are no longer suitable for our solution since they are always requiring an available dataset. In other words, they are essentially offline.

   We need a solution which can be "testing while learning". The intuitive way to implement this idea is to allow learning from the current evaluation results. It should be a feedback-loop-like learning procedure. In view of this, we can classify our problem to the category of *adaptive control*, in which an *optimal policy* will be learned in an interactive online manner. Briefly speaking, an *optimal policy* here refers to a series of actions which can achieve the predefined optimality in a certain application environment. In fact, there are different types of optimality in different application environments. Formal definition of the (*optimal) policy* and different types of optimality will be presented in the next section (section 2.3). The characteristics of *reinforcement learning* remind us that the techniques within such a framework may provide inspirations to solve the problem.

2.3. Reinforcement Learning

2.3.1. The Basics

As emphasized in the previous discussion, a pure online solution is needed in our case. However, after the survey of many existing approaches, we found that most of them should go through two separated phases-- "learning" and "testing" phases. In other words, the discriminative function should be trained by pre-selected samples. That is, "*the credits are implicitly given beforehand as part of training procedure.*"[16]

Reinforcement Learning, more like a self-improving process, allows "credits" to be assigned during the evaluation/testing process. R. S. Sutton et al. in [24] distinguish *"reinforcement learning"* from *"supervised learning"* from a perspective of feedback in the learning procedures. The *"instructive feedback"* [24] in a supervised learning environment will tell that what a correct action looks like. In contrast, the *"evaluative feedback"* [24] in a reinforcement learning environment will evaluate each action made by the learning agents. In this sense, *reinforcement learning* is more like the case that a teacher only gives remarks to student's answers but never unveiling the correct answers.

This brings a very interesting challenge to the *reinforcement learning* research-- "Do we always trust the best action policy up to the moment?" Intuitively, we know that a current best policy will not always perform the best in the future evaluations since the changes in the environment are unpredictable. However, due to the previous experiences, we know that there must be some good points in the current best policy (that's why the policy can be called "the best" up to the moment). In view of this *"conflict"* [24] dilemma, we must consider a problem of *balancing the trade-offs of "exploration" and "exploitation"* [17, 24] in the learning process. Briefly speaking, *exploration* refers to the strategy of trying "new" actions which may lead to possibly even better performance in the future. *Exploitation* refers to the strategy of making use of the good aspects in the current "best" policy. An k arms bandit problem can be seen as an abstract paradigm of the problems of this category [17, 24].

Generally speaking, in a k arms bandit problem, pulling the high payoff arm (*exploitation*) all the time will lead to a sub-optimal solution, especially with a large number of plays [17]. That's because the "long" future has a considerably large space for *exploration*. In other words, *Exploiting* all the time implies missing future opportunities to discover an even better policy. However, on the other side, if the bandit was always trying to pull new arms without *exploiting* the current high pay-off policy, the positive reinforcements will be kept in a low level. There are a number of approaches developed to solve this balancing problem under their different assumptions.[24] We present this problem here only for the purpose of illustrating a basic challenge which the *reinforcement learning* tasks would be faced with.

In plain words, a r*einforcement learning* procedure can be described as: *a learning agent interacts with the dynamic environment to find the optimal action policy. At each step when an*

action has been chosen by the agent, the environment will give a reinforcement signal back to the agent. Then, the agent can adjust its behaviors based on the experiences learned in order to pursue the best performance.

The following figure (Figure 9) sketch out a basic reinforcement learning model:

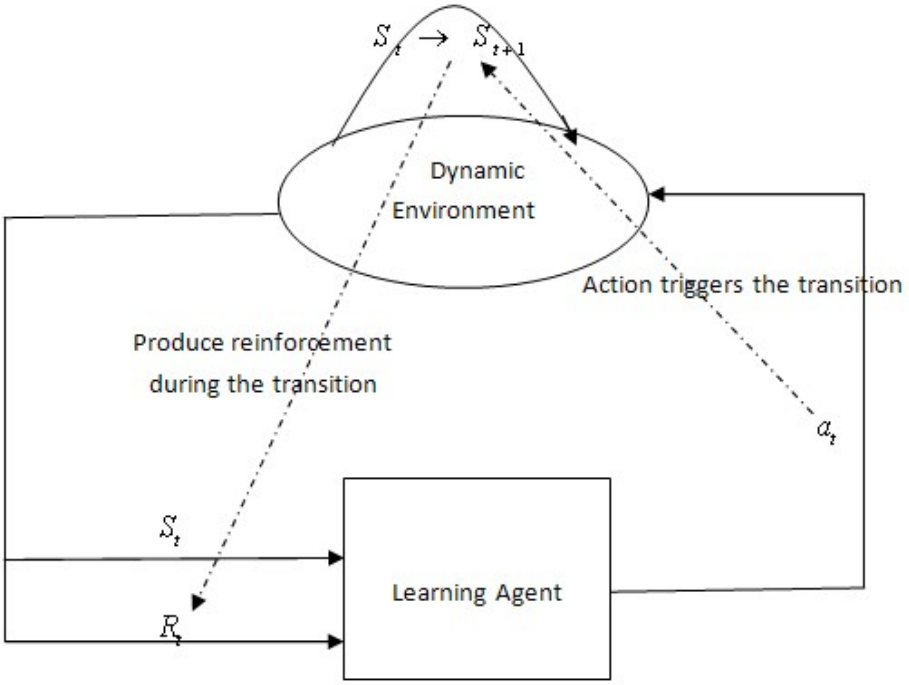


Figure 9 The basic model of reinforcement learning

Based on the above figure, an agent, aware of the current environmental state S_t, takes the action a_t. Consequently, the environment will transit to the next state. A reinforcement signal R_t will be sent back to the agent in the transition process.

A *reinforcement learning* scenario can always be explained by the *Markov Decision Process* (MDP).

A formal definition of MDP can be seen as the follows [19]:

A MDP is composed by a quadruple (S, A, P_a, R_a). *In which* S, A *are the state space and the action space respectively. For action* $a \in A$, P_a *defines the stochastic transition relation between the current state* S_t *(at time step* t*) and the successive state* S_{t+1} *(at the time step* $t+1$*).*

So we have:

$$P_a(S,S') = \{S_{t+1} = S' \mid S_t = S \wedge a_t = a\} \quad (2.1)$$

$R_a(S,S')$ represents the (immediate) reinforcement signal received when the state transition from S to S' happening.

For the purpose of avoiding confusion, we present the differences between *Markov Decision Process* and *Markov Chain* in the Figure 10. It is easy to figure out that both of the two stochastic processes have "*Markov property"* [20]. In a *Markov Chain*, the probability of transition to the next state is only determined by the current state (*first-order Markov Chain*, see equation (2.2)) or a fixed number of past states (*higher-order Markov Chain*). The transition probability function in a *first-order Markov Chain* can be described as the following equation:

$$P(S,S') = \{S_{t+1} = S' \mid S_t = S\} \quad (2.2)$$

However, for a MDP, considering its *Markov property*, the transition probabilities are determined by not only the current state but also the present action choice. Additionally, the MDP will send reinforcement signals during each transition. That means, MDP, compared with *Markov Chain*, is a more active and interactive process.

2.3.2. Different Models of optimality

A MDP aims to pursue the maximum cumulative reinforcements caused by the optimal action policy. However, associated with different applications, the models of optimality can be grouped into three different forms [17].

1) Optimal for the next n steps:

$$Exp(\sum_{t=0}^{n} R_{a_t}(S_t, S_{t+1})) \quad (2.3)$$

This *"finite-horizon model"* assumes that an agent has a limited and predictable life cycle [17].

2) Optimal for the next infinite steps:

$$Exp(\lim_{n\to\infty} \sum_{t=0}^{n} \gamma^t R_{a_t}(S_t, S_{t+1})) \quad (2.4)$$

$\gamma(0 \le \gamma \le 1)$ is called "*discount factor*". The above equation is actually a discounted sum of reinforcement signals. In contrast with equation (2.3), we know that the "*infinite horizon model"* [17] described by equation (2.4) will be reduced to the *"finite horizon model"* [17] described by equation (2.3) if $\gamma = 1$ and n is a finite number. The reason why the discount factor γ has to be introduced in the infinite MDP is that otherwise there would be a convergence problem since the maximum sum of reinforcements for each state can't be reached due to the infinite ongoing

steps.

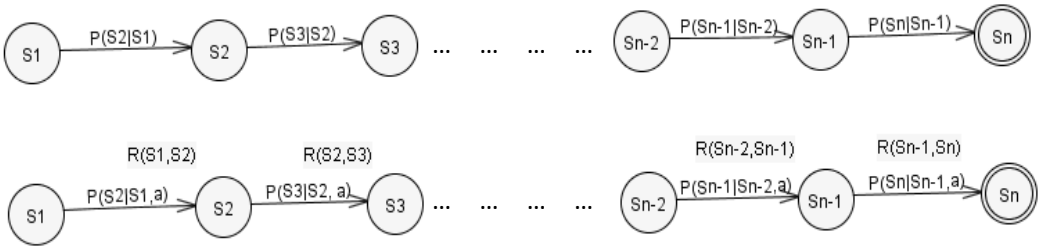


Figure 10 First-order Markov chain (upper part) and MDP (lower part)

3) Optimal for average rewards:

$$Exp(\lim_{n\to\infty}\frac{1}{n}\sum_{t=0}^{n}R(S_t,S_{t+1})) \tag{2.5}$$

In this case, the learning agent is concerned with the average performance in the long-term period. Note that handling such an average optimality problem is also to be done in the infinite horizon.

Apparently, the differences among the three optimality models are determined by different angles of seeing the *reinforcement learning* problem (short-term perspective or long-term perspective, the cumulative performance or the average performance). In a concrete application domain, to adopt one of the three models will depend on the desirable solution goal.

As pointed out in [17], the *"finite horizon model"* only applies to *"a system with a hard deadline"*. According to many prior studies, the *reinforcement learning* tasks with infinite horizon have attracted a lot of research interest due to its generality. Some well-known theories and algorithms have been developed and widely-applied. Dynamic programming, which we are going to introduce in the next section, can be seen as the theoretical foundation of them.

2.3.3. Dynamic Programming

Before presenting the explanation of dynamic programming, four notations should be introduced: An action policy π refers to a series of actions which will be performed as the state transiting from one to another in the MDP, π' representing the optimal policy. It is important to notice that π is essentially an array indexed by states. The value $\pi(S_t)$ is the action a_t which is decided by the policy π. The benefit of using the policy π lies in that this formulation can turn a Markov Decision Process to be a Markov Chain by binding the states with their corresponding actions [19].

$V(S_t)$ records the sum of reinforcements when starting from the state S_t and follow a certain action policy π. Then $V'(S_t)$ records the sum of reinforcements when starting from the state S_t and follow the optimal action policy π'. $V(S_t)$ and $V'(S_{t+1})$ can be formally defined as the equations (2.6), (2.7):

$$V(S_t) = Exp_{\pi}(\lim_{n\to\infty}\sum_{t}^{n} R(S_t, S_{t+1})) \quad (2.6)$$

$$V'(S_t) = Exp_{\pi'}(\lim_{n\to\infty}\sum_{t}^{n} \gamma R(S_t, S_{t+1})) = \arg\max Exp_{\pi}(\lim_{n\to\infty}\sum_{t}^{n} \gamma R(S_t, S_{t+1})) \quad (2.7)$$

Then our solution goal can be formalized as the following equation (2.8):

$$\forall t \qquad V(S_t) \to V'(S_t) \quad (2.8)$$

Moreover, based the *Bellman equation* [18], the value function $V(S_{t+1})$ of the state S_{t+1} can be recursively defined as (M.E.Harmon et al. in [18]):

$$V(S_t) = R(S_t, S_{t+1}) + \gamma V(S_{t+1}) \quad (2.9)$$

We know that the value function $V(S_{t+1})$ is an approximation of $V'(S_{t+1})$. If the errors can be denoted as $\Delta e_t, \Delta e_{t+1}$ at the time step $t, t+1$ respectively, then we have:

$$V'(S_t) - \Delta e_t = R(S_t, S_{t+1}) + \gamma(V'(S_{t+1}) - \Delta e_{t+1}) \quad (2.10)$$

From the equation (2.10), it is not difficult to figure out that the errors for each state have a property of cascade connection. Then we can have (a detailed proof can be found in [18]):

$$\Delta e_t = \gamma \Delta e_{t+1} \quad (2.11)$$

And

$$\Delta e_t = \gamma^n \Delta e_{t+n} \quad (2.12)$$

To satisfy the our solution goal stated by equation (2.8), it is important to point out that if the value function approaches the optimal value function in one state, then the same situation will be propagated to every state in a MDP.

There are two typical techniques in the dynamic programming approach—*Value Iteration* and *Policy Iteration.* They are all based on the *Bellman equation*. The most important difference between these two types of iterations is that whether the policy π can be utilized or not. Based on the *Bellman equation* and other prior studies in [16, 17, 18], we have:

In *Value Iteration* techniques, the procedure can be expressed as:

$$V'(S_t) = \max_a (R_a(S_t, S_{t+1}) + \gamma \sum_{\forall S_{t+1}} P_a(S_t, S_{t+1}) V'(S_{t+1})) \quad (2.13)$$

In *Policy Iteration* case, since a policy π should be taken in to account, the action should be specified by the given policy during each iteration.

Generally speaking, the procedure of *Policy Iteration* can be done with two steps:

1) Evaluation policy π:

$$V_{\pi_i}(S_t) = R_{\pi_i(S_t)}(S_t, S_{t+1}) + \gamma \sum_{\forall S_{t+1}} P(S_t, S_{t+1}) V_{\pi_i}(S_{t+1}) \quad (2.14)$$

2) Improve policy π_i to π_{i+1} ("*Greedy Policy*", [16]):

$$\pi_{i+1} = \arg\max_a (R_a(S_t, S_{t+1}) + \gamma \sum_{\forall S_{t+1}} P_a(S_t, S_{t+1}) V_{\pi_i}(S_{t+1})) \quad (2.15)$$

There are convergence guarantees of both *Value Iteration* and *Policy Iteration*. In equation (2.15), if $\pi_{i+1} = \pi_i$ is found, the process of policy improving can be terminated. It is safe to say that the optimal value function/the optimal policy is always finally reachable.

2.3.4. Monte Carlo Method

Both the Value Iteration and Policy Iteration seem to be very powerful for solving the *reinforcement learning* problems only if the prior knowledge of the MDP model is available. In a common situation, we have to assume that in a *learning agent* [16] is faced with an unknown environment. That means, the agent doesn't know which actions could trigger which reinforcement signals. It also has no knowledge of how the states of external environment will change in the learning process. *"Model-based"* [17] approaches should be replaced by *"Model-free"* [17] approaches in such a case. And moreover, if a "model-free" approach has been adopted, we can call this type of learning *"Autonomous"* [16].

Model-free means that we can't calculate the value function of each state as in the model-based approaches (we even don't how many states there are in an unknown environment). However, if we adopt the *Monte Carlo Method* simulation, the approximation results can be obtained by a number of *"repeated random sampling"* [21]. That means, a certain number of (pseudo) random series simulation can approach the true value which we are not able to obtain in case of incomplete environmental information.

A general way of doing this can be found in Literature [16].Under a fixed policy π, let a state S_t

be visited for N times (N is usually a large number). For the i^{th} time the state S_t has been visited, we denote:

$$V_\pi(S_t, i) = R(S_t, S_{t+1}) + \gamma R(S_{t+1}, S_{t+2}) + \gamma^2 R(S_{t+2}, S_{t+3}) + \ldots \quad (2.16)$$

In order to get an approximation of the true value of $V_\pi(S_t)$, we need to calculate the average of all these *discount cumulative sums*, and then we have:

$$V_\pi(S_t) = \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} V_\pi(S_t, i) \quad (2.17)$$

Though *Monte Carlo Simulation* can solve the problem caused by *"weak model"* to some degree, it will hinder the online manner of the learning process. In equation (2.17), no matter what number i is, *"a complete simulation run"* [16] needs to be done for the updating purpose.

2.3.5. Temporal Differences

Temporal Differences (TD) is in essence the prediction learning. It is originated from the *Monte Carlo Method* and dynamic programming we discussed in the above. The basic idea of TD is related to the *Markov Property.* However, the difference between TD and dynamic programming is that dynamic programming approaches only consider the immediate successive state as in equation (2.9), but TD considers the correlations of several future states [18].

When sampling the environment, under a fixed policy π, at time step t, TD(0) can be defined as the follows [22]:

$$V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(R_\pi(S_t, S_{t+1}) + \gamma V_\pi(S_{t+1}) - V_\pi(S_t)) \quad (2.18)$$

The above equation has the same expressiveness as the dynamic programming equation (2.9). However, in this case, the only difference is that it is being considered in an ongoing simulation process/ updating process.

Similarly, we can have TD(1) as the follows [16]:

$$V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha\{(R_\pi(S_t, S_{t+1}) + \gamma V_\pi(S_{t+1}) - V_\pi(S_t)) + \gamma(R_\pi(S_{t+1}, S_{t+2}) + \gamma V(S_{t+2}) - V_\pi(S_{t+1})) + \gamma^2(R_\pi(S_{t+2}, S_{t+3}) + \gamma V(S_{t+3}) - V_\pi(S_{t+2})) + \ldots\}$$

(2.19)

If we define $R_\pi(S_t, S_{t+1}) + \gamma V(S_{t+1}) - V_\pi(S_t)$ as *temporal difference* at time step t (denoted as

td_t) [16], then we can simplify the above equation (equation (2.19)) to the following equation:

$$V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(td_t + \gamma td_{t+1} + \gamma^2 td_{t+2} + ...) \quad (2.20)$$

As we know, γ is used to adjust the weights of future reinforcements in the *infinite horizon optimality model*. The value of γ is determined by the solution goal. Similarly, when we consider the temporal differences in the future, a parameter λ can be introduced to adjust the weights of future temporal differences.

Then it is safe for us to introduce the concept of TD(λ) as the follows:

$$V_\pi(S_t) \leftarrow V_\pi(S_t) + \alpha(td_t + \gamma\lambda td_{t+1} + \gamma^2\lambda^2 td_{t+2} + ...) \quad (2.21)$$

Intuitively, the larger the value of λ , the further in the future the predictive deviation will be taken in to the consideration.

This updating process of TD(λ) can be described in the following figure(Figure 11):

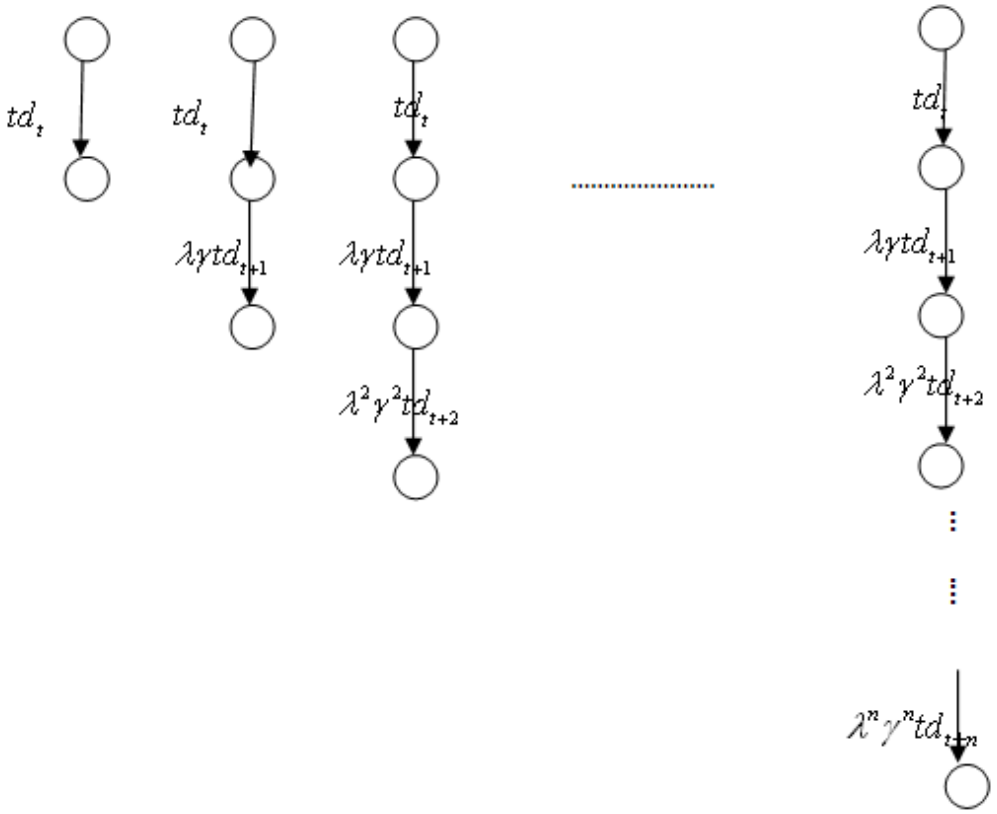


Figure 11 λ -learning iteration

If equation (2.18) is compared with equation (2.9), it is obvious that they are in essence the same. The only difference lies in their implementations. The equation (2.9) requires the process of solving linear equations, whereas the equation (2.18) demonstrates the dynamic

updating/approximation in the *Monte Carlo* process.

Considering the general rules existing in the TD(0), TD(1) and TD(λ), we can transformed the equation (2.21) to a *"causal"* [16]form by introducing the "*eligibility trace"* [16] $e(S)$.

Based the basic form of TD(0)(equation (2.18)), The equation (2.21) can be denoted as:

$$V_\pi(S) \leftarrow V_\pi(S) + \alpha(R_\pi(S_t, S_{t+1}) + \gamma V_\pi(S_{t+1}) - V_\pi(S_t))e_t(S) \quad (2.22)$$

Considering both the characteristics of equation (2.21) and equation (2.22), the $e(S_{t+1})$ can be defined recursively on the previous step $e(S_t)$ as the equation (2.23): (proposed by R. Sutton in [22], summarized in [16, 17]):

$$\begin{aligned} &if\ S == S_t \\ &then \quad e_{t+1}(S) := \gamma\lambda e_t(S) + 1 \\ &else \quad e_{t+1}(S) := \gamma\lambda e_t(S) \end{aligned} \quad (2.23)$$

Now, the equation (2.22),(2.23) make the direct implementation of TD(λ) possible.

2.3.6. Q-Learning

Assume a MDP in which each state can transit to a group of other states with different probabilities. That means, for state S_t, the successive state S_{t+1} is not deterministic. The transitions can be expressed as a probability matrix. We called this kind of MDP *"non-deterministic MDP"* [18]

The traditional *Value Iteration* requires every possible actions will be considered at each state. Formally, in a state space $\mathbb{S}$, the search of the optimality policy will be executed by following all the possible edges in a level, or breadth-first traversal.

For a deterministic state transition process, breadth-first traversal will not bring that much overhead. However, if it is assumed that we are faced with a group of the possible successive states with an unknown probability distribution, the case becomes much more complicated. The example shown in [18] reveals that a certain number (probably not sufficiently large) of sampling process will lead to a wrong convergence result. That means, there are at least two drawbacks in the traditional *Value Iteration:*

1) It requires the global picture of the MDP (*maps of the Markov domain* in [23])

2) A correct convergence usually requires an unacceptable computational complexity in a *non-deterministic MDP.*

Then if we consider the state space $\mathbb{S}$ as a tree-like structure as mentioned above, an intuitive idea to reduce the computational complexity should be change the breadth-first way of searching. That means, the searching process can start from a state and follow a fixed action to go to a deeper level.

From the perspective of set mapping, *value Iteration can* be expressed as:

$$\mathbb{S} \rightarrow \Re \tag{2.24}$$

That means, every state S_t will be mapped to a value function $V(S_t)$ which is the discounted cumulative sum of reinforcement signals starting from S_t.

Then when binding the state S_t and current action a_t, we consider the *Q-value* as $Q(S_t, a_t)$, the mapping relationship becomes as the follows:

$$\mathbb{S} \times \mathrm{A} \rightarrow \Re \tag{2.25}$$

Since we bind the state S_t and action a_t as a pair, then we can re-define the optimal value function $V'(S_t)$ at the state S_t in the form of *Q-value*:

$$V'(S_t) = \max_{a_t \in \mathrm{A}} Q'(S_{t+1}, a_{t+1}) \tag{2.26}$$

The equation (2.26) firstly considers the optimal value for a certain action, and then gets maximum value from the set of these optimal values (of different actions). This is a process of "twice-optimizing".

Then replace $V'(S_t)$ by the right side of equation (2.26) in *Value Iteration* Equation (2.13) [17]:

$$Q'(S_t, a_t) = R_{a_t}(S_t, S_{t+1}) + \gamma \sum P_{at}(S_t, S_{t+1}) \max_{a_{t+1}} Q'(S_{t+1}, a_{t+1}) \tag{2.27}$$

Equation (2.27) can be written in the form of simulation/updating process as equation (2.28) by replacing the equation (2.18) with the *Q-value* [17]:

$$Q(S_t, a_t) := Q(S_t, a_t) + \alpha(R_{a_t}(S_t, S_{t+1}) + \gamma \max_{a_t} Q(S_{t+1}, a_{t+1}) - Q(S_t, a_t)) \tag{2.28}$$

The convergence of Q-learning is also being guaranteed. L.P. kaelbling et al. in [17] points out that Q-learning is naturally *"exploration insensitive"*. That means the optimal policy will be found finally (convergence will be reached) no matter which kind of *exploration* strategies was adopted.

2.3.7. Comparison of Different Methods

The following figure (Figure 12) will show the differences among the *Value Iteration*, *Q-learning, Monte Carlo Simulation and TD* with respect to searching the state space $\mathbb{S}$:

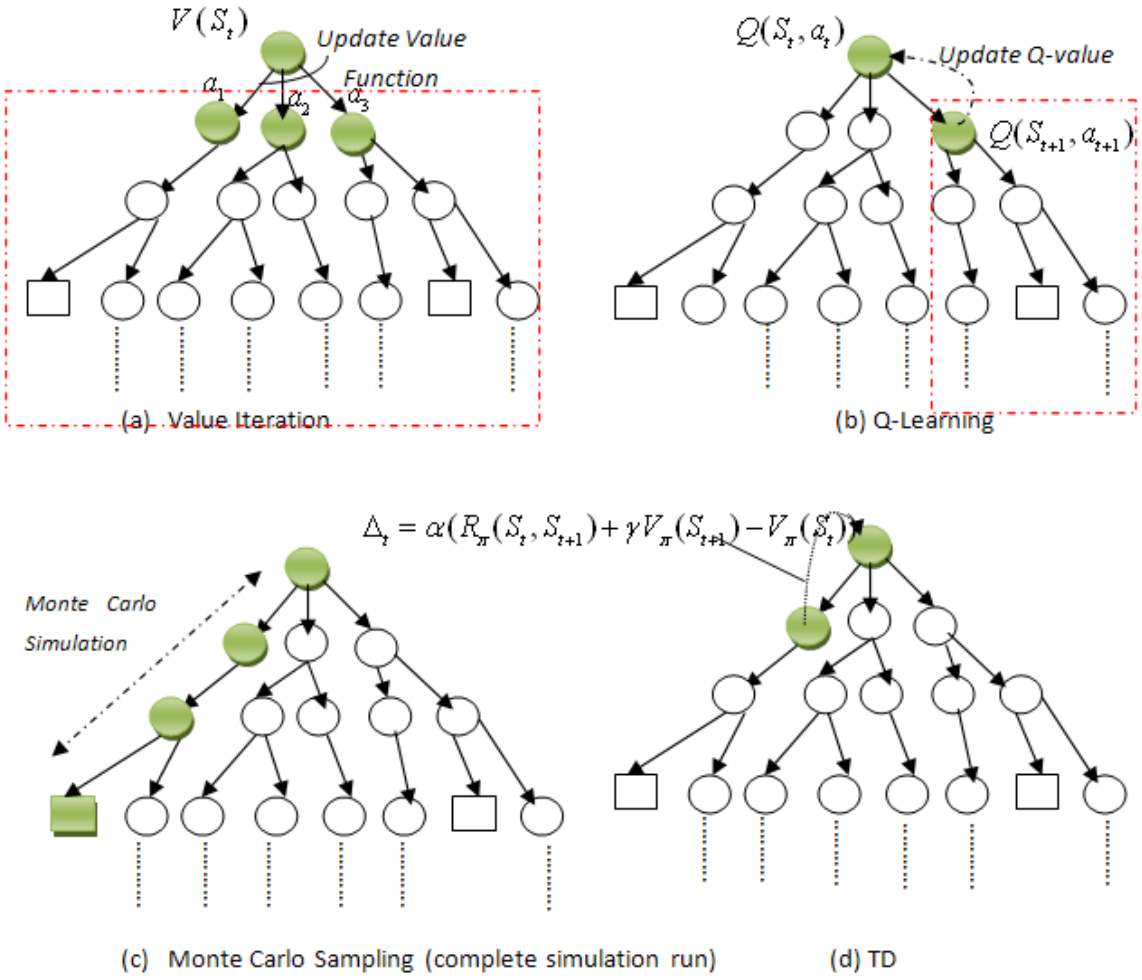


Figure 12 Comparisons of different RL methods

Please note that circle shape refers to the intermediate states, and the rectangle shape refers to the terminal states. The filled ones illustrate the states visited according to the iteration rules in different methods.

2.4. Summary

The state of the art chapter contains the theoretical frameworks for our studies in this thesis. We started from section 2.1 in which an overview of various *Anomaly Detection* techniques has been presented. However, we couldn't identify a technique among them to solve our problem directly

since our research problem does introduce some unique challenges. Then we analyzed the first challenge regarding the periodic nature of patterns in section 2.2. The approaches in this regard are still mining-based (offline). In view of this, in section 2.3 we turned to consider the second challenge regarding online mode and search within the *reinforcement learning* framework which addresses adaptive control problems in an interactive online manner. We can borrow the basic thought of *reinforcement learning* theories. Still, it is necessary to design the computationally efficient reinforcement schemes for our specific research problem.

3. Proposed Solutions

3.1. Introduction to Learning Automata

Though varieties of *reinforcement learning* techniques may provide powerful and formally correct potential solutions to our problem (*online*, *model-free*), they are generally computationally complex. The computational complexity (per iteration) for *value iteration* and *policy iteration* are listed as the following equation (3.1) and equation (3.2) (A refers to the action set, $\mathbb{S}$ refers to the state space, summarized in [17]):

$$\mathrm{O}(|\mathrm{A}||\mathbb{S}|^2) \tag{3.1}$$

$$\mathrm{O}(|\mathrm{A}||\mathbb{S}|^2 + |\mathbb{S}|^3) \tag{3.2}$$

It is obvious that both of them need exponential time cost per iteration. If the state space expands rapidly in a highly dynamic and complex environment, the computational complexity of the two methods will be unacceptable in many application domains. We expect that the proposed solution will not only be able to interact with the unknown environment in an online manner, but also can run with a low computational complexity for the sake of resources (power, memory, processor capability) saving in a mobile platform.

Since a light-weight solution is required in our case, we tend to find another learning model closely related to the above mentioned *reinforcement learning* schemes. *Learning Automata* (LA), an automaton-based learning model which aims to find the optimal action in the interaction with the random environment, has attracted our attention.

The interesting association point between *reinforcement learning* and LA can be found in solving the above mentioned k-armed bandit problem. Literature [17] points out that the LA, along with two other methods (*dynamic programming, Gittins allocation indices*), are three *formally justified* solutions to the bandit problem. O-C Granmo in [25] presented a Bayesian Learning Automata (BLA) to solve the *Two-armed Bernoulli bandit problem* (TABB).

The theory of LA originated in the 1960's. Tsetlin [26], a mathematician from Russia, first proposed using automaton to modeling biological systems for computer learning. Those early

models are generally fixed structure. A *Fixed Structure Learning Automata* (FALA) refers to those models with time-invariant state transition probabilities and output matrix. In contrast, if the state transition probabilities and the output can vary with time, LA models of this category can be called as *Variable Structure Learning Automata* (VSLA).

Formally, a stochastic LA can be defined as a sextuple $\{x, \phi, \alpha, p, A, G\}$ [27]:

1) x is the set of reinforcement signals. Usually x can only be 0 or 1 (*P-model*). If x can be obtained from a finite number of discrete values within the interval [0, 1], then the model is *Q-model*. In some cases, x can even be continuous values from interval [0, 1], we call this model as *S-model*.

2) $\phi = \{\phi_1, \phi_2, ..., \phi_s\}$ is the state space of a stochastic LA. ϕ_i represents the current state the LA is in.

3) $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ is the action set. In most of the applications, α can be seen as a set of finite number of actions. Therefore, we call LA models of this category as *Finite Actions Learning Automata (FALA)*. As the same as the reinforcements set x, α can also be an infinite set of continuous values ("*a real line",* [28]), then we call those models as *Continuous Actions Learning Automata (CALA).*

4) $p(t) = \{p_1(t), p_2(t), ..., p_r(t)\}$ is the probability distribution space on the action set α. t refers to the current stage (time step). $p_i(t)\ (1 \le i \le r)$ is the probability of choosing action α_i at the stage t. Since at each stage, one of the actions in the set p have to be selected, then we have:

$$\forall t \qquad \sum_{i=1}^{r} p_i(t) = 1 \qquad p_i(t) = \Pr(\alpha(t) = \alpha_i \mid p_i(t) \in p(t), \alpha_i \in \alpha) \tag{3.3}$$

Within the framework of probability space p, we can have the condition for the convergence as the following equation:

$$\lim_{t \to \infty} p_m(t) = 1 \tag{3.4}$$

m refers to the index of the maximum action probability. In light of equation (3.3), equation (3.4) is equal to:

$$\lim_{t \to \infty} \sum p_j(t) = 0 \qquad (1 \le j \le r, j \ne m) \tag{3.5}$$

Equation (3.4) (3.5) implies that a LA is converged only when a specific action was chosen with the maximum probability (other actions can be neglected due to the very low probabilities). The selected action might be the optimal (or sub-optimal) action we are trying to find. However, if the action was totally not desirable, we call such a case that the LA

wrongly converged.

In the early models p is a continuous probability space. That is, $p_i(t) \in p(t)$ can fetch every possible real values from the interval [0, 1]. B.J.Ommen et al. in [29] discussed the discrete probability space in order to achieve a faster convergence. The basic idea lies behind the discretized LA is that the convergence procedure can be done in an ad-hoc (one hop to another) mode, other than approaching by following a real line in the traditional continuous LA models.

5) A represents the updating algorithm on the probability space p. In terms of *reinforcement learning*, A is the reinforcement scheme. A general form of A can be denoted as:

$$p(t+1) = T(p(t), \alpha(t), x(t)) \tag{3.6}$$

Here, T can be a linear or non-linear function, which corresponds to *linear LA* and *non-linear LA* respectively.

The interesting phenomena here is that the T function could determine the *Markov property* of the probability space p. If $p(t+1)$ was merely determined by $p(t)$, then we call $p(t)$ ($for\ each\quad stage\quad t$) a *first-order Markov process*.

6) G is the mapping relationship from the state space ϕ to the action set α. Without loss of generality, we can assume G is a deterministic (one-to-one) function [27]. So we have $r = s$ for α and ϕ.

If we take the *estimator* algorithm [30] into account, the above definition of stochastic LA can be supplemented with a reward probability vector $d = \{d_1, d_2, ..., d_r\}$. $d_i (1 \le i \le r)$ corresponding to the individual reward probability of action α_i. Then the purpose of our LA turns to be pursuing the action with the maximum reward probabilities:

$$d_{\max} = \max(d_i) \qquad (1 \le i \le r) \tag{3.7}$$

This is straightforward when the reward probability d is a time-invariant vector. Based on whether d is time-invariant or not, we can group the random environments into two categories – *stationary environment* and *non-stationary environment.*

In terms of *reinforcement learning*, if the LA can be seen as a *policy iteration* procedure [31], the *stationary environment* means a time-invariant optimal action policy can be found in the environment, whereas the time-variant optimal action policy will be identified in a *non-stationary environment*.

Therefore, in contrast to the stochastic LA which only considers the immediate reinforcements from the environment, the purpose of *estimator* is to consider long-term rewards in the *policy iteration* procedure. B.J.Ommen et al. in [29] present the discretized *pursuit learning* schemes (*"a subset of estimator algorithms"* [29]) which can take both the long-term and short-term view of reinforcements into the consideration. Generally speaking, to the best of our knowledge, the LAs with estimator algorithms converge faster than those without estimator algorithms. The discretized schemes converge faster than those continuous schemes.

Based on the above discussion, the interaction between the LA and the random environment can be described in the following figure (Figure 13) [27]:

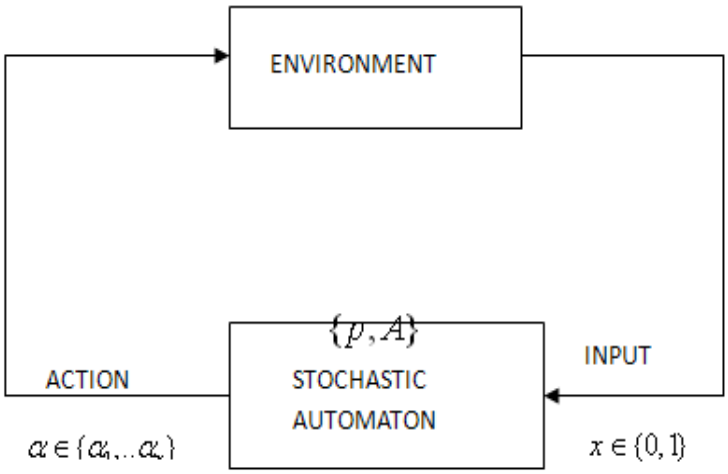


Figure 13 The basic model of learning automata

Generally speaking, the LA, according to its reinforcement scheme A, can be grouped into three main categories—*reward-penalty*, *reward-inaction*, *reward-ε-penalty*. The first scheme refers to increasing (decreasing) the action probability $p_i(t)$ as receiving rewards (penalties) after the action α_i was performed. The second one only increases the probability of a specific action α_i when receiving rewards. If a penalty was received, nothing will be done with the performed action. The last one is similar with the first scheme. The only difference is that the extent to which a penalty signal could affect the action probability will be much less severe than a reward signal. Definitely, there are other structures, e.g. inaction-penalty, but the above three schemes are the most commonly used ones.

If the reinforcement schemes of all the three types are *linear*, we can denote them as L_{R-P}, L_{R-I}, $L_{R-\varepsilon-P}$, respectively. L_{R-I} has some nice properties: it is ε-optimal in all the random *stationary environments* [27]. And moreover, it is demonstrated in [31] that L_{R-I} can converge

to a *"pure policy"*, whereas the *reward-penalty* scheme would make LA converge to a "*mixed policy"*.

From a different point of view, if we consider the penalty probabilities distribution $c = \{c_1, c_2, ..., c_r\}$ on the action set α, then our solution goal turns to be selecting the action with the minimal penalty probability:

$$c_{\min} = \min(c_i) \qquad c_i \in c \tag{3.8}$$

However, if a LA works only by pure chance, this is a common situation when a learning process just starts. That means, for each action $\alpha_i (1 \le i \le r)$, it has the probability of $\frac{1}{r}$ to be selected in the next round. then we can get a mean value of the penalty probability from c;

$$c_{mean} = \frac{\sum_{i=1}^{r} c_i}{r} \tag{3.9}$$

For a given action probability vector $p(t)$ at the stage t, we calculate the "*average penalty"* [27] on $p(t)$ as the follows:

$$M(t) = \sum_{i=1}^{r} c_i p_i(t) \tag{3.10}$$

Based the notation in equation (3.8), (3.9),(3.10), we have the following definitions to assess the performance of LA:

First of all, LA, which are able to learn in a random environment, can be called *"expedient"*[27], formally denoted as:

$$\lim_{t \to \infty} Exp(M(t)) < c_{mean} \tag{3.11}$$

"Expedient" LA can only ensure a better performance than a pure *Random Number Generator* (RNG). The self-improving property of LA needs a more strong constraint to be guaranteed. Then the term of "*absolutely expedient*" [27] can be introduced as the following equation:

$$Exp(M(t+1) \mid p(t)) < M(t) \tag{3.12}$$

Equation (3.12) convey the idea that an even better LA than the one constrained by equation (3.11) should keep moving forward to a better solution in which the average penalty probability

can be further reduced. However, the remaining question would be, to what extent, the LA can finally reach the desired minimum penalty probability $c_{\min}$?

Base on the above discussion, and we have already known that the minimal penalty probability should be $c_{\min}$ which is defined by equation (3.8), now it's safe to introduce the definition of optimality of LA as the following equation:

$$\lim_{t\to\infty} Exp(M(t)) = c_{\min} \tag{3.13}$$

Unfortunately, in most cases, it is unrealistic to expect the optimal action will always be picked out. However, what we can accept is the sub-optimality with a pretty low error rate. In view of this, it is necessary to introduce a very small real value ε to represent the error rate so that the definition of sub-optimality can be introduced as the following equation:

$$\lim_{t\to\infty} Exp(M(t)) = c_{\min} + \varepsilon \qquad \varepsilon > 0 \tag{3.14}$$

The LA fulfill the equation (3.14) can be called as *"ε-optimality"* [27]. The smaller ε is, the closer LA can reach the optimality.

In our case, the LA should have fast convergence since it is not desirable that many false alerts may bother the users. There are some significant prior research works addressing the efficiency (fast convergence) problem in LA models. To the best of our knowledge, in the 1980's, the *continuous pursuit algorithm* proposed by M. A. L. Thathachar and P. S. Sastry [30] has been proved converging much faster than the non-estimator LAs. Then B.J. Ommen et al. [29] proposed that the convergence rate can be further improved by discretizing the action probability space. If the pursuit algorithms were *generalized* to pursue a group of actions with higher reward probabilities, not only the possibly best action, an even faster convergence rate can be achieved [34].

The improvement of convergence rate for a single stochastic LA is kind of limited. M.A.L Thathachar in [28] point out that faster convergence can be achieved by the *"parallel operation".* A general rule in the traditional LA can be described as: *as the value of the learning parameter increasing, the convergence will speed-up at the cost of accuracy.* That means, the rate and accuracy of convergence are "conflicting" with each other within a single LA. The *"parallel operation"* [28] considers using a group of LA as a *"module"* [28] to substitute a single LA in the operation. A *LA module* will share the same action probability vector $p(t)$. The improvement of convergence rate without loss of accuracy lies in the idea that a group of selected actions and reinforcement signals will decrease the error rate in the updating process. The more LAs form a module, the faster convergence rate can be achieved.

It seems that the parallel algorithms can significantly improve the speed of convergence with the stability of accuracy maintained. However, this is done at the cost of memory. According to literature [28], if we want the convergence rate to be improved by N times, the size of module has to be expanded by N times. It is simply a case of trading space for time.

Another interesting way to improve the efficiency of LA can be found in [33]. The authors try to update a *basic probability vector* instead of updating the action probability space $p(t)$ directly. The *basic probability space* consists of a finite number of values. The advantage of updating one probability vector ($p(t)$) based on the other one (*basic probability vector*) is to guarantee the LA moving towards the convergence in a discrete-step manner. The simulation results in [33] demonstrate that the modified reinforcement scheme can improve the efficiency of LA significantly.

Though the prior research works significantly speed-up convergence of LA, we found it is demonstrated in their papers that the convergence of LA still needs hundreds of iterations. In our case, hundreds of iterations are unacceptable since in such a case too many false alerts would be caused before a pattern was learned.

And moreover, it is usually difficult to select a proper action probability updating algorithm A for a practical problem, especially in a *non-stationary* environment.

Base on the above discussion, allowing for both fast convergence rate and low computational complexity, we prefer to adopt *Finite Learning automata (FLA)* to solve the *periodic pattern recognition* problem in the thesis. A successful case of applying FLA can be found in [35] in which O-C Granmo and N. Bouhamala adopted the FLA to solve the *satisfiability problem*.

A formal definition of FLA can be found in [36], briefly speaking, a FLA consists of a quintuple:

$$\{\mathbb{S}, \alpha, X, F(\mathbb{S}, X), G(\mathbb{S})\}$$

1) $\mathbb{S}$ is the state space, consisting of finite number of states. $\mathbb{S} = \{s_1, s_2, s_3, ..., s_n\}$

2) α is the action set, $\alpha = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_r\}$

3) X is the set of reinforcement signals, $X = \{x_1, x_2, x_3, ..., x_m\}$

4) $F(\mathbb{S}, X)$ is the set mapping relationship. To be more specific, $F(s_t, x_t)$ means the transition from state S_t to the next state S_{t+1}, is determined by the current state s_t and the immediate reinforcement X_t.

5) $G(\mathbb{S})$ is also a set mapping relationship. $G(s_t) = \alpha_t$ means the action α_t is to be chosen at the current state s_t .

Comparing the definition of *FLA* and *SLA* (Stochastic Learning Automata), it can be figured out that *FLA* has introduced a state transition function F to replace the updating algorithm A in the *SLA*. In practical applications, the state transition function could be deterministic. In this sense, we can also call the *FLA* of this category *Deterministic Learning Automata* (DLA)

We will bring out our design of *FLAs* in the following sections.

3.2. Problem Analysis

Before presenting our design of *FLAs*, we would like to refine our problem a little bit more precisely.

In order to avoid unnecessary confusion, we will firstly consider the problem of detecting anomalous time in a single location. Then the readers will see in the last section of this chapter how the problem in the context of multi-locations can be composed by the problem in the context of a single location.

Here, we define the term “anomalous time” as the following statement:

In a certain level of time granularity, anomalous time refers to the time granules which do not conform to the well-defined periodic patterns.

However, in people’s real life, the possibilities of periodic patterns could be very hard to predict. The periodic patterns may differ from person to person. Moreover, even for a single person, the periodic patterns in his/her life may vary with time.

The following are three scenarios which can sketch out the situations discussed above:

1) Multiple periodic patterns within a single person’s life:

Alice always goes to school every day except weekends. She always goes to the cinema at a specific day every month (for example, 15th of each month). She only goes to the swimming pool at every Tuesday and Saturday. She goes to his office for the part-time job every the second day (That means, if she goes to the office today, then she will not go to office until the day after tomorrow.) She always comes back to the student apartment for relax and sleep every day.

2) Different people usually have different periodic patterns

Bob is one of the classmates of Alice. He goes to school every Monday, Wednesday and Thursday. He always participate the party held in Saturday. By the way, he dates with his girl friend and have a dinner together every two weeks. He always comes back to the student

apartment every day, just like Alice does at this point. In comparison with Alice's life, Bob obviously has a very different life style.

3) Time-variant periodic patterns for a single person:

Recently Alice is doing her master thesis. In the first several months, in order to save more time for working, she adjusted her schedule. She will go to school every day (not just weekdays as before). She canceled her plan of watching movies every month (no longer once per month in the cinema). She is no more doing her part-time job during the thesis phase (no longer going to the office every the second day). However, she still keeps her sport hobbies (go to swimming every Tuesday and Saturday as before). Since the thesis work needs the regular consultations, she will meet with her supervisor once per week in the first several months. The frequency will be adjusted to one time every two weeks in the last few months of her thesis work since she want to leave more time for the paperwork.

It is important to note that the periodic patterns in our problem should be the periodic patterns for communal meetings other than the patterns belonging to a single person. However, the characteristics in the above three scenarios can be found in our case—*Multiple periodic patterns for a single group, different groups usually have different periodic patterns, Time-variant periodic patterns for a single group.*

In view of the multiple-level complexities stated above, we find out that it is extremely hard to find a *FLA* solution which is suitable for all these individual-specific and time-variant periodic patterns.

However, when analyzing the problem, we found out it is necessary to sketch out several important periodic patterns for our research. In view of the practical situations, we only consider the *daily pattern, weekly pattern, monthly pattern, weekday pattern* and *the second day pattern* in our thesis. An interesting point is that the above mentioned five periodic patterns can actually be grouped in to the following two categories:

1) The first three patterns can be considered within a single solution template. The reason is that all of them are concerned with the problem of "consecutive occurrences". The only difference is time granularity we are addressing. If we consider the consecutive days, then it is the *daily pattern* we are trying to recognize. The *weekly patterns* and the *monthly patterns* just simply change the time granularity to the granules of week and month. For example, if we consider the meeting on every Wednesday, then the other six days within a week (from Monday to Sunday, except for Wednesday) have no impact on the *weekly pattern* of "every Wednesday". Similarly, the *monthly pattern* is only concerned with a specific day in every month. The other days in a month are of no significance on a specific day's *monthly pattern*. To sum up, for *weekly pattern* and *monthly pattern*, the consecutive granules we are taking into consideration turn to be weeks and months respectively, without other complexities introduced in comparison with the *daily pattern*.

2) The last two patterns can be considered within another solution template. And moreover,

the solution template for these two periodic patterns will also be applicable to many other "odd" periodic patterns, such as "two days on, one day off", "three days on, two days off". In our case, the *weekday pattern* and the *second day pattern* are just "five days on, two days off", "one day on, one day off" periodic patterns respectively. All these periodic patterns can be described as "x days on followed by y days off". In a sense, the *weekday pattern* can be seen as a collection of five *weekly patterns*—from every Monday to every Friday. Intuitively, considering the *weekday pattern* in the time granularity of days will help to speed-up the learning process.

3.3. Design of FLAs

Our solution is base on the *Hypothesis Testing* framework. Generally speaking, *Hypothesis Testing* refers to the procedure of making and verifying hypotheses. *Hypothesis Testing* was originally a term closely related to the *Concept Learning* [37].

In the *Concept Learning*, the training samples will *supervise* the learner to make correct or nearly correct hypothesis to judge whether an new statement will conform to the target concept implied by the training samples or not. The learner will produce an ordered (usually in a general to specific manner) hypothesis space in which new hypothesis will be added each time evaluating a training sample. The solution goal is to identify the most suitable hypothesis which can approach the target concept to the maximum extent.

The above discussed *Concept Learning* can be described as the following figure (Figure 14):

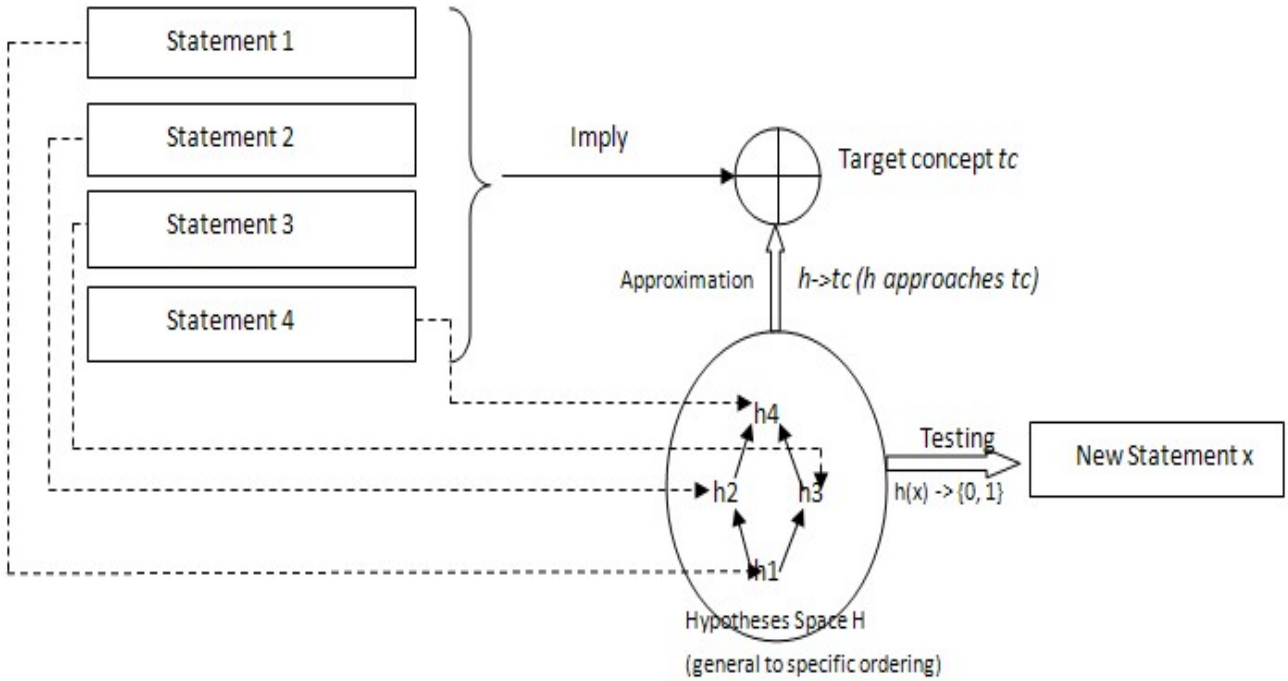


Figure 14 The basic model of hypothesis testing

From the above figure, it is obvious that the traditional *Concept Learning/ Hypothesis Testing* was supervised in nature. And a large and carefully selected training sample set will lead to higher approximation accuracy in an inductive process.

Now, in our case, what are the hypotheses?

For the *daily pattern*, the hypothesis can be stated as *"They always meet every day"*. Formally,

$$\forall day_i \in \{day_1, day_2, day_3, day_4, \ldots\},$$
$$meet(day_i)$$

The similar representation can be obtained by replaced *"day"* to *"week"* and "month" for the *weekly pattern* and *monthly pattern.*

For *weekday pattern*, the hypothesis can be stated as *"They always meet for the first five days in a week, and don't meet for the two weekend days"*. Formally,

$$\forall week_i \in \{week_1, week_2, week_3, week_4, \ldots\},$$
$$\forall day \in \{Mon., Tue., Wed., Thu., Fri.\} in \quad week_i,$$
$$meet(day)$$
$$\forall day \in \{Sat., Sun.\} in \quad week_i,$$
$$\neg meet(day)$$

For the *second day pattern,* the hypothesis can be stated as "*If they meet today, they will not meet tomorrow*". Formally, (Assume they meet at the first day)

$$\forall day_i \in \{day_1, day_2, day_3, day_4, \ldots\}$$
$$if\ i \mod 2 = 1$$
$$meet(day_i)$$
$$else:$$
$$\neg meet(day_i)$$

However, as we always emphasized, what we do need is an online solution. Inspired by state-of –the-art of *reinforcement learning*, the general reinforcement schemes for our problem can be designed as the following figure (Figure 15):

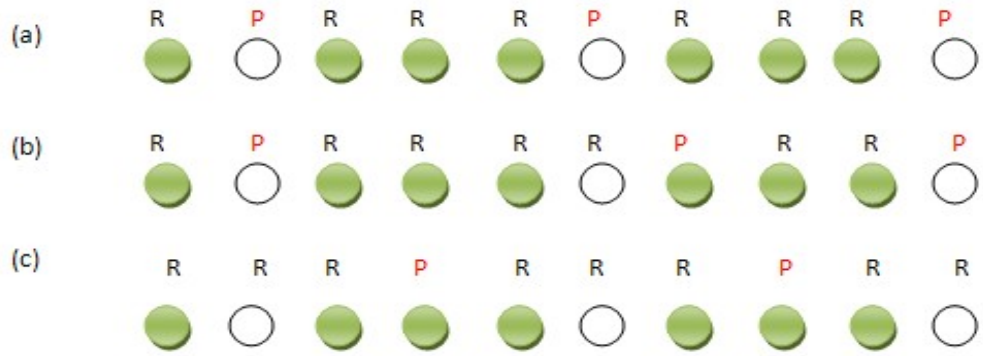


Figure 15 Reinforcement Schemes (R-Reward, P-Penalty)
a) for *daily pattern*, b) for *weekday pattern* c) for the *second day pattern*

The above figure shows the general reinforcement schemes for *daily pattern (a), weekday pattern (b)* and *the second day pattern (c)* respectively. Please note that the filled circles refer to the meeting days with the empty ones representing gap days ("gap" means there is no meeting). For *daily pattern,* each meeting days will be rewarded since they are conforming to the hypothesis of *daily pattern*. The gap days will then be penalized since they are "surprises" according to the hypothesis of *daily pattern*. For *weekday pattern*, the hypothesis predicts the first five days in a week should be meeting days with the last two days (weekend) in the same week being gap days. Then, as described in (b), the sixth circle is empty but with a reward signal associated. The seventh circle is filled but a penalty signal associated. These reinforcements are exactly the opposite of (a). However, if an empty circle emerges in the first five days of a week, like the tenth circle in (b), it should be penalized due to the violation of the hypothesis of *weekday pattern*. In (c), since this reinforcement scheme serves for the *second pattern*, the fourth and eighth filled circles which are rewarded in (a) and (b) are both penalized in (c). All the three empty circles in (c) are rewarded since they are correct representations of the hypothesis of the *second day pattern*.

To sum up, though it seems there are different forms to determine when to reward or penalty in (a), (b) and (c), the essence of the reinforcement schemes is exactly the same. This essence can be stated as the follows:

A learning agent can be rewarded (penalized) if and only if the current state (doesn't) comply with the prediction made by the hypothesis, though different patterns hold different hypotheses.

In a sense, our reinforcement schemes determine that they are all *prediction learning* [16] processes. However, at this stage, we only know when to reward or penalize for different hypotheses in a general sense. How to reward or penalize should be our next focus.

Before we really start to illustrate the structure of our *FLAs* design, an important thing to be understood is that there are actually no perfect periodic patterns in the real-life activities. The occurrence of each periodic pattern is always accompanied with noises. Sometimes the noises can be quite influential so that pattern discovery work might be confused by them.

Here, base on the true situation we define two kinds of noises in our case:

1) *Random gaps. A potential encounter that is a part of a regular pattern, however, that is randomly left out. In other words, the encounter was supposed to take place according to the pattern, but did not.*

2) *Random encounters. An encounter that is not part of a regular pattern. Instead the encounter happens by chance.*

Additionally, the *random gaps* can be called as "*false negatives*". The *random encounters* can be called as "*false positives*". They both are the accidental cases in people's real life. For examples, Alice may cancel the regular meetings with Bob due to the unpredictable sickness. Besides the

regular meetings, Alice may meet with Bob sometime outside the regular meeting schedules purely by chance (e.g. accidental meeting in the canteen).

Please note that *random gaps* and *random encounters* are two mutually independent random processes. However, the concurrence of these two types of noises is a common case in our real life. That means, the actual *random noises* are *synthesized* by these two types of random processes.

Intuitively, we feel that the pattern shifting over time will have similar effects as the *random gaps* and *random encounters.* This will be discussed in the next chapter by showing the relevant empirical results.

Since we pointed out that different patterns hold different hypotheses, it's very difficult and even farfetched to design a single *FLA* for recognizing all the different periodic patterns. A better way is to design a specific *FLA* for learning an individual concept (hypothesis). In the last section of this chapter, we will show how these *FLAs* would probably operate together.

3.3.1. DPFLA

Based on the above discussion, it is time to present our original design of the *daily pattern FLA* (*DPFLA*) as the following figure (Figure 16):

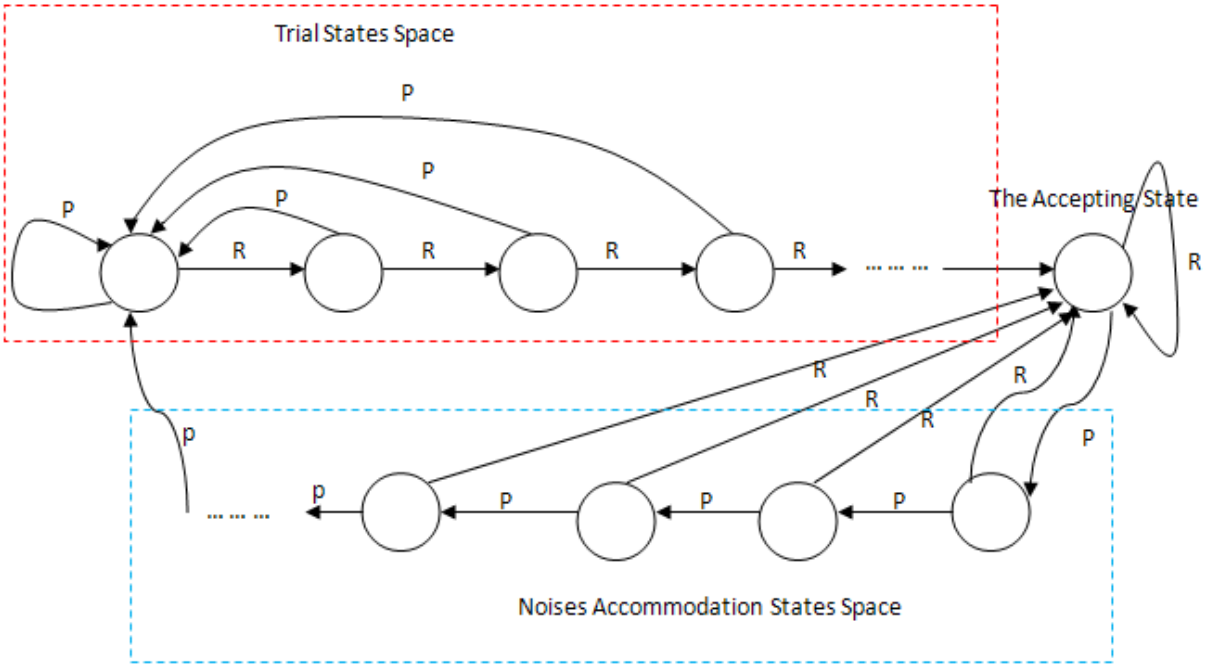


Figure 16 DPFLA (R-Reward, P-Penalty)

It is obvious that the above *FLA* is a symmetry structure. It can be divided into three parts:

1) The first part is the trial states space, formally denoted as $\mathbb{S}_{ts}$. We call the number of

states in $\mathbb{S}_{ts}$ as the size of $\mathbb{S}_{ts}$, denoted as $|\mathbb{S}_{ts}|$.

The significance of trial states space is to online check whether the ongoing days can form the *daily pattern*. Within this space, if the current day is a meeting day, in other words, conforming to the hypothesis of *daily pattern*, then the *DPFLA* will move forward a state. Otherwise, the *FLA* will go back directly to the first state.

Formally, the best scenario is that the *daily pattern* can be learned in the shortest time. That means, the learning process will traverse each state in the trial states space only once. The probability of the best scenario can be formulated as the following equation:

$$P_{best-scenario} = (1 - P_{gap})^{|\mathbb{S}_{ts}|} \tag{3.15}$$

P_{gap} is the probability of the occurrence of a gap day. If there are only *random gaps* noises, then we have:

$$P_{gap} = P_{random-gap} \tag{3.16}$$

Here, $P_{random-gap}$ refers to the probability of *random gaps* generated by a random process. However, if we take the *random encounters* noises into consideration, then the equation (3.16) turns to be as the follows:

$$\begin{aligned} P_{gap} &= P_{random-gap \wedge \neg random-encounter} \\ &= P_{random-gap} \wedge P_{\neg random-encounter} \\ &= P_{random-gap} * (1 - P_{random-encounter}) \\ &= P_{random-gap} - P_{random-gap} * P_{random-encounter} \end{aligned} \tag{3.17}$$

$P_{random-encounter}$ refers to the probability of *random encounters* generated by a random process. Here, since *random gaps* and *random encounters* are two mutually independent random processes, the *random encounters* may conflict with the *random gaps* accidentally. In such a case, the *random encounters* will compensate for the gap left by the *random gaps* noises. This phenomenon can be generalized and stated as the follows:

Assume an environment consisting of multiple kinds of noises, when these noises are occurring together simultaneously, one noise may compensate/hide the negative effect caused by another noise. In other words, they are conflicting with each other. This situation

can be called as "noises collision".

In our case, the collision of *random gaps* and *random encounters* will reduce the actual probability of gap days. Since the two types of noises are independent from each other, then the probability of collision can be denoted as $P_{random-gap} * P_{random-encounter}$, which is a part of the equation (3.17)

However, usually the best scenario will not be found in the real life. Traversal of the trial states space can be a multi-run stochastic process, since every time the occurrence of a gap day within the trial states space will lead the transition back to the first state.

Let's consider a forward-backward run. We denote the current state as $S_t(t \leq | \mathbb{S}_{ts} |)$, then the probability of reaching this state can be described as $(1-P_{gap})^{t-1}$. At the current state, the occurrence of a gap day will lead the transition to go backward. Then the probability of forward-backward run can be denoted as the following equation:

$$P_{forward-S_t-backward} = (1-P_{gap})^{t-1} P_{gap} \quad (3.18)$$

After backing to the first state, the next forward procedure will start once encountering a meeting day. Otherwise, the learning process will keep self-looping at the first state.

However, there is no guarantee that the next forward run will reach a further state ($S_{t'}$) than the prior run (S_t). We call the situation in which the next run will reach a further state than the prior run as *"expedient"* in our case. However, it's obvious that the learning process within the trial states space is hard to realize *consistent expedient.* That means the forward procedure in each forward-backward run may reach a further state than the last time, but not always (no guarantee).

From a probabilistic point of view, we can formulate the probability of an *expedient* forward-backward run as the following equation:

$$P_{\exp dient} = (1-P_{gap})^{t'} \quad t \leq t' \leq | \mathbb{S}_{ts} | \quad (3.19)$$

As mentioned above, $S_{t'}$ refers to the state the forward-backward run can reach at this

time. S_t refers to the state the forward-backward run reached at last time.

The reinforcement scheme in the trial states space can prevent the wrongly learned *daily pattern* caused by *random encounters* and other patterns.

For the sake of better understanding, let's consider the following structure as an alternative structure for the trial states space. (Figure 17)

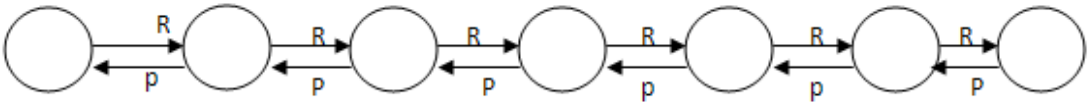


Figure 17 An alternative structure for trial states space (R-Reward, P-Penalty)

For the above structure, if the *random encounters* occur with a relatively high probability (>=0.5), then this structure would make the learning process more easily get out of the trial states space. The underlying reason is that the penalty signal weighs equally as the reward signal. This structure is also vulnerable for some other patterns like "x meeting days followed by y gap days". We say that if $x > y$, then this alternative structure will lead to a wrongly learned *daily pattern* since there is actually no *daily pattern* at all. If $x = y$, then the learning process will be caught in the infinite loops between two adjacent states.

However, our original design of the structure of the trial states space will place strict requirements on the testing data before the *daily pattern* can be learned. Only a certain number ($|\mathbb{S}_{ts}|$) of consecutive meeting days will make the *FLA* recognize that this is the *daily pattern*.

2) The second part of *DPFLA* is the accepting state S_a. The accepting state signifies that the *daily pattern* has been completely learned. A reward on the S_a will lead the transition back to S_a itself. That means a meeting day can reconfirm that this is the *daily pattern*. However, a penalty on S_a will lead the transition to the next part—noises accommodation states space, not back to the trial states space. The reason why we designed in this way is that a learned pattern should not be destroyed so curtly. It should be able to accommodate some *random noises*.

3) The third part of *DPFLA* is the noise accommodation states space $\mathbb{S}_{nas}$. As we mentioned in the above discussion, the *random gaps* are the main noises which have negative effect on the *daily pattern* recognition. The *random encounters*, by themselves, has no effect on the *daily pattern* since we can't distinguish a meeting day is caused by the expected *daily*

pattern or a *random encounter.* Intuitively, the existence of *random encounters* will alleviate the negative effects caused by the *random gaps* to some extent. Based on this idea, we design the noises accommodation states space mainly for the *random gaps* noises.

A notable characteristic of the noise accommodation states space in *DPFLA* is that the weight of a reward (triggered by a meeting day) should be much higher than that of a penalty (triggered by a gap day). The reason behind this is that it is assumed we already have the learned *daily pattern* in the noises accommodation states space. That means, the re-occurrence of a meeting days is a sufficient confirmation that the learned *daily pattern* is correct. The occurrences of gap days are considered as just some noises which may confuse the existing *daily pattern*.

In view of the above discussion, in the noises accommodation states space, a reward will lead the transition directly back to the accepting state S_a . A penalty will lead the transition to the next state. That means, once the *daily pattern* has been completely learned, the worst scenario in which the daily pattern will be destroyed in the shortest time, can be described in terms of probability as the following equation (similarly, we denote the size of the noises accommodation states space as $|\mathbb{S}_{nas}|$):

$$P_{worst-scenario} = P_{gap}^{|\mathbb{S}_{nas}|} \tag{3.20}$$

The above equation shows the probability of getting out of the noises accommodation states space in the shortest time. In such a case, each state in the noises accommodation states space would be traversed only once.

If P_{gap} is a very small number, then $1 - P_{gap} >> P_{gap}$. That means we have a much higher probability to confirm the learned *daily pattern* than destroy it at any state in the noises accommodation states space.

After introduction of the three parts, the learning process with *DPFLA* can be summarized as the follows:

At the beginning, the learning process traverses the trial states space. Then, it may reach the accepting state so as to demonstrate the daily pattern has been completely learned. For the sake of noises accommodation, a noises accommodation states space should be traversed after the daily pattern was learned. When the environment changes drastically and exceeds the noises accommodation capacity, then the daily pattern can be destroyed. In such a case, the FLA goes back to the initial state and prepares to re-learn the daily pattern.

An important point regarding the *DPFLA* is that the alerts will only be issued when the learning process are staying within the trial states space. In other words, once the accepting state is

reached, no alert will be issued until the learning process get out of the noises accommodation states space (comes to the initial state again). The underlying principle is based on whether the current situation (a meeting day or a gap day) is a part of the *daily pattern* or not.

3.3.2. Alternative DPFLA

Careful reader may discover there might be a problem associated with the original *DPFLA* after the daily pattern has been learned. This problem becomes even more interesting if the pattern shifts after it was learned.

Here, we would like to distinguish two situations:

1) *The learned pattern hasn't changed. We do not want the random noises to affect the stability of the learned pattern.*

2) *The learned pattern has been changed. We want the learned pattern to be destroyed as soon as the new pattern emerging. The recognition of the new pattern should be the task of another FLA, not the current one.*

Let's consider what will happen to the *DPFLA* if *the second day pattern* occurs after the *daily pattern* was learned. We have mentioned this is a common case in people's real life (*time-variant periodic patterns*). Then the noise accommodation states space will never allow the learning process get out of the noises accommodation states space and then come back to the unlearned status (the very first state). The reason is obvious. Each meeting day will lead the transition directly to the accepting state S_a despite that the gap days will temporarily make the learning process move forward only one state towards the exit point.

Inspired by the above discussion, we should design an alternative structure that allows a learned pattern can be canceled as soon as a new pattern emerging. Then, the only way to implement this is to adjust the ratio of penalty rate to reward rate. Here, we define: *the penalty (reward) rate refers to the number of states a penalty (reward) takes.*

However, the ratio of penalty rate to reward may depend on both the penalty rate and the reward rate. For the sake of clarity, we must fixed one of these two rates, and let the ratio vary as the other rate changing. In our *alternative DPFLA,* we set the reward rate to be one step (state). The penalty rate can be adjustable for the experimental purpose.

Moreover, we still welcome a symmetry structure for the *alternative DPFLA*. That means, the same reinforcement scheme should be applied to both the trial states space and the noises accommodation states space. The underlying reason is that the trial states space and the noises accommodation states space are essentially very similar. The only difference between them is that they serve for two different purposes: one is for learning the pattern; the other is for holding the learned pattern. The symmetry structure is also convenient for the simulation purpose since

we don't have to consider adjusting two different schemes of both the two sides at the same time in a single experiment.

The *alternative DPFLA* can be shown as the following figure (Figure 18):

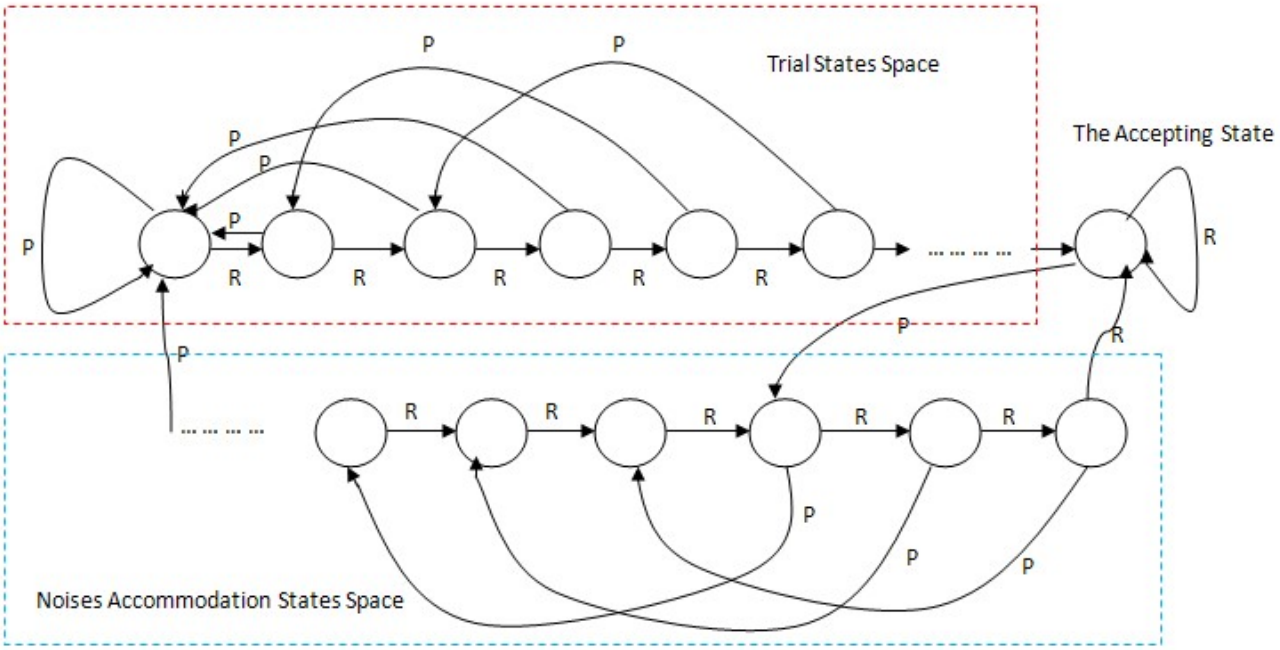


Figure 18 Alternative DPFLA (R-Reward, P-Penalty)

In contrast to the *DPFLA*, the *alternative DPFLA* can be differed from both the trial states space and the noises accommodation states space.

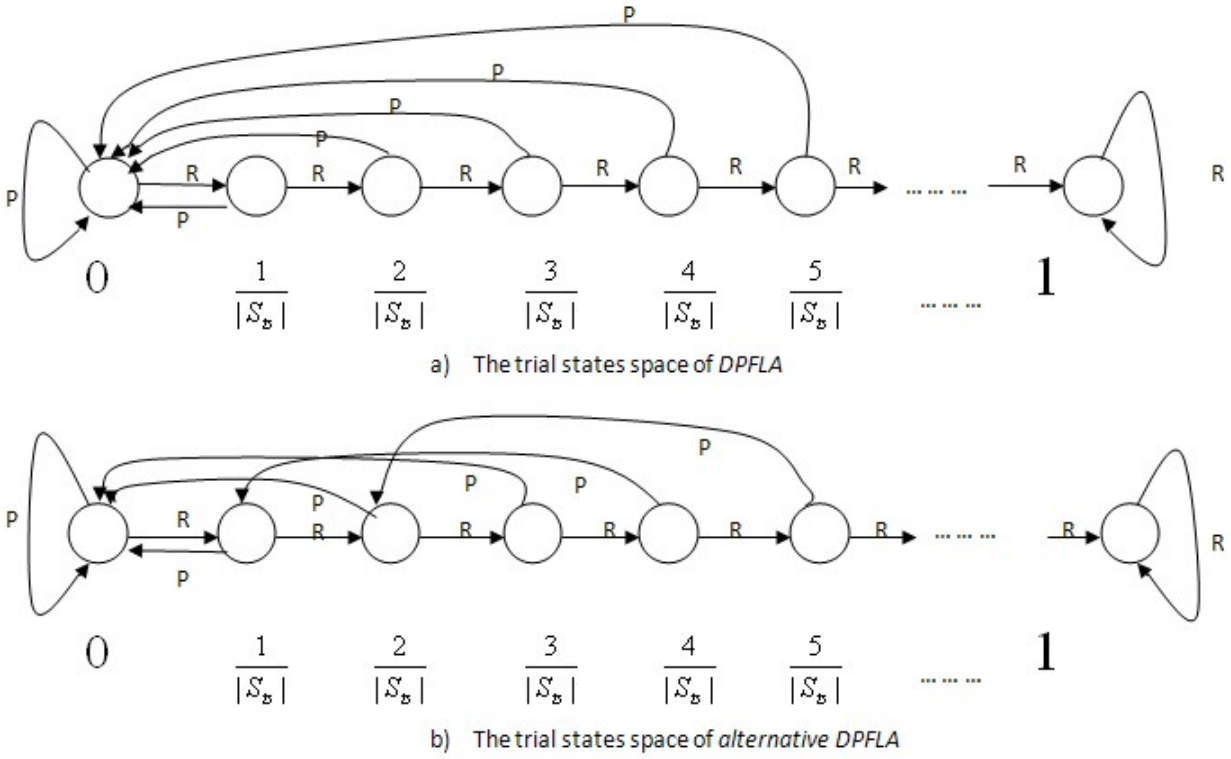
Firstly, in the trial states space, the principle of rewarding is being kept as the same as the *DPFLA*. That means, the learning process *of daily pattern* is still required to pass through a sequence of consecutive states. However, at this time, once a gap occurs, the learning process will not go back directly to the first state. It will go back by a certain number of states, which is determined by the penalty rate. In the above figure, penalty rate has been set to a fixed value 3, only for the purpose of demonstration.

In order to illustrate the difference we made in the trial states space of the *alternative DPFLA*, we introduce the term of *"confidence"*. As we know, reaching the accepting state S_a means the *daily pattern* has been formed. To quantify our discussion, we assigned the value 1 (%100) to this learned status, representing that we have learned the *daily pattern* completely. Therefore, the size of the trial states space $|S_{ts}|$ can be explained as the number of steps needed to reach such a learned status. The learning process can be explained as a process of pursuing the unity in the trial states space. This pursuit process has two distinct characteristics:

1) It is a *discretized* pursuit process. Each state in the trial states space corresponds to a discrete value of *confidence*. Since the number of the trial states is finite, there are a finite number of discrete values for selection.

2) It is a *non-monotonic* pursuit process. There may be a number of forward-backward runs in the learning process. In other words, the learning process is scenario-based.

The following figure (Figure 19) will show the pursuit processes of *DPFLA* and alternative *DPFLA* in terms of *confidence:*



a) The trial states space of *DPFLA*

b) The trial states space of *alternative DPFLA*

Figure 19 Representation of the trial states space in terms of *confidence* (R-Reward, P-Penalty)

From the above figure, it can be figured out the following differences:

1) For the *DPFLA*, each backward in a forward-backward run will make the accumulated *confidence* be cleared (return to zero).

2) For the alternative *DPFLA*, each backward in a forward-backward run will lead the accumulated *confidence* to a smaller but non-zero *confidence* or zero. To be more precisely, if we denote the current state as S_t, the first state as S_1, the penalty rate as pr, the *confidence* value as cfd then we have:

$$
\begin{aligned}
&if\ t-1 > pr \quad then \\
&\quad S_t \quad transits \quad to \quad S_{t-pr} \quad cfd = \frac{t-pr-1}{|S_{ts}|} \\
&elif\ t-1 <= pr \quad then \\
&\quad S_t \quad transits \quad to \quad S_1 \quad cfd = 0
\end{aligned}
\tag{3.21}
$$

Based on the above discussion, Since the trial states space of the *alternative DPFLA* can make use of the pr-step-earlier accumulated *confidence*, it is straightforward that if $|S_{ts}| >> pr$, then the learning process (before the *daily pattern* was formed) will be much less vulnerable to the *random gaps* noises.

Please note the difference between the *confidence* and *probability* in the above discussion. *Confidence* describes the static status which the learning process has reached right now. It is a demonstration that how much the daily pattern has been learned up to the moment. In contrast, the *probability* describes the likelihood that reaching a certain *confidence* value. For example, the confidence at the state S_t is $\frac{t-1}{|S_{ts}|}$, the probability of reaching this *confidence* value in a direct run (no backward procedure)could be $(1-P_{gap})^t$. In fact, the calculation of the *probability* of reaching a certain *confidence* can be dynamic and scenario-based. That means, it depends on the concrete execution/traversal path specified by a scenario.

Then let's consider the noises accommodation states space. In the *alternative DPFLA,* if the *second day pattern* occurs after the *daily pattern* was learned, this way of design will let the destruction of the existing *daily pattern* become much faster. However, in contrast to the original *DPFLA,* the capability of quick destruction of the learned *daily pattern* once encountering a new pattern will also bring new problems since a penalty signal weighs much higher than a reward signal. Intuitively, it will make the *alternative DPFLA* become much more vulnerable to the *random gaps* noises after the *daily pattern* was learned. For example, the alternative *DPFLA* only needs $|\frac{|S_{nas}|}{pr}|$ *consecutive* gaps to destroy the learned *daily pattern* whereas the original *DPFLA* needs $|S_{nas}|$. In view of this, the *alternative DPFLA* is generally pr times weaker of resistance to the *consecutive* gaps. The chapter regarding empirical results will show this point.

3.3.3. WDFLA, SDFLA and their Generic Template

Before we present the design of *FLAs* for the *weekday pattern* and the *second day pattern,* it is important to know the common ground of these two patterns. They are both composed by a certain number of meeting days followed by a certain number of gap days in a certain time period. In view of this, we started our design from the *Finite State Machine* (*FSM*) which can recognize the *words* specified by the corresponding periodic patterns.

Assume a periodic pattern consists of x meeting days followed by y gap days. Without loss of the essential characteristic, this pattern is equal to the following word consisting of 1s and 0s (1 denotes the meeting day, 0 denotes the gap day):

In comparison with the *daily pattern*, the most obvious difference here is that *the regular occurrences of gap days is also part of the pattern in nature.* In other words, the hypothesis places the requirements on the meeting days, gap days, and even their sequential order.

However, since the learning process is represented as a pure online evaluative process, we shouldn't expect any external mechanism (e.g. counters) to decide when to get out of the self-looping at a specific state. That means, we should design a structure suitable for the online evaluative operation by decomposing the self-looping procedure to a linear-feedback-like structure. This is also the reason why we prefer to use the *FLA* than the *FSM* in our case.

Inspired by the original *DPFLA*, we present our design of *FLA* which aims at recognizing the generic periodic pattern *"x meeting days followed by y gap days"* as the following figure (Figure 20). Please note that the *weekday pattern finite learning automata* (*WDPFLA*) and the *second day finite learning automata* (*SDPFLA*) are just two special cases of this structure.

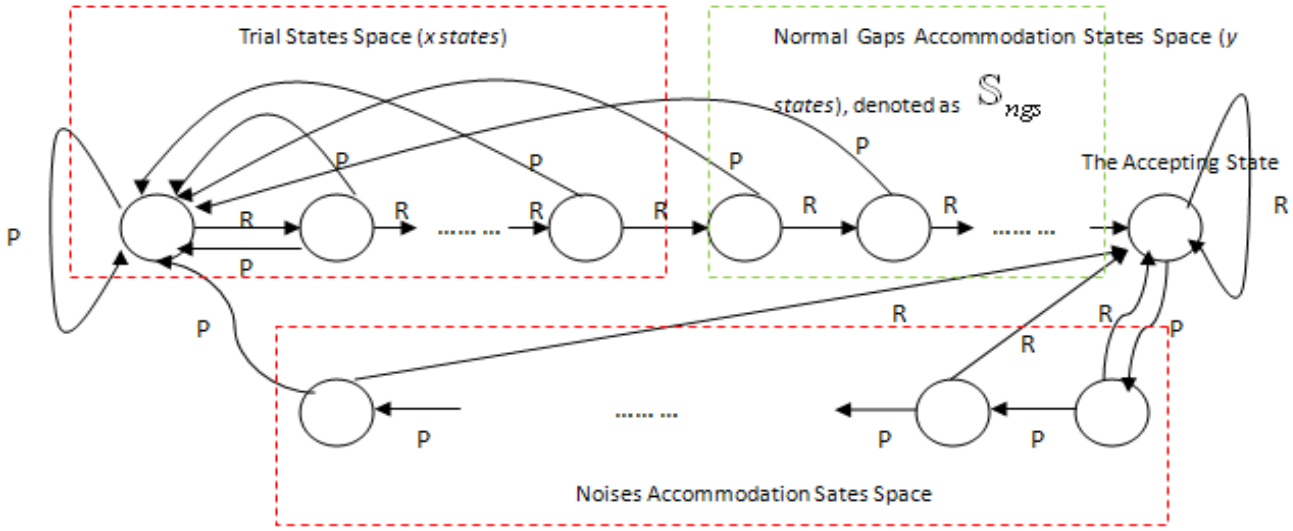


Figure 20 A generic FLA template (R-Reward, P-Penalty)

With regard to the generic FLA for recognizing the *"x meeting days followed by y gap days"* pattern, it is obvious that we have $|\mathbb{S}_{ts}|=5$ and $|\mathbb{S}_{ngs}|=2$ for the *weekday pattern*, $|\mathbb{S}_{ts}|=1$ and $|\mathbb{S}_{ngs}|=1$ for the *second day pattern*.

It seems the above figure (Figure 20) has the same structure as the *DPFLA* (Figure 16). However, at this time, we separate the normal gaps states space from the trial states space since the hypothesis for the *daily pattern* doesn't hold for the periodic pattern of *"x meeting days followed by y gap days"* any more.

1) For the trial states space $\mathbb{S}_{ts}$ in the above figure (Figure 20), the reward signals will be given on the x meeting days. That means, those meeting days are exactly what we expect. During the meeting segment (consisting of x meeting days), any occurrences of gap days will be penalized so that the state transition will directly go back to the first state, just like what we've discussed in the *DPFLA* section.

2) For the normal gaps accommodation states space $\mathbb{S}_{ngs}$, the reinforcement scheme has been reversed. That means, during the normal gap segment, only the consecutive y gaps will be seen as an expected sequence based on the hypothesis. The meeting days within the normal gaps accommodation states space will be penalized. The penalty signal will lead the transition directly back to the first state since we think the normal gaps should be evaluated with the same weight as the normal meeting days in the periodic pattern of *"x meeting days followed by y gap days".*

The best scenario of learning the *"x meeting days followed by y gap days"* patterns should be the one-run forward to the accepting state S_a without any intermediate interruptions (backward procedures). Formally, the probability of the best scenario can be described as the following equation:

$$P_{best-scenario}=(1-P_{gap})^x(1-P_{random-encounter})^y$$
$$=(1-(P_{random-gap}-P_{random-gap}*P_{random-encounter}))^x(1-P_{random-encounter})^y$$
$$=(1-P_{random-gap}+P_{ranom-gap}*P_{random-encounter})^x(1-P_{random-encounter})^y$$

(3.22)

To explain the above equation (3.22), it is important to notice that the *random encounters* noises will take advantage if the *random encounters* process and the *random gaps* process are conflicting with each other at the same day. In view of this, the "noises collision" will only

happen to the trial states space since *the random encounters* will cover *the random gaps* to some extent ($P_{random-gap} * P_{random-enconter}$). However, regarding the normal gaps accommodation states space S_{ngs} , only the probability of the *random encounters* $P_{random-encounter}$ will have effects on the learning process since it is not possible to distinguish *random gap* days and normal gap days in a practical environment.

From a systematic point of view, it is not difficult to figure out that our *FLA* for recognizing the *"x meeting days followed by y gap days"* pattern can be negatively affected by both the r*andom gaps* and *random encounters* noises. This is very different from the *DPFLA* or *alternative DPFLA* for recognizing the *daily pattern* since the *random encounters* will always compensate for the negative effects caused by the *random gaps* noises in the learning process with *DPFLA* or *alternative DPFLA.*

Since we continue to use the noises accommodation states space of *DPFLA* in our *FLA* for recognizing the *"x meeting days followed by y gaps days"* pattern. Intuitively, this practice may make our FLA naturally vulnerable to the *random encounters* noises. Moreover, we felt that the size of the noises accommodation states space $|\mathbb{S}_{nas}|$ we actually need for a practical application will have some kind of relationships with the number of the normal gap days in the *"x meeting days followed by y gap days"* pattern. We will discuss all these aspects in the next chapter regarding empirical results.

3.3.4. Game of FLAs

Based on the above discussion of our proposed solution, the readers may find out there are still two questions left:

1) *How do these FLAs operate together in a real-world application?*

2) *How can we deal with multi-location problems since the above mentioned periodic patterns were only concerned with a single location?*

For the first question, based on the definition of *Anomalous time* in the second section of this chapter, we can say that the meeting days should be considered as anomalies if they do not conform to any of the existing periodic patterns. Under such an assumption, when multiple *FLAs* operating together, the situation would be simply that no alert will be issued for a meeting day unless the current states of all the *FLAs* are in the left side of the accepting state $|\mathbb{S}_a|$. If we

denote the decision of "alert or not" of each FLA as a Boolean value d_i (*True* for alerting, *False* for not alerting), then the final decision of "alert or not" from a group of n *FLAs* should be $d_1 \wedge d_2 \wedge d_3 \wedge ... \wedge d_n$ (*A Conjunctive Norm Form).*

Please note that a *dictionary* data structure should be used in the game of *FLAs.* Since every *FLA* in a game can be seen as an independent *hypothesis testing* process, each hypothesis made by different *FLAs* in the group should be stored in the *dictionary* for future testing. Then, the overall reinforcement scheme can be described as the follows:

Those FLAs which made the hypothesis that can be confirmed with the actual situation (meeting or not) of the current day, should all be rewarded. Those FLAs which made the hypothesis that can be contradictory with the actual situation (meeting or not) of the current day, should all be penalized. Those FLAs which didn't make any hypothesis about the current day should stay still (Inaction).

For the second question, it is important to figure out whether there are any assumptions about the semantic relationships among different locations. This was supposed to be related to the problem of *location granularity*. For example, the reception office is *a part of* (a location within) the administration building. However, considering the same application will be deployed on all the parties involved in the meeting, the *location granularity* should be always the same for everyone (It depends on the accuracy of the localization mechanism adopted, the reception office or the administration building, but never both).

Based on the above discussion, there is no point in taking the semantic relationships among different locations into account in our case. We simply take different locations as different and independent units. For each fresh location (different from any locations we have recognized before), a group of *FLAs* for the specified periodic patterns should be created for the purpose of detecting *anomalous time*.

It can be summed up that the overall style of the proposed solution is that *each FLA for a specified periodic pattern, each group of FLAs for a single location.*

4. Empirical Results

The above discussion of our proposed solutions for the *daily pattern*, the *weekday pattern* and the *second day pattern,* has already included some conjectures of how these *FLAs* will work in the simulation environments. However, from a *quantitative* point of view, extensive experiments should be conducted in order to empirically demonstrate the performance of these *FLAs*.

The challenges of conducting experiments for these *FLA*s can be concluded in the following aspects:

1) The real-world experiments, which usually take a long time (e.g. several months), are basically infeasible due to the scope of this thesis. The alternative way is to design the computer simulation based experiments.

2) The basic principle of design scientific experiments is the method of *control variables.* However, the variables involved in our experiments can be quite a few. Take the *alternative DPFLA* for example. The probabilities of generating *random gaps*, *random encounters*, the size of the trial states space and the noises accommodation states space, the selection of penalty rate are all needed to be taken into the account when conducting the experiments. Each time we have to select a single variable for testing purpose with all other variables fixed.

3) For the sake of observation, **it is a must to select a consistent performance criterion**. Since the proposed solutions serve for the real-world application, the criterion must be selected on a practical basis. Since the proposed solutions aim to issue "correct" alerts when encountering an anomalous context, it is natural to consider how many false alerts will be issued in the experiments. Please note that we consider the number of false alerts as the representation of the accuracy. More false alerts imply lower accuracy, vice versa. In this sense, we will use both the terms "false alerts" and "accuracy" in the following sections. The number of false alarms/alerts, as a performance criterion, has been widely used in the simulations of *Intrusion Detection System*.

4) For the sake of clarity, we only test the *FLAs* for the corresponding periodic patterns. That means, *DPFLA* and *alternative DPFLA* for the *daily pattern*, *WDFLA* for the *weekday pattern* and *SDFLA* for the *second day pattern*. This "separate" experimental style will avoid the confusion caused by multiple periodic patterns since sometimes it is very hard for even us to artificially distinguish which alerts are false in the environments consisting of multiple and time-variant periodic patterns. However, this would not be a problem if only one periodic pattern is being considered at a time.

The following sections in this chapter will discuss the empirical results of *DPFLA*, *alternative DPFLA*, *WDFLA*, *SDFLA* one by one. A few comparisons will also be included in our discussion. Please note that for each experiment we have done, we will elaborate the experimental

conditions in the first place, and then explain the corresponding empirical results.

Additionally, each experiment was conducted for a 100 days' period. To get a better experimental accuracy, each data sample in the empirical results was sourced from the average value of 1000 simulation runs. Also, as we mentioned before, the number of alerts is to be counted only when 1) the current day is a meeting day and 2) the current state is located in the left side of the accepting state S_a. To make sure our experiments were done in a consistent manner, these general settings will apply for all the experiments in the following sections.

To emphasize again, we will use the number of false alerts (accuracy) as the **performance criterion** for all these experiments. We are going to present how the number of false alerts varies with different factors. Since the required accuracy (the *acceptable* number of false alerts) of a practical application should be decided by the industrial communities, here we are only planning to demonstrate and explain the trends implied by the empirical results, not the judgments.

4.1. DPFLA

The basic simulation environment for the *DPFLA* is a sequence of consecutive meeting days unless there is a special declaration (Experiment 4.1.4). The *random gaps* noises and *random encounters* noise*s* will be added based on the specific condition of each experiment.

Experiment 4.1.1

Experimental condition:

The objective of this experiment is to test the performance of *DPFLA* when working in an environment containing both *random gaps* noises and *random encounters* noises. The experiment will demonstrate that how the number of false alerts will change as the probabilities of *random gaps* varying from 0.05 to 0.45 in ascending order with the incremental step 0.05, under three different but fixed probabilities (0.1, 0.2 and 0.3) of *random encounters* noises. Here, $\left|\mathbb{S}_{ts}\right| = \left|\mathbb{S}_{nas}\right| = 4$. The empirical results can be shown as the following figure (Figure 21)

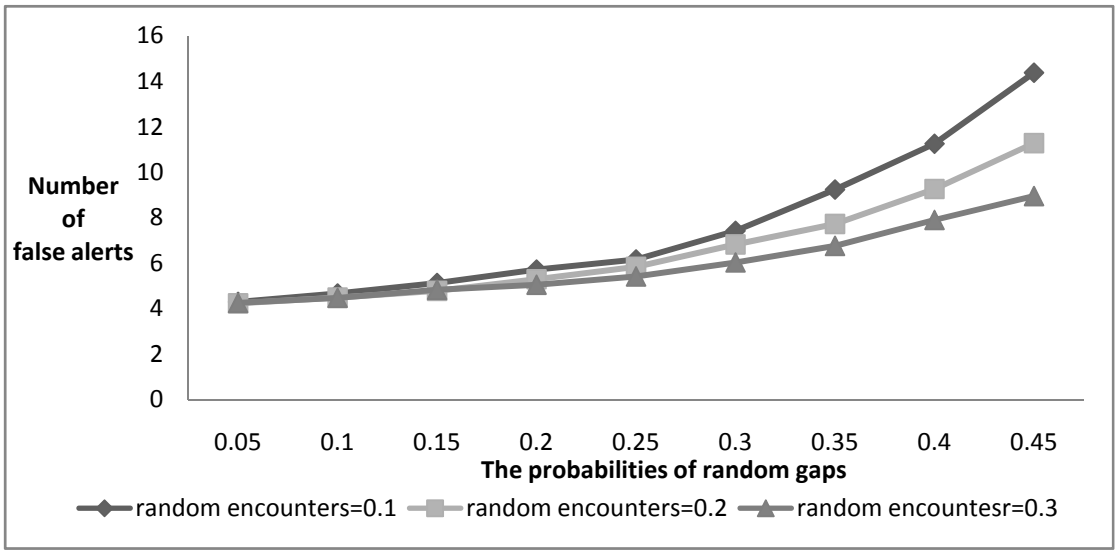


Figure 21 Experiment 4.1.1

Empirical results explanation:

It is obvious that for each fixed probability of *random encounters,* the number of false alerts will increase as the probabilities of *random gaps* goes up. However, if we put all the three curves in the above figure together for comparison, it can be figured out that the curve under a higher *random encounter* probability has the smaller number of false alerts as a whole, though the rising trend of the three curves remains the same.

This experiment shows that the r*andom gaps* noises have negative effect on the accuracy of the *DPFLA*. However, this kind of negative effect will be weaker as the *random encounters* being assigned a greater value since the *noises collision* (denoted as $P_{random-gap} * P_{random-encounter}$) will decrease the number of actual gap days in the simulation environment. This discipline would be particularly notable when the *random gaps* and *random encounters* are both assigned greater values. That's exactly why the tail ends of these three curves diverge more significantly than the front ends.

Please note that the probability of *random gaps* must be a value smaller than 0.5 (maximum 0.45 in our experiments). Otherwise, it is hard to say whether the *daily pattern* can still make sense. Then, the experiments in such a case (the probability of *random gaps* larger than 0.5) will become meaningless.

Also, for the *DPFLA,* there are at least the number of false alerts as the size of the trial states space $|\mathbb{S}_{ts}|$ since it is required the learning process has to pass through all the states in the trial states space at least once (in the best scenario). This is exactly the reason why all the three curves start from the point of value 4 in the *y* axis.

This experiment was mainly concerned with the impact of the *random gaps* on the number of false alerts. The following experiment will be mainly concerned with the impact of the random encounters.

Experiment 4.1.2

Experimental condition:

The objective of this experiment is to test the performance of *DPFLA* when working in an environment containing both *random gaps* noises and *random encounters* noises. The experiment will demonstrate that how the number of false alerts will change as the probabilities of *random encounters* varying from 0.05 to 0.45 in ascending order with the incremental step 0.05, under three different but fixed probabilities (0.1, 0.2 and 0.3) of *random gaps.* We still have $|\mathbb{S}_{ts}| = |\mathbb{S}_{nas}| = 4$ in this experiment. The empirical results can be shown as the following figure (Figure 22)

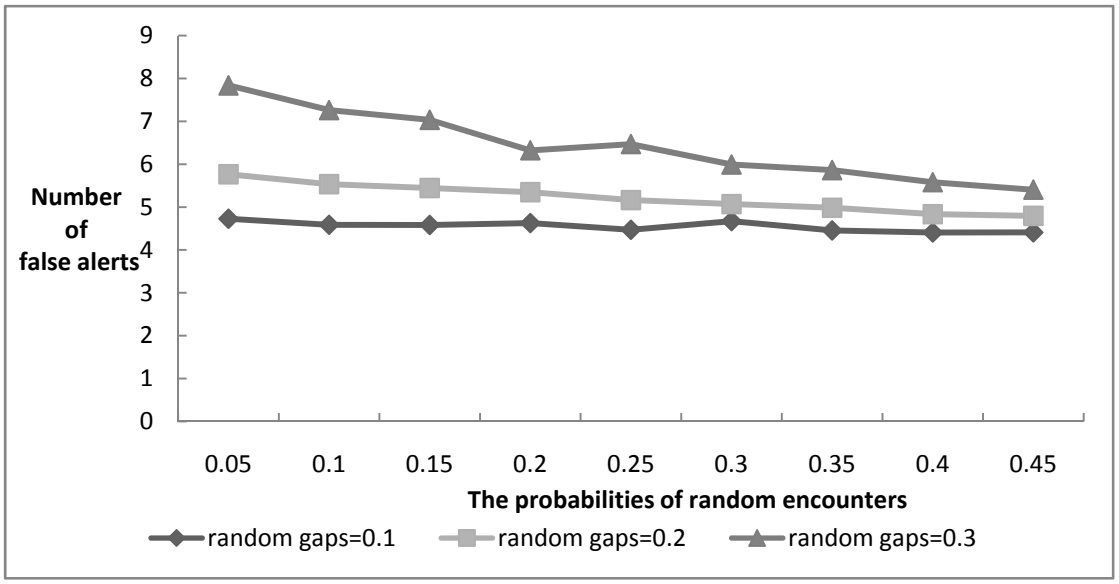


Figure 22 Experiment 4.1.2

Empirical results explanation:

The above plot (Figure 22) demonstrates the interesting phenomenon in which the *random encounters* noises have the positive effect on the accuracy of the *DPFLA*. For a fixed probability of *random gaps,* the growth in the probabilities of *random encounters* will make the number of false alerts go down. This trend is particularly obvious when the probability of *random gaps* was set to 0.3. That was because a larger value of the probability of *random gaps* will amplify the effect of the *random encounters* in the *noises collision* equation $P_{random-gap} * P_{random-encounter}$.

The empirical results of Experiment 4.1.1 and Experiment 4.1.2 basically lend credence to our

previous predictions—*the random gaps noises affect the accuracy of the DPFLA negatively whereas the random encounters will only compensate for the random gaps to some extent.*

The following two experiments (Experiment 4.1.3 and Experiment 4.1.4) will further support the conclusions drawn from the above two experiments.

Experiment 4.1.3:

Experimental condition:

The objective of this experiment is to test the performance of *DPFLA* when working in an environment containing only *random gaps* noises. The experiment will demonstrate that how the number of false alerts will change as the probabilities of *random gaps* varying from 0.05 to 0.45 in ascending order with the incremental step 0.05. We still have $|\mathbb{S}_{ts}| = |\mathbb{S}_{nas}| = 4$ in this experiment. The empirical results can be shown as the following figure (Figure 23)

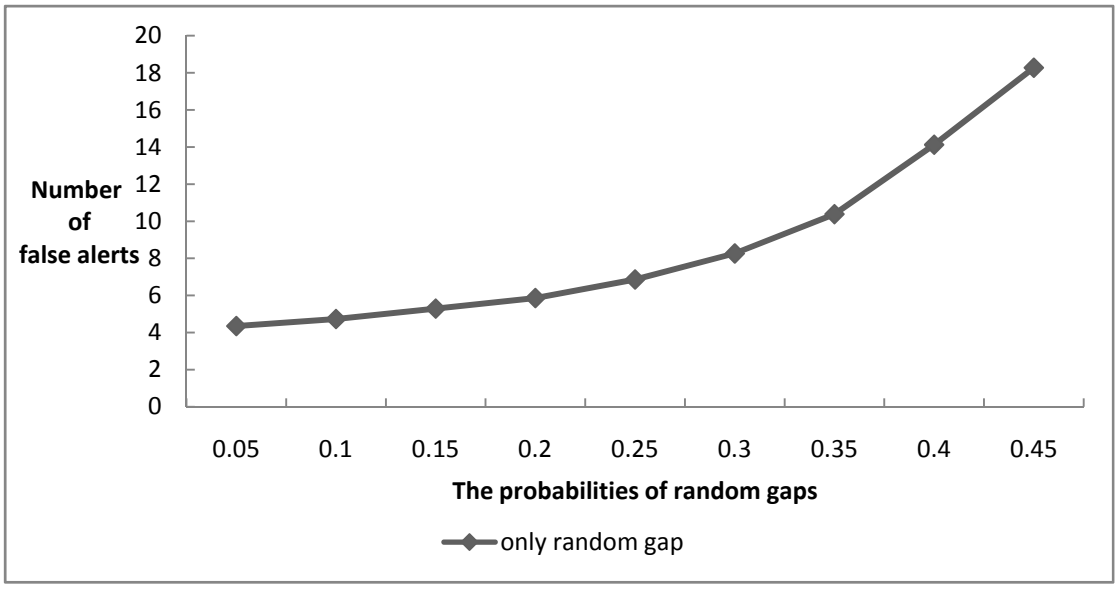


Figure 23 Experiment 4.1.3

Empirical results explanation:

Not surprisingly, the *random gaps* noises will trigger the number of false alerts to go up significantly. Please note that the ending point of the curve in Figure 23 has a greater value than any one of the three curves in Figure 21 since there is no *random encounter* at all at this time.

Experiment 4.1.4:

Experimental condition:

The goal of this experiment is to test the performance of *DPFLA* when working in a pure *random encounters* environment. That means, unlike the experiments we have conducted before, we

don't assume the existence of the *daily pattern* in this experiment. This experiment will demonstrate that how the number of alerts will change as the probabilities of *random encounters* varying from 0.05 to 0.45 in ascending order with the incremental step 0.05. We still have $|\mathbb{S}_{ts}| = |\mathbb{S}_{nas}| = 4$ in this experiment. The empirical results can be shown as the following figure (Figure 24)

Please note that the concept "false alert" is not applicable to this experiment. The "false alert" is a term which can only apply to the environment which incorporates at least one periodic pattern. In this experiment, all the meeting days are supposed to be alerted since they were occurring purely by chance (not conform to any existing periodic patterns).

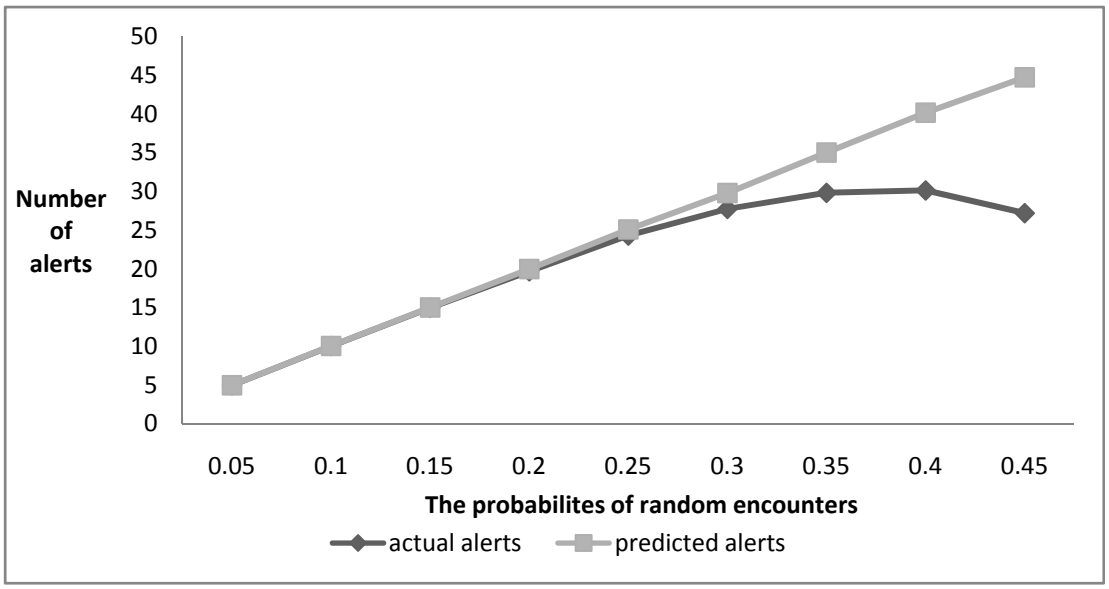


Figure 24 Experiment 4.1.4

Empirical results explanation:

It is necessary to distinguish two terms here—*predicted alerts* and *actual alerts*. The *predicted alerts* refer to those alerts which were supposed to be there. Since we simulated for a 100 days' period, the *predicted alerts* in this experiment should be $100 * P_{random-encounter}$. However, the *actual alerts* are not as many as the *predicted alerts* from the above figure (Figure 24). The underlying reason is that even the *random encounters* can lead the state transition of the *DPFLA* to the accepting state S_a . The probability of such a *wrongly accepting case* would be:

$$P_{wrongly-accepting} = P_{random-encounter}^{|\mathbb{S}_{ts}|} \quad (4.1)$$

In a *wrongly accepting case*, the remaining meeting days are to be considered as reconfirmations of the *daily pattern* until a certain number of consecutive gap days make the learning process jump out of the noises accommodation states space. That's exactly why the number of *actual alerts* is less than the *predicted alerts.* According to the empirical results, it is demonstrated that

the *random encounters* with the probability larger than 0.25 will obviously present such a phenomenon.

Then, after considering how the two types of random noises can affect the performance of the *DPFLA*, we would like to present some empirical results regarding the sizes of the states spaces in the *DPFLA*.

Experiment 4.1.5

Experimental condition:

The objective of this experiment is to test the performance of *DPFLA* when expanding both the trial states spaces and the noises accommodation states space in the same manner. $|\mathbb{S}_{ts}|$ and $|\mathbb{S}_{nas}|$ will be changed simultaneously in ascending order of integer numbers within the interval [1,10]. As learned from the earlier experiments, only *random gaps* should be considered as the true noises in the simulations of the *daily pattern.* In view of this, we fixed different probabilities of *random gaps* to test how the number of false alerts will be changed as both $|\mathbb{S}_{ts}|$ and $|\mathbb{S}_{nas}|$ being expanded simultaneously in the same manner. No *random encounters* noises were added in this experiment. The empirical results can be shown as the following figure (Figure 25)

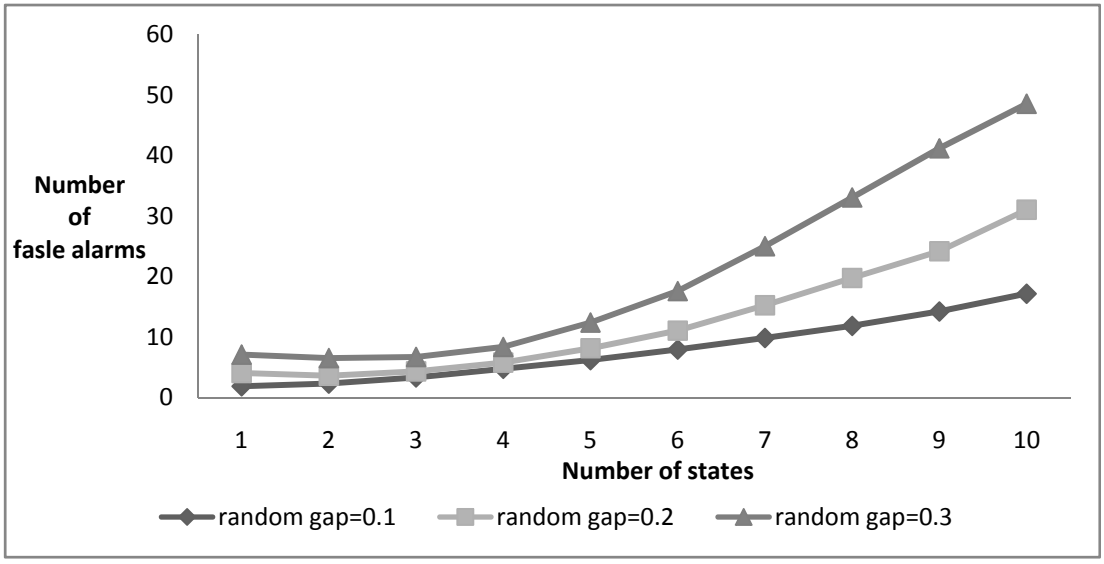


Figure 25 Experiment 4.1.5

Empirical result explanation:

Expanding both the trial states space and the noises accommodation states space is still vulnerable to the *random gaps* noises. The number of false alerts will increase drastically as the

state spaces being expanded in a linear manner. Definitely, the negative effect of the *random gaps* noises can be reflected in this experiment, too. That means, a greater probability of the *random gaps* noises will lead to significantly more false alerts. It seems that though we expand the noise accommodation states space at the same time, the number of false alerts will still climb up. This phenomenon raised a question, is the accuracy of the *DPFLA* mostly determined by the size of trial states space $\left|\mathbb{S}_{ts}\right|$? The next experiment (Experiment 4.1.6) will explore this question.

Considering the coordinates in the *y* axis, we know that expanding both the two states spaces simultaneously in the same manner would even worsen the accuracy of the *DPFLA.*

Experiment 4.1.6

Experimental condition:

The objective of this experiment is to compare the performance of *DPFLA* under two different conditions, one is to expand the size of both the two state spaces $\left|\mathbb{S}_{ts}\right|$ and $\left|\mathbb{S}_{nas}\right|$ simultaneously in the same manner as what we have done in the Experiment 4.1.5; the other is to expand only the size of the trial states space $\left|\mathbb{S}_{ts}\right|$. The expanding operations will be done in ascending order of integer numbers within the interval [1, 10]. In this experiment, we fixed the *random gaps* probability to 0.2. We also fixed $\left|\mathbb{S}_{nas}\right| = 4$ while only expanding the $\left|\mathbb{S}_{ts}\right|$. No *random encounters* noises were added in this experiment. The empirical results can be shown as the following figure (Figure 26)

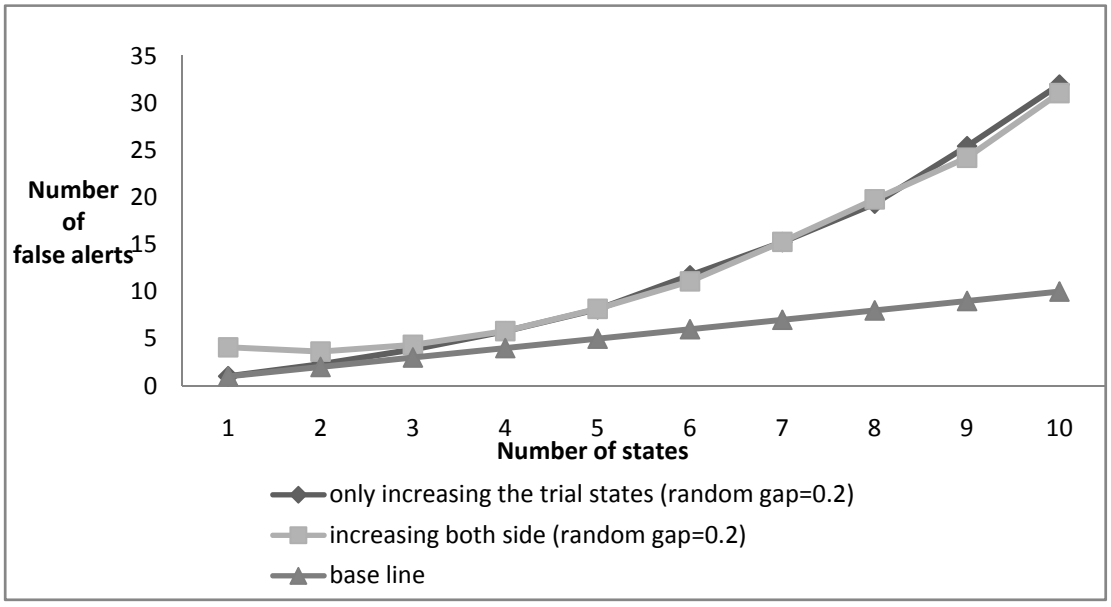


Figure 26 Experiment 4.1.6

Empirical results explanation:

From the above figure (Figure 26), it is very hard to distinguish between the curve with only the trial states spaces expanded and the curve with both the trial states space and the noises accommodation states spaces expanded. It seems this demonstration gives a positive answer to the question in the Experiment 4.1.5. However, this phenomenon might be *a false appearance* since the size of the noises accommodation states spaces ($|\mathbb{S}_{nas}| = 4$) might be sufficient to accommodate the *random gaps* noises under the condition that the probability of *random gaps* is 0.2. That's probably why it seems the size of the trial states space will "mostly" affect the performance.

Up to the moment we are still not clear about to what extent expanding the noises accommodation states space will affect the accuracy of the *DPFLA*. This requires a further experiment (Experiment 4.1.7) for the purpose of demonstration.

Please note that the *base line* shows that the lower bound (the minimum number) of the false alerts will increase as the size of the trial states space $|\mathbb{S}_{ts}|$ increasing. Moreover, this can be represented as a *linear* relationship. However, the actual number of false alerts is a *non-linear* relationship with the size of the trial states space $|\mathbb{S}_{ts}|$. The underlying reason is that there are much more complicated scenario-based probabilistic relationships between $|\mathbb{S}_{ts}|$ and the number of false alerts. Each time when the transition starts from the first state, it has the probability $P_{\text{int}ermediat-\text{int}erupption}$ that the learning process would go back directly to the first state again. This situation can be described as the following equation:

$$P_{\text{int}ermediate-\text{int}erruption} = 1 - P_{best-case} = 1 - (1 - P_{gap})^{|\mathbb{S}_{ts}|} \quad (4.2)$$

It can be figured out that as the size of the trial states space $|\mathbb{S}_{ts}|$ increasing, then the $P_{\text{int}ermediat-\text{int}erupption}$ will become greater. In such a case, the *DPFLA* is getting more opportunities to produce false alerts.

Experiment 4.1.7

Experimental condition:

The objective of this experiment is to test the performance of DPFLA while only expanding the size of the noise accommodation states space $|\mathbb{S}_{nas}|$. The expanding operation will be done in

ascending order of integer numbers within the interval [1, 10]. We fixed the random gaps probability to 0.2 and $|\mathbb{S}_{ts}| = 4$. No *random encounters* noises were added in this experiment. The empirical results can be shown as the following figure (Figure 27)

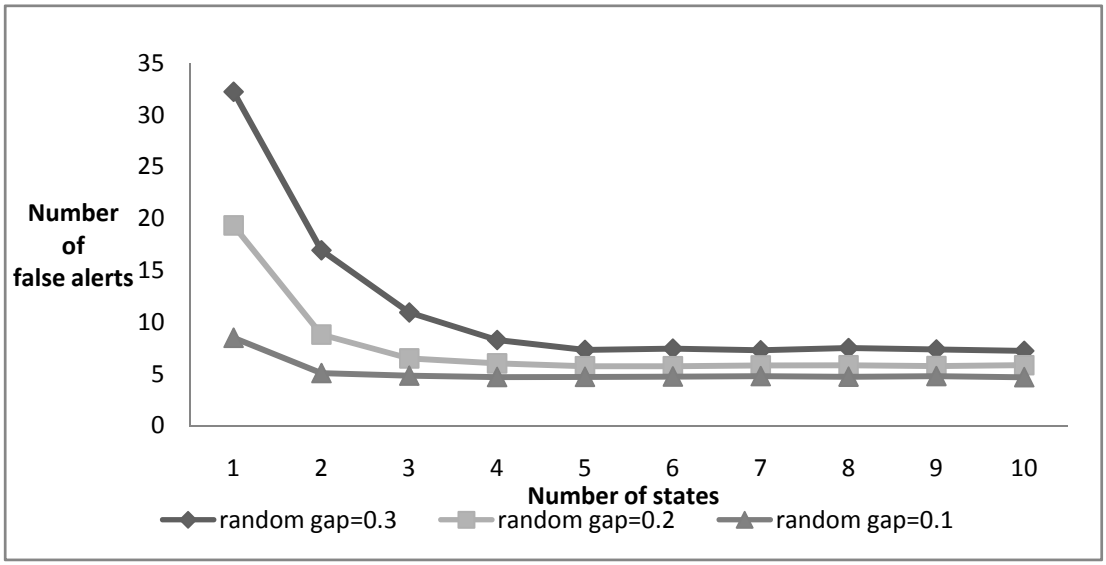


Figure 27 Experiment 4.1.7

Empirical results explanation:

It is obvious that expanding the noise accommodation states space will decrease the number of false alerts at the first several stages. After that, all the three curves will become nearly horizontal. We call this phenomenon as "*sub-optimal convergence*". However, the above empirical results demonstrate that the three curves with different *random gaps* probabilities will converge to different sub-optimal degrees of accuracy. The sub-optimal accuracy the *DPFLA* can finally reach is basically scenario-dependent. In this experiment, it is empirically demonstrated that different probabilities of *random gaps* may lead to different sub-optimal accuracies. The next experiment (Experiment 4.1.8) will show that not only the probabilities of *random gaps* will affect the sub-optimality, but also the size of the trial states space $|\mathbb{S}_{ts}|$.

Experiment 4.1.8

Experimental condition:

The objective of this experiment is to test whether the size of the trial states space $|\mathbb{S}_{ts}|$ can affect the sub-optimal accuracies as expanding the noises accommodation states space. For the sake of comparison, we consider two cases in this experiment: in one case, $|\mathbb{S}_{ts}| = 5$; in the other case, $|\mathbb{S}_{ts}| = 4$. The expanding operation on the noises accommodation states space

will be done in ascending order of integer numbers within the interval [1, 10]. We fixed the probability of *random gaps* to 0.2. No *random encounters* noises were added in this experiment. The empirical results can be shown as the following figure (Figure 28)

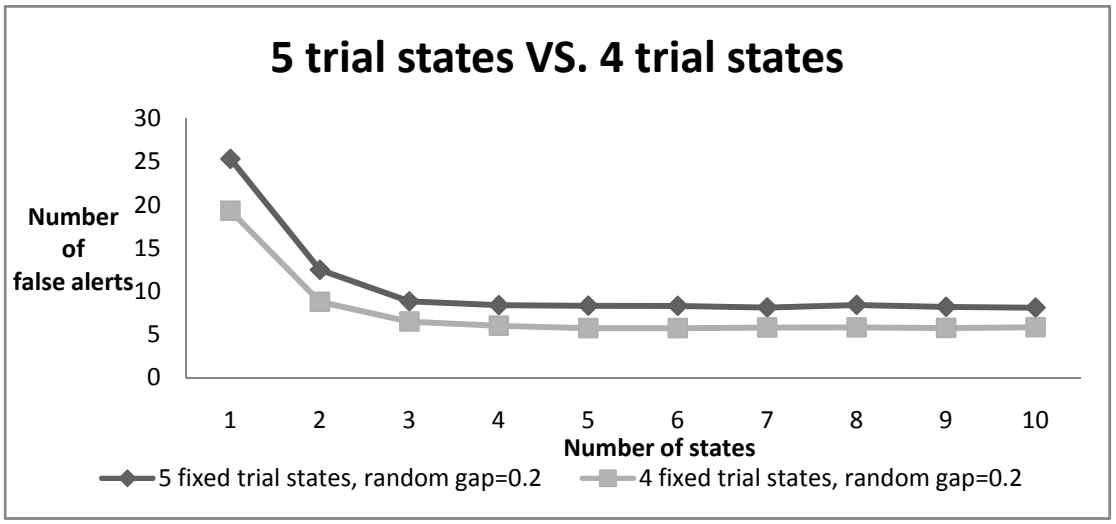


Figure 28 Experiment 4.1.8

Empirical results explanation:

Combining the empirical results from Experiment 4.1.7 and Experiment 4.1.8, it can be figured out that both the probabilities of *random gaps* and the size of the trial states space $|\mathbb{S}_{ts}|$ will affect the sub-optimality of the *DPFLA*. Generally speaking, if one or both of these two variables increase, then the sub-optimal accuracy of the *DPFLA* will be reduced (there will be more false alerts in case of *sub-optimal convergence*).

Please note that the above discussion of *sub-optimal convergence* was purely based on the empirical results. The convergence to a sub-optimal accuracy in the *DPFLA* needs more sophisticated mathematical formulations. However, due to the scope of the Master thesis, we leave this to the future work.

4.2. Alternative DPFLA

The basic simulation environment for the *alternative DPFLA* is the same as the *DPFLA* (a sequence of consecutive meeting days), unless there is a special declaration (Experiment 4.2.4). The *random gaps* noises and *random encounters* noise*s* will be added based on the specific condition of each experiment.

In this section, since many of the experiments were done in the same manner as the corresponding ones in the previous section, we are not going to discuss all of them in detail (some of them might be discussed together). We are going to place the emphasis of the discussion in this section on the comparisons of empirical results between the *DPFLA* and

alternative DPFLA.

Experiment 4.2.1, Experiment 4.2.2 were done in the same manner as the Experiment 4.1.1 and Experiment 4.1.2 respectively in the previous section. The only difference in the experimental conditions lies in that we have to set up the penalty rate to a fixed value ($pr = 3$) in the simulations of the *alternative DPFLA*.

The empirical results can be shown as the following figures (Figure 29 and Figure 30)

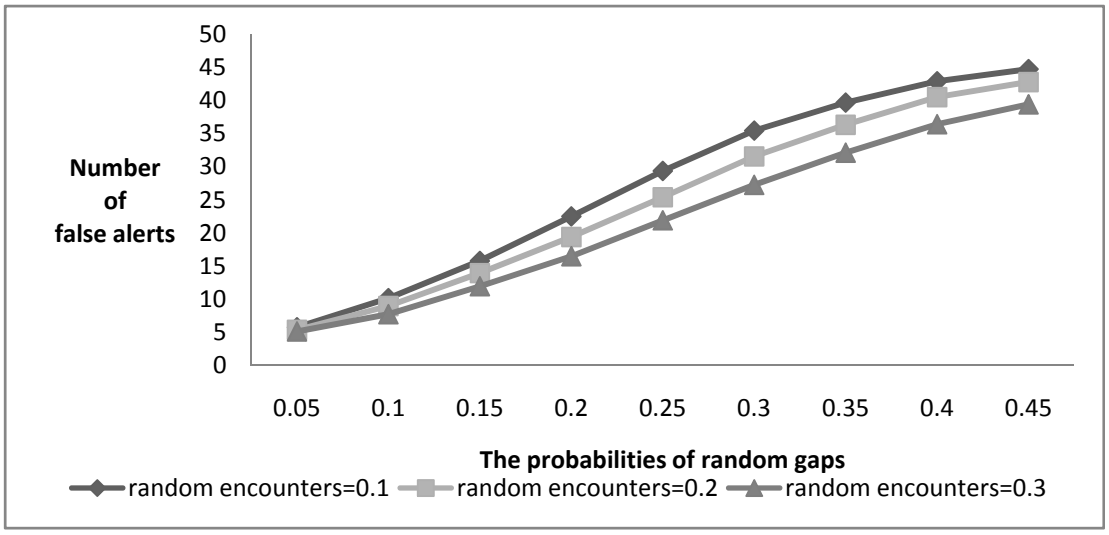


Figure 29 Experiment 4.2.1

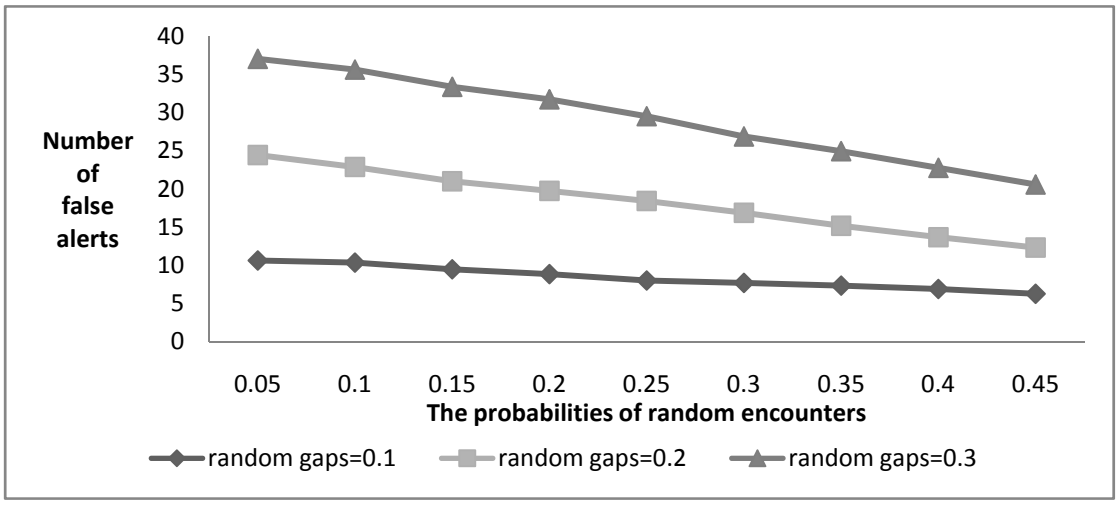


Figure 30 Experiment 4.2.2

The empirical results demonstrate the same principle which we have discussed in the previous section. For the *alternative DPFLA*, we still have the conclusion that *the random gaps noises affect the accuracy of the DPFLA negatively whereas the random encounters will only compensate for the random gaps to some extent.*

Experiment 4.2.3 and Experiment 4.2.4 were done in the same manner as the Experiment 4.1.3

and Experiment 4.1.4 respectively in the previous section. Similarly, we have to set up the penalty rate to a fixed value ($pr = 3$) in the simulations of the *alternative DPFLA*.

We would like to present two comparison-based demonstrations of the empirical results between Experiment 4.2.3 (Experiment 4.2.4) in this section and Experiment 4.1.3 (Experiment 4.1.4) in the previous section. The following two figures will show these two comparisons (Figure 31 and Figure 32):

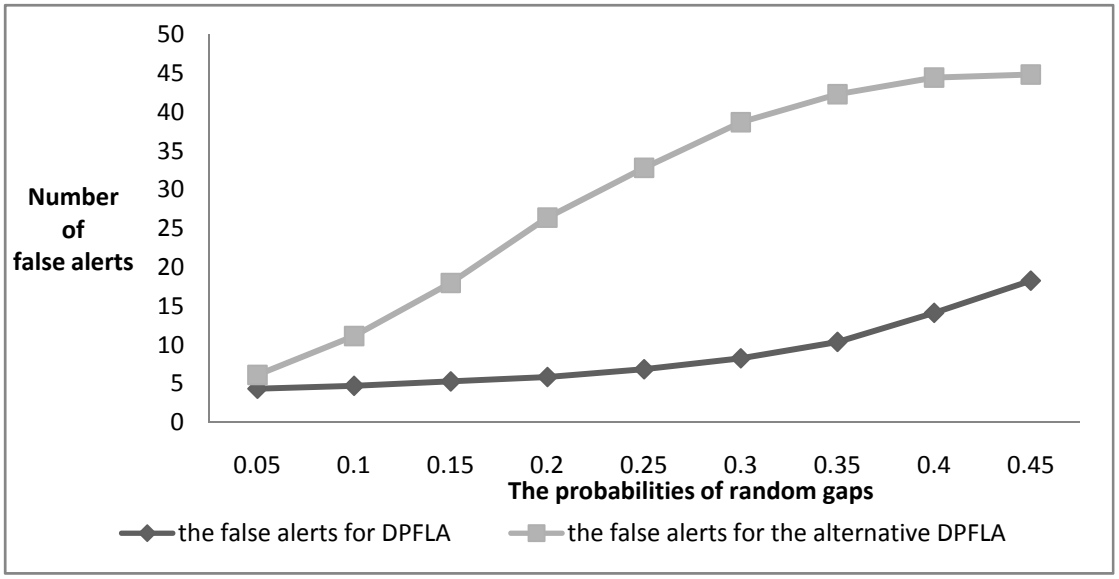


Figure 31 Experiment 4.2.3 V.S. Experiment 4.1.3

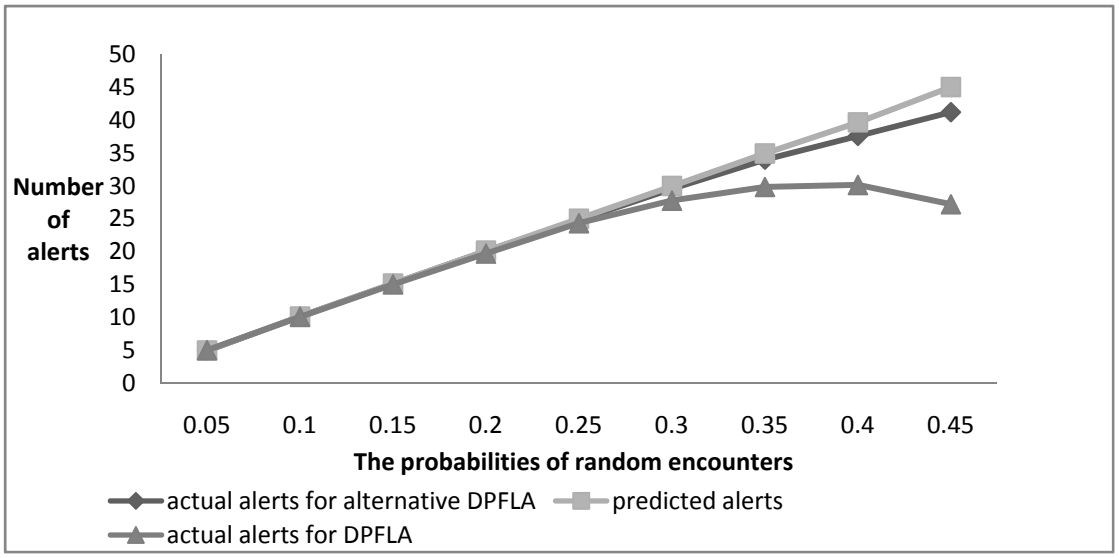


Figure 32 Experiment 4.2.4 V.S. Experiment 4.1.4

From the Figure 31, it is obvious that the *alternative DPFLA* is much more vulnerable to the *random gaps* noises, compared with the *DPFLA*. The underlying reason is that the weightings of penalties and rewards are very different in the noises accommodation states spaces of these two kinds of *FLA*s.

Each coin has two sides. Though *alternative DPFLA* is vulnerable to the *random gaps* noises, empirical results shows it performs significantly better in a pure random encounters' environment. From the Figure 32, it can be figured out that the curve of actual alerts for *DPFLA* diverges more sharply from the curve of predicted alerts than the curve of actual alerts for *alternative DPFLA.* This phenomenon is particularly obvious in the tail ends of these two curves. It seems the pure random meeting days are much more difficult to make the *alternative DPFLA wrongly accept* them as parts of the *daily pattern* since the penalties on the gap days have significantly heavier weights than the rewards on the meeting days in the *alternative DPFLA* structure.

Experiment 4.2.5 and Experiment 4.2.6 were done in the same manner as the Experiment 4.1.5 and Experiment 4.1.6 respectively in the previous section. The only difference in the experimental conditions is that we still have to fix the penalty rate ($pr = 3$) while expanding the states spaces. Also, please note that the expanding operation will be done in ascending order of integer numbers within [4, 13] since a states space with less than 4 states would be meaningless in case of the penalty rate being set to 3 (states). Moreover, in Experiment 4.2.6, the size of the noises accommodation states space $\left|\mathbb{S}_{nas}\right|$ was fixed to 4 while only expanding the trial states space. The empirical results of Experiment 4.2.5 and Experiment 4.2.6 can be found in the following figures (Figure 33 and Figure 34) respectively.

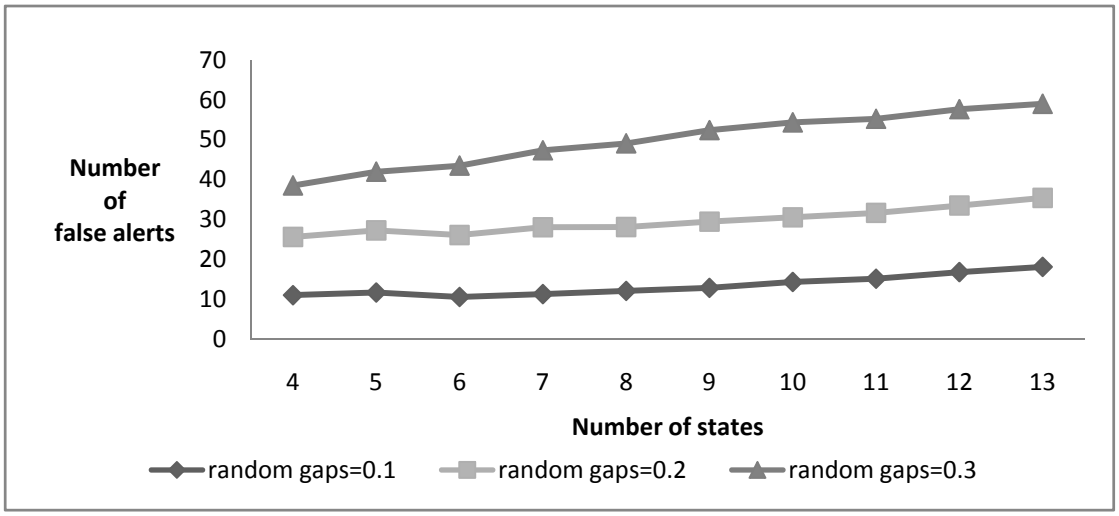


Figure 33 Experiment 4.2.5

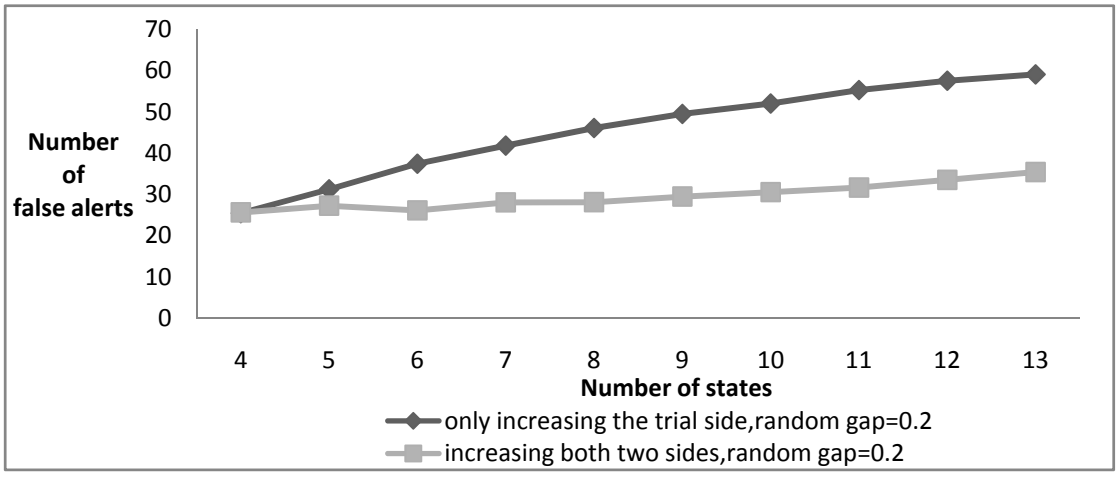


Figure 34 Experiment 4.2.6

In Figure 33, the number of false alerts will climb up slightly when expanding both the trial states space and the noises accommodation states space simultaneously in the same manner. Please note that the growth rate of each curve in Figure 33 is obviously lower than that in Figure 25. The underlying reason is that expanding the noises accommodation states space would limit the growth of the number of false alerts to some extent.

The above discussion can be further supported in Figure 34. The growth rate of the number of false alerts is significantly higher when only expanding the trial states space. This demonstration empirically confirms that the growth of the size of the noise accommodation states space will efficiently help to reduce the number of false alerts.

The next experiment (Experiment 4.2.7) aims to investigate how the growth of the size of the noises states space $\left|\mathbb{S}_{nas}\right|$ can reduce the number of false alerts.

Experiment 4.2.7

Experimental condition

The objective of this experiment is to test the performance of the *alternative DPFLA* while only expanding the noise accommodation states space, under the three conditions that the probabilities of *random gaps* are 0.1, 0.2 and 0.3, respectively. The expanding operations will be done in ascending order of integer numbers within [4, 13] since the penalty rate was set to 3 ($pr = 3$). We also fixed the size of the trial states space to 4 ($\left|S_{ts}\right| = 4$). No *random encounters* will be added in this experiment. The empirical results can be shown as the following figure (Figure 35)

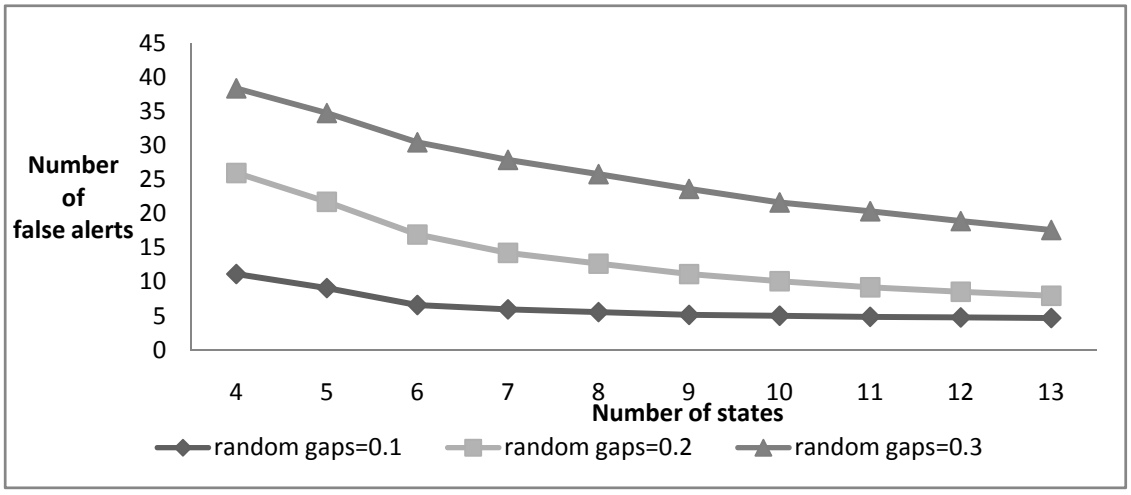


Figure 35 Experiment 4.2.7

If the expanding operations were done in an exponential growth manner ($4^x \quad x \in [1,10]$), then we can obtain the empirical results as the following figure (Figure 36)

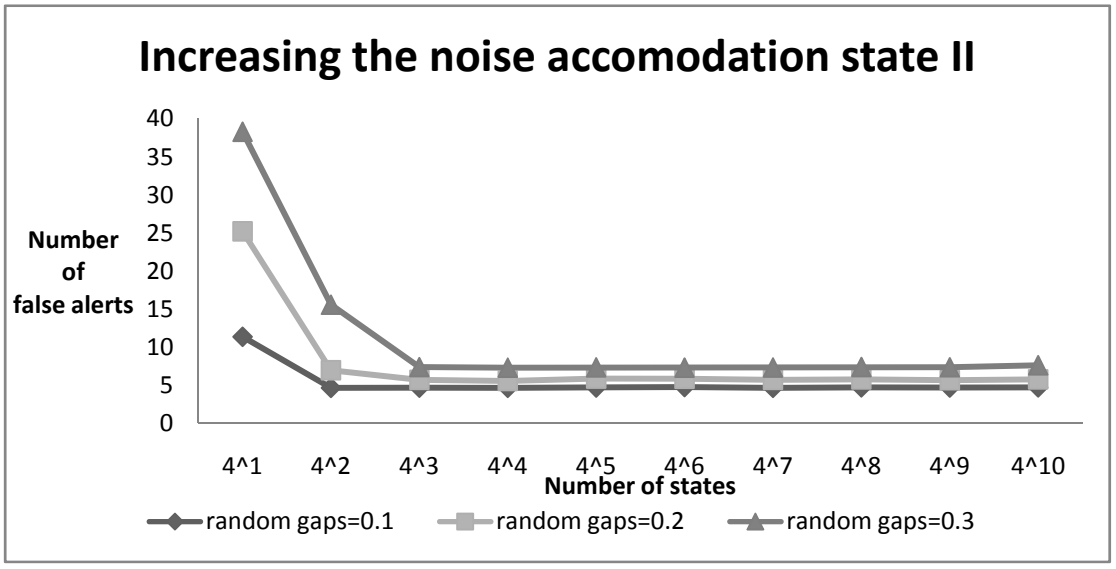


Figure 36 Experiment 4.2.7 (expanding exponentially)

Empirical results explanation:

At this point, the readers may feel confused about why we changed the expanding operations from a linear growth manner to an exponential growth manner. In fact, the descending trend of the number of false alerts is already very obvious in the Figure 35. However, these empirical results didn't tell us anything about whether this descending trend will lead to a sub-optimal accuracy since even the tail ends of the three curves in the Figure 35 are still not stable.

In view of this, we decided to expand the noise accommodation states space in a much faster way. Then in Figure 36, the extended experiment tells that the *alternative DPFLA* would converge to different sub-optimal accuracies under different probabilities of the *random gaps*. It is empirically

demonstrated in Figure 36 that all the curves turn to be nearly horizontal if the size of the noise accommodation states space can be sufficiently large. However, please note that the above discussion was merely based on the empirical results. The mathematical proof is still needed to be formulated in this regard.

In all of the above experiments, the penalty rate was set to a fixed value 3 ($pr = 3$). Now, it is interesting to see what will happen if the penalty rate can be adjusted.

Experiment 4.2.8

Experimental condition:

The objective of this experiment is to test the performance of the *alternative DPFLA* in case of adjusting the penalty rate pr . The adjusting operation will be done in ascending order of integer numbers with [1, 9]. In view of this, we set both the sizes of the trial states space and the noises accommodation states space to the same fixed value 10 ($\left|\mathbb{S}_{ts}\right| = \left|\mathbb{S}_{nas}\right| = 10$). Also, we fixed the probabilities of both the *random gaps* and the random encounters to the same value 0.2 in order to build a noisy environment. The empirical results can be shown as the following figure (Figure 37)

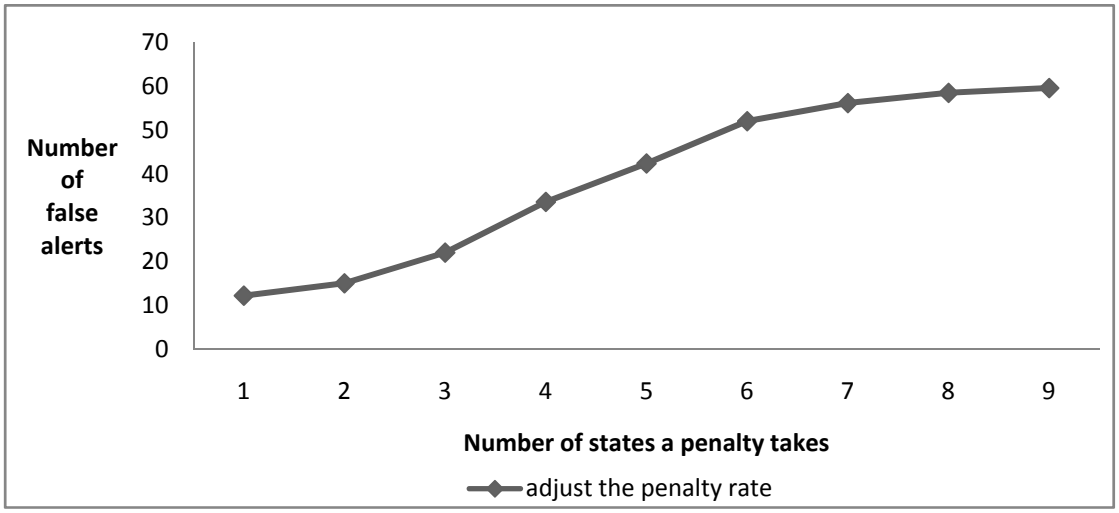


Figure 37 Experiment 4.2.8

Empirical results explanation:

Since we have agreed that the only effect of the *random encounters* is to compensate for a small portion of the gap days caused by the *random gaps* (*noises collision*). Consequently, the same trend in the above figure (Figure 37) can be applicable to the case that there is no *random encounter* at all.

The empirical results demonstrate that the *alternative DPFLA* will become more vulnerable to the

random gaps noises as the penalty rate increases. This is straightforward since a greater penalty rate will make it easier for the learning process to jump out of the noises accommodation states space (It is easier for the learned *daily pattern* to be destroyed).

4.3. Perceptual Aliasing Problem

Many *reinforcement learning* tasks would be faced with the "*Perceptual Aliasing*" [16] problem. The *perceptual aliasing* problem refers to the fact that *"different states may have the same representation for the agent, and as a consequence the same expected costs will be assigned to them, even when this should not be the case."* [16]

In our case, the *perceptual aliasing* problem can be stated as the follows:

After the daily pattern was learned, when processing a sequence consists of both the meeting days and the gap days, How does our FLA know this is still the daily pattern with some random gaps noises, or this is a sign of the occurrence of another pattern?

In order to investigate this problem, we designed two experiments: one is for the *DPFLA;* the other is for the *alternative DPFLA.* The common assumption of these two experiments is that the *daily pattern* has already been completely learned.

Experiment 4.3.1

Experimental condition:

The objective of this experiment is to test the performance of *DPFLA* after the *daily pattern* was learned. As we usually did before, $\left|\mathbb{S}_{ts}\right| = \left|\mathbb{S}_{nas}\right| = 4$. In order to keep the same representations as other patters from "10" (one day one, one day off), "110" (two days on, one day off), … , to "1111111110" (nine days on, one day off), the probabilities of *random gaps* can only get values from an ordered set {0.1, 0.111, 0.125, 0.143, 0.167, 0.2, 0.25, 0.333, 0.5} one by one. This experiment will demonstrate if the *DPFLA* can distinguish whether the current situation is the occurrence of a new pattern or still the *daily pattern* with some *random gaps* noises. The empirical results can be shown as the following figure (Figure 38)

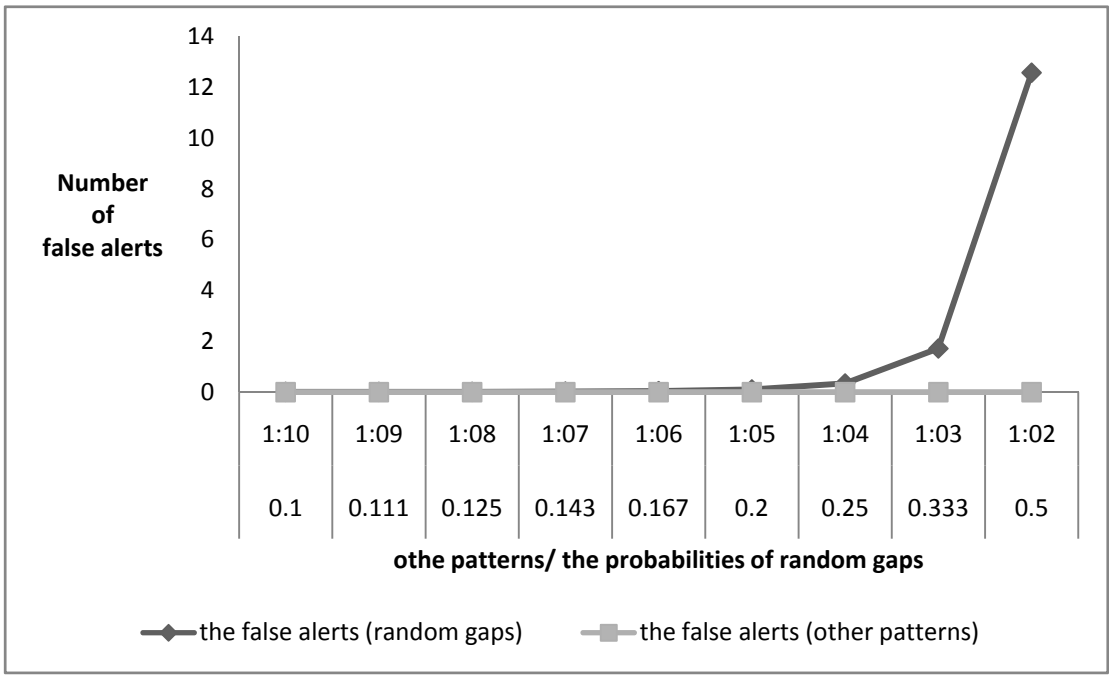


Figure 38 Experiment 4.3.1

Empirical results explanation:

The horizontal line overlapping with x axis in the above figure (Figure 38) demonstrates that the *DPFLA* can't destroy the learned *daily pattern* in case other patterns (specified in the *experimental condition*) are occurring.

However, this is not the case for the *random gaps* noises. When the probability of the *random gaps* noises exceeds approximately 0.2, the number of false alerts will go up drastically. That means, if the probability of the *random gap* noises can be sufficiently large, then the learned *daily pattern* would be destroyed.

Covering up other patterns but vulnerable to the *random gap* noises, obviously this is not desirable for the real-world application.

Experiment 4.3.2

Experimental condition:

The objective of this experiment is to test the performance of the *alternative DPFLA* after the *daily pattern* was learned. As we usually did before, $\left|\mathbb{S}_{ts}\right| = \left|\mathbb{S}_{nas}\right| = 4$ and $pr = 3$. In order to keep the same representations as other patters from "10" (one day one, one day off), "110" (two days on, one day off), … , to "1111111110" (nine days on, one day off), the probabilities of *random gaps* can only get values from an ordered set {0.1, 0.111, 0.125, 0.143, 0.167, 0.2, 0.25, 0.333, 0.5} one by one. The experiment will test if the *alternative DPFLA* can

distinguish whether the current situation is the occurrence of a new pattern or still the *daily pattern* with some *random gaps* noises. The empirical results can be shown as the following figure (Figure 39)

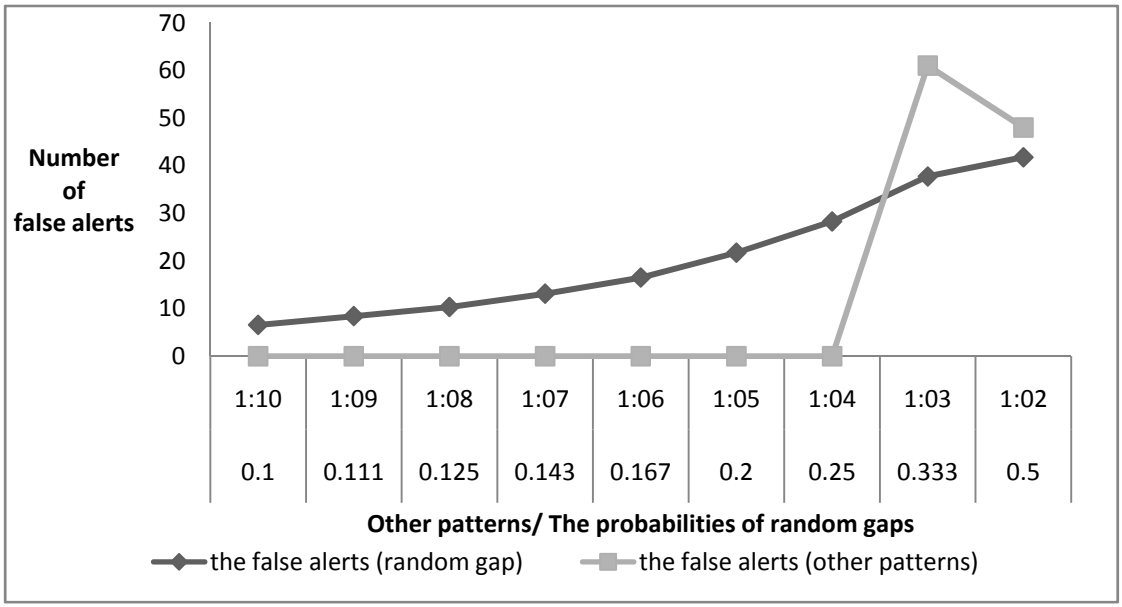


Figure 39 Experiment 4.3.2

Empirical results explanation:

The above empirical results demonstrate that the *alternative DPFLA* can distinguish some other patterns by destroying the current *daily pattern*. Please note that in our case we have $|\mathbb{S}_{nas}| = 4$ and $pr = 3$. That means, the patterns "110" (two days on, one day off), "10" (one day on, one day off) will make the *alternative DPFLA* jump out of the noises accommodation states space. The underlying reason is that the rewards triggered by the meeting days were not able to compensate for the penalties triggered by the gap days.

However, though some other patterns could make the *alternative DPFLA* destroy the outdated *daily pattern,* the *alternative DPFLA* is generally even more vulnerable to the *random gaps* noises, compared with the *DPFLA.*

Now, we can bring out a new challenge based on all of the above empirical results—*It is difficult to find a pure online solution which can deal with the pattern shifting situation as soon as possible while still keeping robust to the random gaps noises.*

4.4. WDPFLA and SDPFLA

Since both the *WDPFLA* and *SDPFLA* can be generated from the same generic *FLA* template as discussed in the previous chapter, here we put them together for the purpose of analysis in order to avoid redundant discussion. In this section, we will only present the empirical results of the *WDFLA.* Since our extensive empirical results bear the fact that the same trend will also be found

for the *SDFLA*, we will not include the empirical results of *SDFLA* in order to avoid prolixity.

The basic simulation environment for the *WDFLA* and *SDFLA* should be a sequence consists of the repeated occurrences of the corresponding periodic pattern (the *weekday pattern* or the *second day pattern*). The *random gaps* noises and *random encounters* noises can be added based on the specific condition of each experiment.

Experiment 4.4.1

Experimental condition:

The objective of this experiment is to test the performance of *WDPFLA* when working in an environment containing both the *random gaps* noises and *random encounters* noises. The experiment will demonstrate that how the number of false alerts will change as the probabilities of *random gaps* varying from 0.05 to 0.45 in ascending order with the incremental step 0.05, under three different but fixed probabilities (0.1, 0.2 and 0.3) of the *random encounters.* Here, $\left|\mathbb{S}_{ts}\right| = 5$ and $\left|\mathbb{S}_{ngs}\right| = 2$ (for the *weekday pattern*). The empirical results can be shown as the following figure (Figure 40)

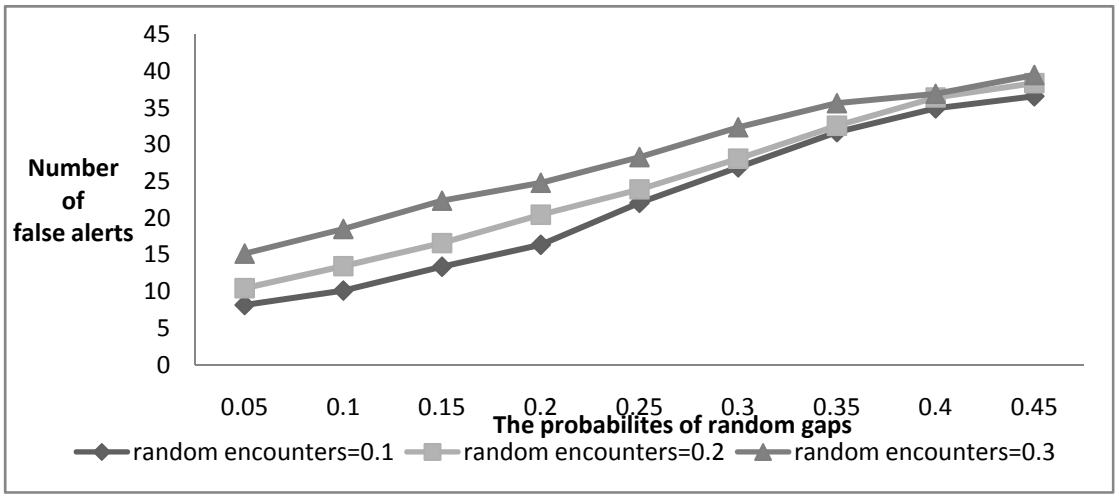


Figure 40 Experiment 4.4.1

Empirical results explanation:

All the three curves in the above figure (Figure 40) demonstrate that the *random gaps* noises have negative effect on the accuracy of the *WDPFLA*. As the probability of the *random gaps* increases, the number of false alerts will go up accordingly. However, if we put the three curves together for comparison, it can be figured out that the number of false alerts under a higher *random encounters* probability is generally larger than those under lower *random encounters* probabilities. That implies the *random encounters* may also have negative effect on the accuracy of *WDPFLA*. This situation differs greatly from the similar experiments we have done for the *DPFLA* and alternative *DPFLA* in which the only effect of r*andom encounters* is to compensate for a small portion of the gap days.

In order to further investigate the effect of the *random encounters* in the simulations of *WDPFLA*, the next experiment (Experiment 4.4.2) was conducted for the purpose of collecting more relevant empirical results in this regard.

Experiment 4.4.2

Experimental condition:

The objective of this experiment is to test the performance of *WDPFLA* when working in an environment containing both the *random gaps* noises and *random encounters* noises. The experiment will demonstrate that how the number of false alerts will change as the probabilities of *random encounters* varying from 0.05 to 0.45 in ascending order with the incremental step 0.05, under three different but fixed probabilities (0.1, 0.2 and 0.3) of *random gaps.* Here, $\left|\mathbb{S}_{ts}\right| = 5$ and $\left|\mathbb{S}_{ngs}\right| = 2$ (for the *weekday pattern*). The empirical results can be shown as the following figure (Figure 41)

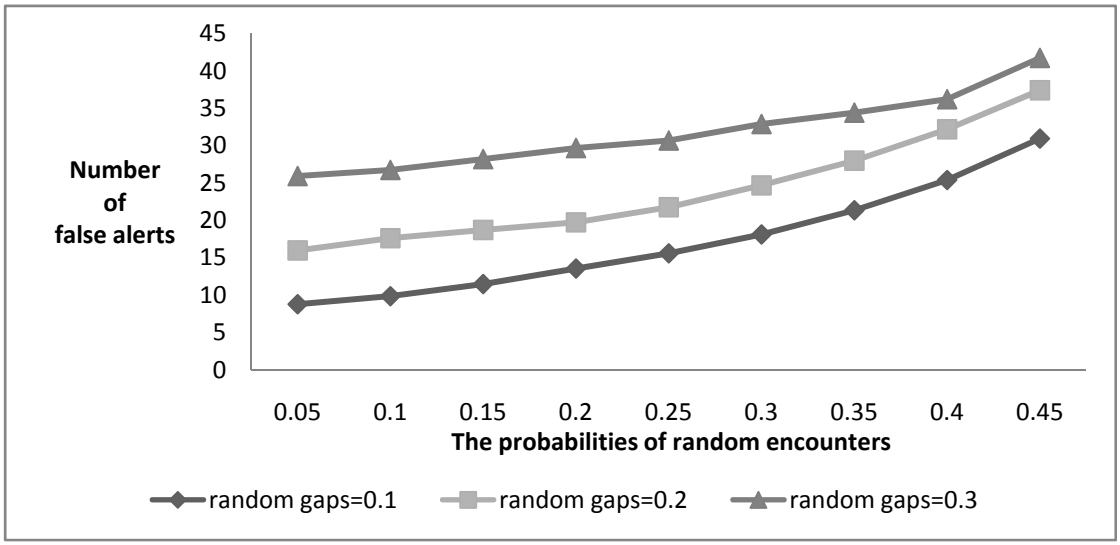


Figure 41 Experiment 4.4.2

Empirical results explanation:

The above figure (Figure 41) demonstrates that the increasing probabilities of *random encounters* will trigger more false alerts. This is different in comparison with the *DPFLA* and *alternative DPFLA*. The underlying reason caused this difference is that the *random encounters* may lead the state transition directly back to the first state if the *random encounters* were occurring in the normal gap days defined by the hypothesis.

At this stage, it is safe to sum up that the *WDPLFA* (and also the *SDPFLA*) are vulnerable to both the *random encounters* noises and *random gaps* noises.

However, up to the moment we are still not clear about to which of the *random gaps* noise and

random encounters noises the *WDPFLA* is more sensitive.

Experiment 4.4.3

Experimental condition:

The objective of this comparison-based experiment is to show which one of the *random gaps* noises and *random encounters* noises has greater effect on the accuracy of the *WDPFLA.* The probabilities of *random gaps noises* and *random encounters noises* are varying from 0.05 to 0.45 in ascending order with the incremental step 0.05, respectively. We still have $\left|\mathbb{S}_{ts}\right| = 5$ and $\left|\mathbb{S}_{ngs}\right| = 2$ (for the *weekday pattern*). The empirical results can be shown as the following figure (Figure 42)

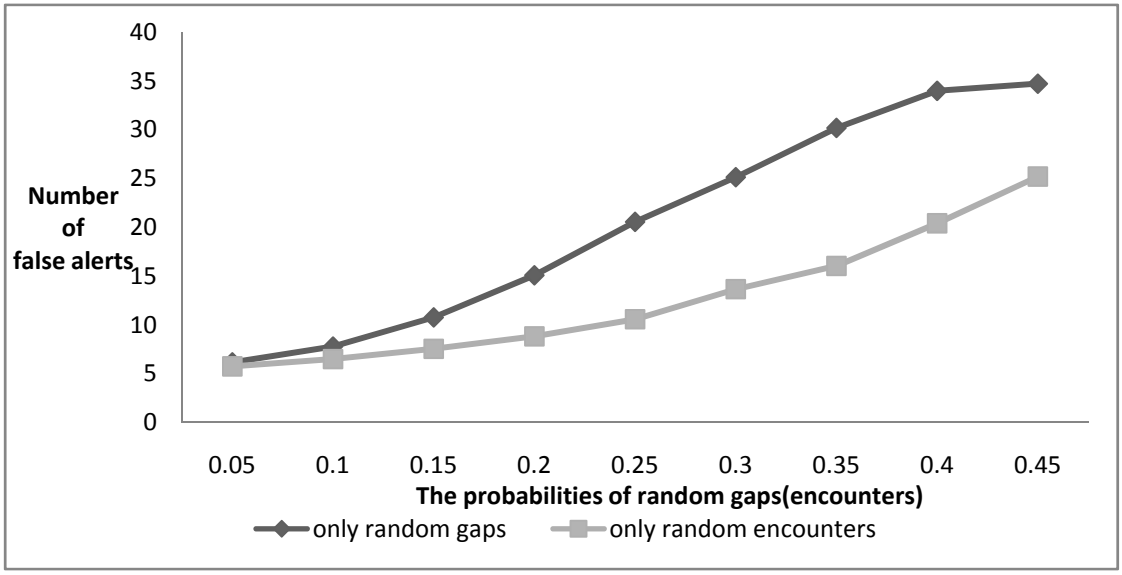


Figure 42 Experiment 4.4.3

Empirical results explanation:

From the above figure (Figure 42), it is obvious that the *random gaps* noises will have greater negative effect on the accuracy of the *WDPFLA* than the *random encounters* noises. The underlying reason is that we have $\left|\mathbb{S}_{ngs}\right| < \left|\mathbb{S}_{ts}\right|$ in the design of *WDPFLA*. That means, even with the same probability, the *random gaps* noises will get more opportunities to lead the state transition directly back to the first state. For the *random encounters* noises, it won't be that easy to make them overlap with the normal gap days in the simulations.

Experiment 4.4.4

Experimental condition:

The objective of this experiment is to test the performance of the *WDPFLA* in case of expanding the noises accommodation states space, under three different noisy environments. The first testing environment was the basic environment with only the *random gaps* noises added with the probability of 0.2. The second testing environment was the basic environment with both the *random gaps* noises and *random encounters* noises added with the same probability of 0.2. The third testing environment was the basic environment with only the *random encounters* noises added with the probability of 0.2. We still have $\left|\mathbb{S}_{ts}\right| = 5$ and $\left|\mathbb{S}_{ngs}\right| = 2$ (for the *weekday pattern*). The empirical results can be shown as the following figure (Figure 43)

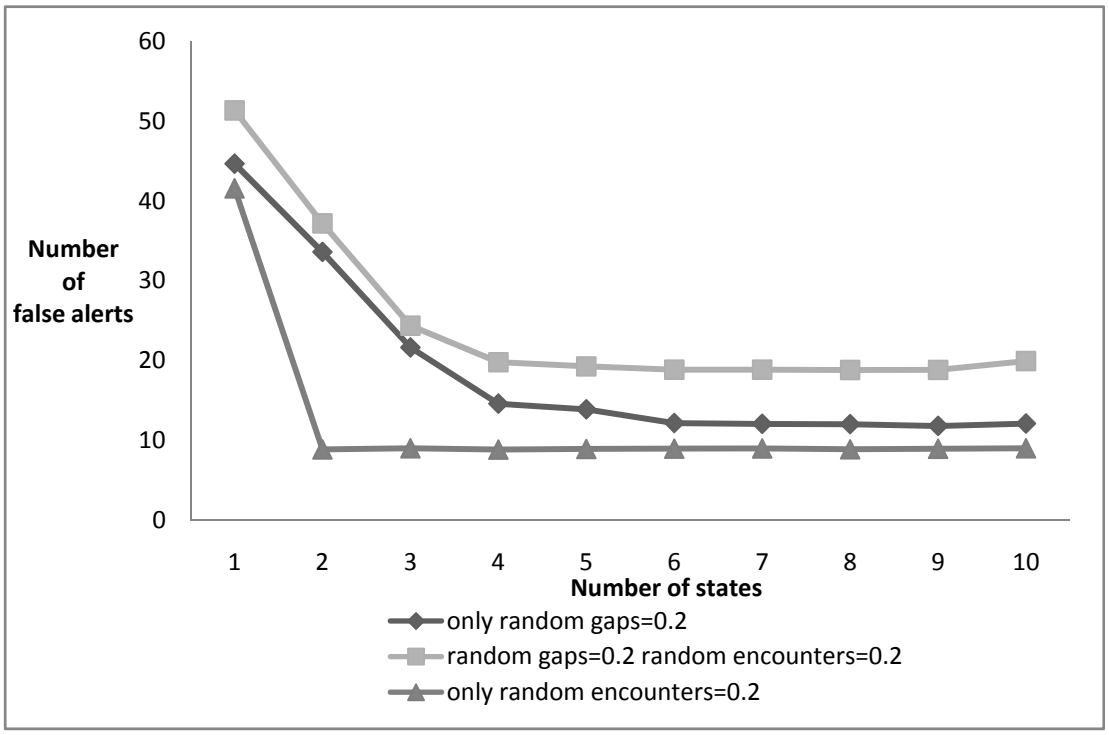


Figure 43 Experiment 4.4.4

Empirical results explanation:

From the above figure (Figure 43), an interesting point is that the size of the noise accommodation states space $\left|\mathbb{S}_{nas}\right|$ should be greater than or equal to the size of the normal gaps states space $\left|\mathbb{S}_{ngs}\right|$. Formally,

$$\left|\mathbb{S}_{nas}\right| >= \left|\mathbb{S}_{ngs}\right| \tag{4.3}$$

This relationship will be particularly obvious in case only the *random encounters* noises were

added. The lowest curve (*only random encounters=0.2*) in the above figure demonstrates that the number of false alerts drops drastically when the $\left|\mathbb{S}_{nas}\right|$ is varying from 1 to 2 ($\left|\mathbb{S}_{ngs}\right|$). The underlying reason is that we need at least $\left|\mathbb{S}_{ngs}\right|$ states to accommodate the normal gap days after the *weekday pattern* was learned, even if there is no *random gap* noise at all. However, it can also be figured out that further increasing the $\left|\mathbb{S}_{nas}\right|$ will not bring further improvements to the accuracy of the *WDPFLA* since the $\mathbb{S}_{nas}$ was originally designed to accommodate only the *random gaps* noises.

The sub-optimality of *WDPFLA* is also scenario-based. That means, the sub-optimal accuracy the *WDPFLA* can reach has to depend on the probabilities of the *random gaps* noises and r*andom encounters* noises. However, please note that the above discussion was merely based on the empirical results. The mathematical proof is still needed to be formulated in this regard.

In addition, the similar law can also be found for the *SDPFLA*.

4.5. Summary

4.5.1. Discussion

Based on the extensive empirical results obtained, it can be summed up that the performances of our *FLA*-based solutions (*DPFLA, Alternative DPFLA, WDPFLA,* and *SDPFLA*) are scenario-dependent.

1) For the *DPFLA* and *Alternative DPFLA,* only the *random gaps* noises can have negative effect on the performance. The *random encounters* noises will compensate for a small portion of gap days so that this kind of noises will do no harm to the learning process of the *daily pattern*. However, this is not the case for the *WDPFLA* and *SDPFLA*. Both the *random gaps* noises and the *random encounters* noises can have negative effects on the performance of *WDPFLA* and *SDPFLA*, though the effect of the former type can be greater.

2) Empirical results demonstrate that it is hard to make pure online solutions (e.g. *DPFLA*, *Alternative DPFLA*) both adaptive to the pattern-shifting situation and robust to the noises. The *perceptual aliasing* problem which is faced by many *reinforcement learning* tasks can also be found in our case. For example, the *Alternative DPFLA* which can be more sensitive to the pattern-shifting situation, however, more vulnerable to the *random gaps* noises.

3) Empirical results demonstrate that expanding the noises accommodation states space will help to resist the negative effects of the random noises in a certain noisy environment, with the trial state spaces fixed. The empirical results also demonstrate that in a certain noisy environment with the trial states space fixed, expanding the noises accommodation states space will make the performance of *FLA* converge to a sub-optimal accuracy. However, this is only the empirical conclusion which still needs further mathematical proofs. This is also the reason why our *FLA*-based solutions can be called scenario-dependent.

4) In particular, for *WDPFLA* and *SDPFLA*, it is important to note that the number of noises accommodation states (the size of the noise accommodation states space $\left|\mathbb{S}_{nas}\right|$) must be at least the number of normal gap days in the corresponding periodic pattern.

4.5.2. Remarks on the Comprehensive Scenario Simulation

As mentioned at the very beginning of this chapter, to avoid unnecessary confusions introduced, the simulations we have done were arranged in a separate manner. That means, we tested the performance of each *FLA* in their "private" environment. However, there might be many different and time-variant periodic patterns occurring together in a real-world scenario.

The difficulty of running such a comprehensive simulation lies in that it is very confused for even people to distinguish anomalous meetings from those regular meetings *artificially* in a complicated scenario consisting of multiple and time-variant periodic patterns. Considering the case that even we cannot pick out the anomalies from the testing dataset precisely, how can we know which alerts should be the false ones in such a comprehensive simulation?

In view of this, plus considering the scope of a Master thesis, we decided to leave the comprehensive scenario simulation for the proposed solutions to the future work. However, this doesn't prevent us suggesting a powerful "*context simulator"* [38] in the ubiquitous computing environment. With this simulator, the developers have the freedom to design any user-defined scenarios. These scenarios can be visualized and represented in a friendly way. After running the scenarios, the corresponding contextual datasets can be gathered for the purpose of machine learning [38].

Based on the *context simulator* describe above, We have used the map of Grimstad to develop a comprehensive scenario in which the main roles are the two author of this Master thesis (Figure 44). The readers who are interested in the scenario development can feel free to contact us.

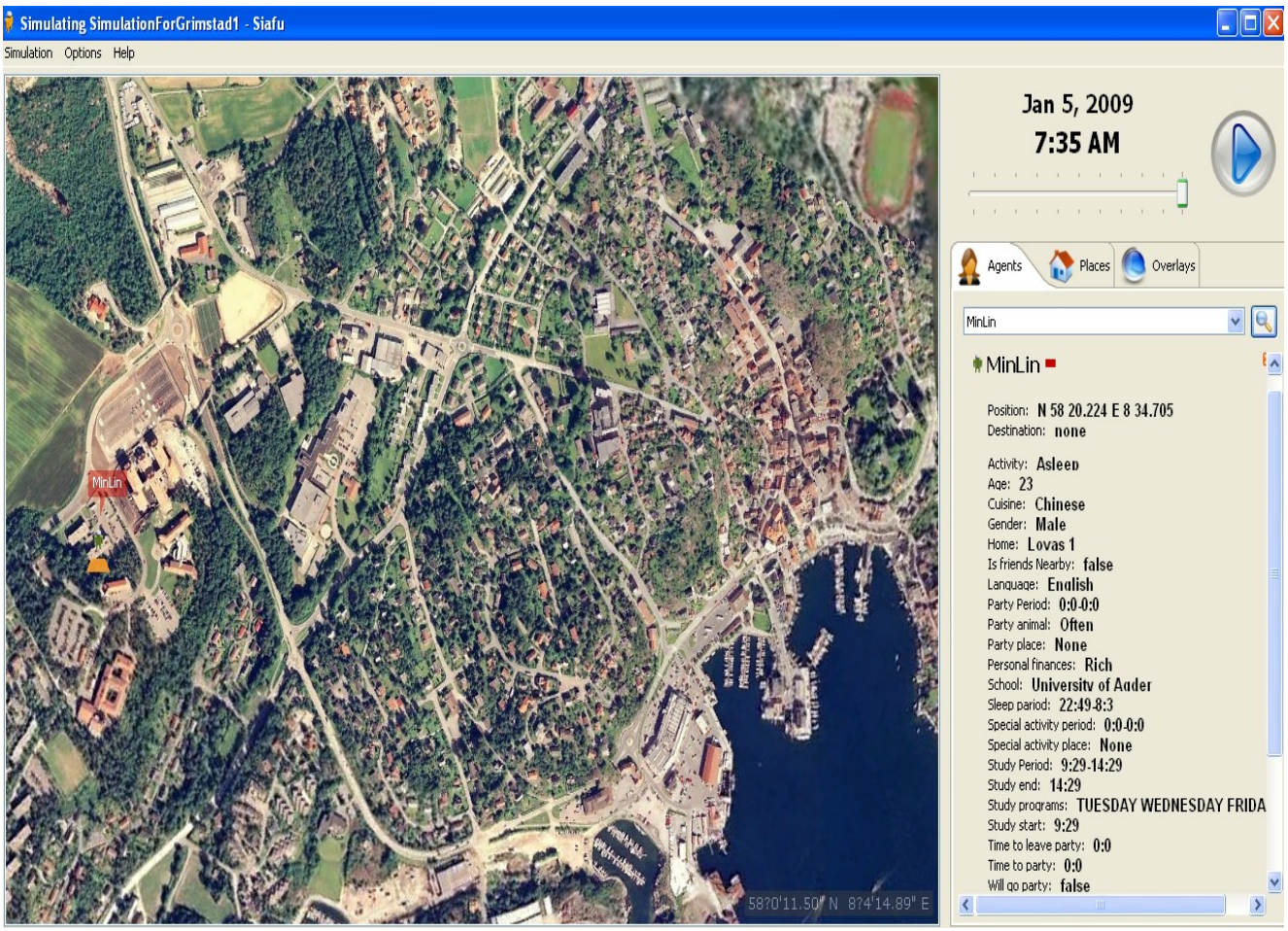


Figure 44 A comprehensive scenario simulation (in Grimstad)

5. Prototype Development

Mobile phones act as the role bridging the geographical environments to connect peoples. Ten years ago, mobile phones were just used to transmit messages in modes of calling or SMS. Recent years, as the rapid increasing in mobile phone market makes a new infrastructure for various novel applications, mobile phones are more than message transmitters, but personal assistants, especially incorporated with a variety of accessories, such as Bluetooth, Wi-Fi, GPS and accelerometers. A mobile phone user, with the help from his 'personal assistant', is able to experience two worlds, one with human's eyes, the other one with the personal assistant's sensing abilities. Over time, more and more creative sensors are yet deployed on mobile phones and these sensors become more and more important to users.

Two years ago, when iPhone was released, people must be astonished by a variety of considerate sensors, such as accelerometer sensor, proximity sensor. Proximity sensor is able to detect the presence of nearby objects without any physical contact. When users lift the mobile phones to their ears, the proximity sensor immediately turns off the display to save power and prevent inadvertent touches. Accelerometer sensor detects iPhone rotating and registers these fluctuations, enabling iPhone to execute specific commands according to the preconfigured settings. Nowadays, accelerometer-equipped mobile phones can improve localization, and can identify human-activity, including sitting, standing, walking and others, proposed in CenceMe[39]. Project Satire [40] develops smart clothes equipped with accelerometer sensor motes to monitor human activities and trigger alerts in specific situations.

As discussed above, it is not hard to find that mobile phones possessed by more than two billion people have the inherent character of being functional carriers which resemble interfaces. Without doubts, sensor technology has become essential portion of satisfying the requirements of ubiquitous computing, in particular the evolution of wireless sensor networks, the detailed implementation scheme to be presented later. In the following section, we make introduction to the wireless sensor network, analyzing its properties, because it applies to our communication structure. In addition, in order to design an interaction architecture fitting our proposed "Periodic Patterns Recognition", another technology is introduced—Localization, which is the common need in ubiquitous computing, because location and time are always the most basic concerns, as well as privacy-protecting. Since this architecture aims at providing a suitable platform for "Periodic Patterns Recognition", the privacy-protecting policies in this architecture must be fitting presence-sharing.

5.1. Related Technologies

5.1.1. Wireless Sensor Network

Wireless sensor network is the self-organizing network consisting of a large number of low-cost, sensing, data processing, and wireless communicating sensor nodes, and independent of base stations or mobile routers to form a network through distribution protocols [41].

Wireless sensor network is characterized with self-organizing, large scalability, high redundancy, easy topology, limited power and so on. The major difference between wireless sensor network and former networks is that wireless sensor network operates at the data-processing as core, with the function of data transmission. These features represent advantage of wireless sensor network, and wireless sensor network could be broadly applied, especially in ubiquitous computing.

Figure 45 shows the typical architecture of wireless sensor network [41].

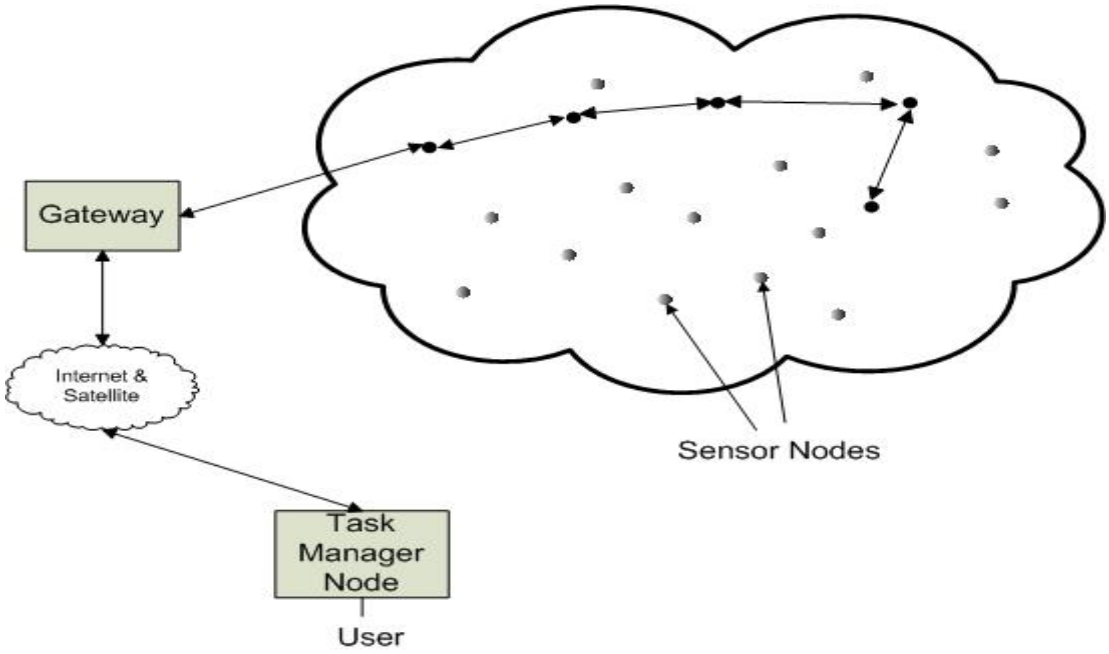


Figure 45 Architecture of wireless sensor network

Sensor nodes have the capabilities of sensing, signal processing and wireless communicating, which are not only the packet senders, but also the packet forwarders. Through the mechanisms of self-organizing and multi-hop communication, the valid data is sent to gateway which is able to communicate with external network in different ways, such as Internet, satellite and so on. In the situation that a large number of nodes reside, we can deploy a number of gateways to work in distributed control.

In wireless sensor network, nodes that support only short distance communication because of limitation of power provision can exchange data only with neighboring nodes. If it is needed to

access nodes that are out of communication range, multi-hop routers are necessary. To ensure most of nodes can set up radio link to gateway, the distribution of nodes must be considerably dense. In addition, the wireless sensor network shall have nice self-organizing, as the states of nodes can easily vary due to mobility or power consumption and the topology structure of wireless sensor network varies easily.

Currently, wireless sensor network has been adopted to the applications based on ubiquitous computing, and its' enormous value in military reconnaissance, industry production and health monitoring draws countries' attention, and wireless sensor network evolves as a hot field of research in global industry and academics.

5.1.2. Localization

To serve our "Periodic Patterns Recognition" well, localization of users is included in our design. There are four candidate schemes to be considered, GPS, Place Lab [42], Mac address based localization and hybrid localization scheme proposed in [43].Of course, to leverage Google latitude is a good idea, however, no API is provided from Google, and therefore we don't take it into account.

"A location system can provide two kinds of information: physical and symbolic. GPS provide physical positions. For example, our building is situated at 47°39'17" N by 122°18'23" W, at a 20.5-meter elevation. In contrast, symbolic location encompasses abstract ideas of where something is: in the kitchen, in Kalamazoo, next to a mailbox, on a train approaching Denver." [44]

Place Labe and Mac address based localization are the symbolic localization system, whereas hybrid localization scheme can be identified as both symbolic and physical localization system. The following illustration presents their respective advantages and weaknesses.

- **GPS;** The Global Positioning System is perhaps the most widely popular localization system. In present, the most of high-end mobile phones are equipped with GPS. Apparently, to locate user's current position, GPS is our best and convenient choice on high-end mobile phones. However, some of consumers prefer not to purchase high-end mobile phones because of high price and relatively big size. Besides, unacceptable battery life of less than 7 hours when GPS is used continuously is a fatal weak point. The good accuracy (around 8 meters) GPS can bring to users, nevertheless, is the primary reason to choose GPS.
- **Mac address based localization;** Without GPS, people can also locate where they are, but at least their mobile phones are equipped with Wi-Fi modules. The principle of this

localization is simple: uploading those Mac addresses of access points which you are interested in, writing names corresponding to those Mac addresses, identifying them at next time you detect them. Every wireless access point broadcasts a unique ID (Mac address), which is used as a "landmark" to identify a particular location. They exist as our "address book", so when our devices see an access point A with its unique ID again, it can recognize this access point and tell user the name associated with this access point. There are two applications [45] [46] based on this method of localization, which are applicable to those people who don't have GPS on their mobile phones and wish to locate their positions. Speaking of the weakness of this localization, the number of positions able to be located depends on the number of recorded access points and false position information may occur when the access point is removed from a location. However, it can work better than GPS indoors.

- **Place Lab [42];** Place Lab is more or less similar to the method above. Their analogy is that both of them use the signal transmitted from certain place as "Landmark" to locate current position, yet Place Lab works smarter and provides more accuracy position. Envision a person can see three landmarks; University of Agder, Rema 1000, Løvås Student Accommodation. Using these landmarks as fixed points of reference and triangulation, people can approximate current position, which is more accuracy than the position given only by one signal. Additionally, signal strength becomes a very important factor while approximating position. Of course, signal may be from base station or Wi-Fi access point.

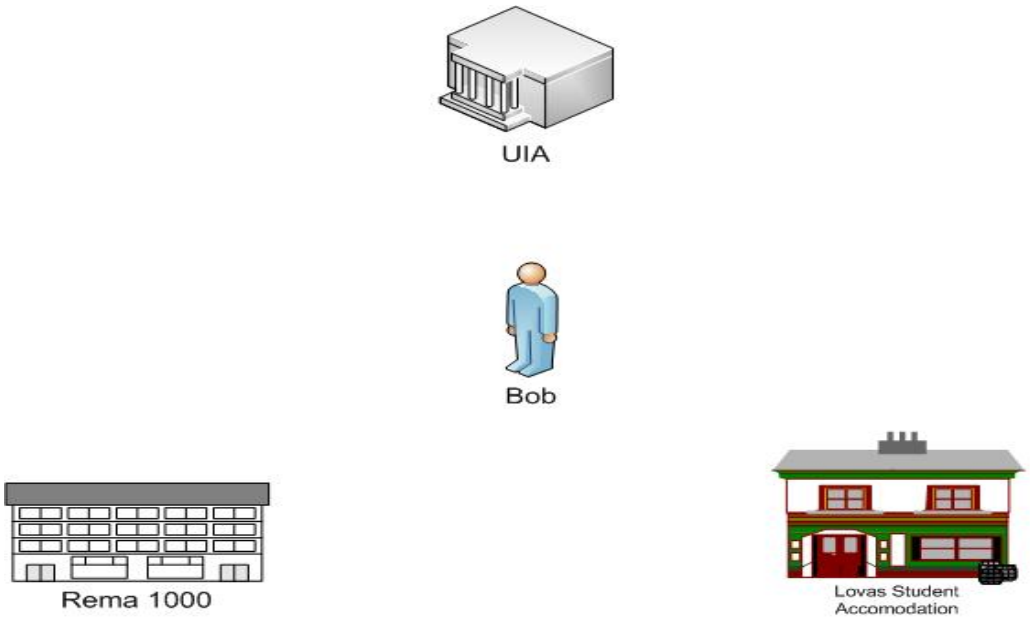


Figure 46 Triangulation positioning

- **Hybrid scheme;** In the research of "Micro-Blog: sharing and querying content through mobile phones and social participation" [43], a clever approach is developed, which argues that switching between different schemes to achieve a better energy-accuracy tradeoff might be beneficial, instead of choosing a static localization scheme.[43] Because, whatever scheme of single-facility based localization to be compared with other ones, energy –accuracy tradeoffs is always a point of contention. Hybrid form of localization looks for the most suitable mode to provide position information in different situations. The complete performance analysis between hybrid form and single-facility scheme is presented in [43].

5.1.3. Privacy Security

Privacy concern not only in this architecture to be presented later, but also in all of practical architectures holds magnificent attention, for supplying with the basic reliability. Especially in an open wireless network, malicious actions are latent in order to expose information from communication. Our focus is on privacy protection of presence-sharing in which communication between friends must be secure and identities of users are protected against exposing from inference or brutal breaking. In [47], "SmokeScreen", an effective approach to resolve issues of presence-sharing is proposed. We will borrow some ideas from "SmokeScreen" and adapt them to our design. The below statement based on "SmokeScreen" is the design goals for protecting presence-sharing in our case.

The features of independency, self-interested and non-constraints underlie the basic criteria for users to decide whether accepting to participate in presence-sharing. With respect to designing a privacy-protection policy for presence-sharing, the design goals should be set out along with maintaining those three features. We state the three design goals below, which are derived from "SmokeScreen".

- ***"Control"*** User-related information should be fully under users' control. Users are able to authorize the specific people with the right to reveal user-related information [47].
- ***"Disclosure"*** Users should be capable of knowing who viewed their presence after their presence is revealed [47].
- ***"Isolation"*** To prevent potential uncertain threats, the validity of revealing presence at one moment should not be influenced by past or future. In other words, this is either time-bound or condition-bound presence-revealing system [47].

5.2. Architecture and Design

Our basic requirement is a platform which can run “Periodical Pattern Recognition”. As mobiles phones are good carriers for sensors and exist in our daily life, the concept of sensor network is introduced to describe this kind of interaction environment where people use mobile phones as their assistants to provide more sensing abilities. For example: Users can use their mobile phones to know their current position, if their phones are equipped with GPS. This kind of sensing ability is relatively simple, as GPS is coming to the common module in mobile phones. As to our issue, the difficulty is greater, because our issue is to add friend-presence sensing ability to our mobile phones. Besides, this friend-presence sensing service (In general, sensing abilities bring users the sensed information services) demands that users can be aware of presence of their friends at anytime and anywhere. Therefore, Infrastructure-based architecture does not suit the demand from friend-presence sensing service. Ad-hoc mode based architecture is our choice, due to its self-organizing, easy-to-connect and mobile phones with Wi-Fi are more common than with GPS, therefore that it is easy for users to construct a temporary network. Figure 47 illustrates the overview of ad-hoc network running on mobile phones.

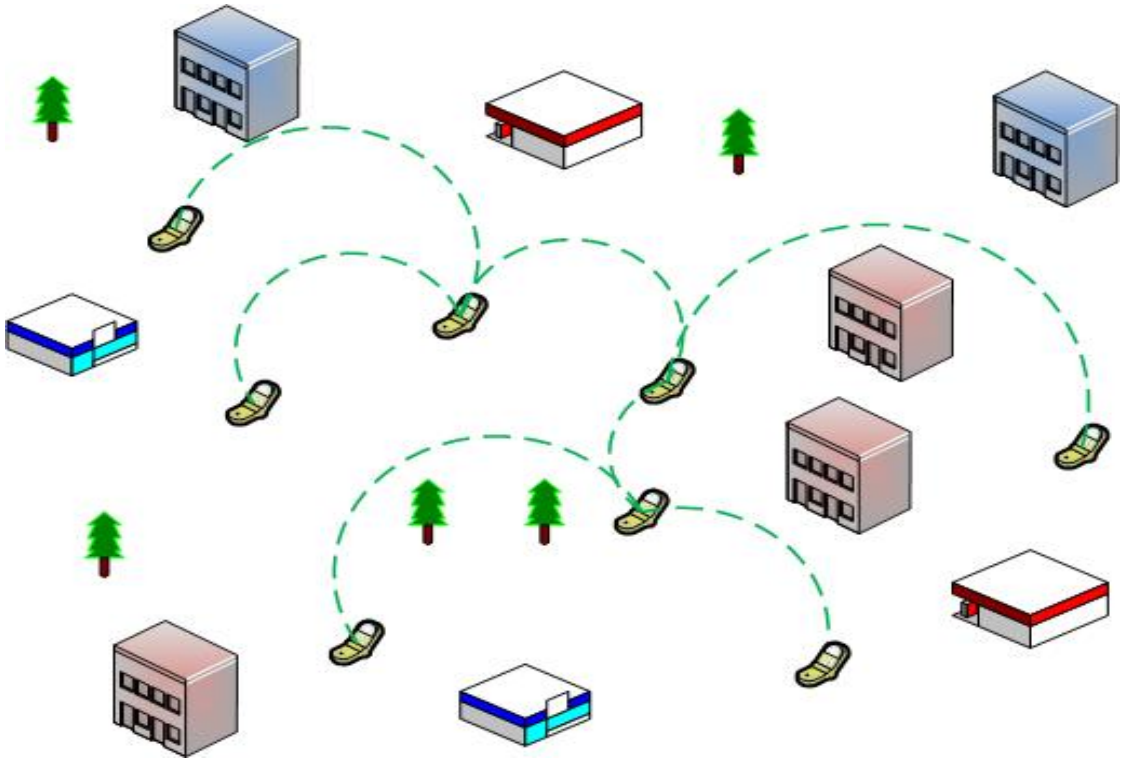


Figure 47 Ad-hoc network

5.2.1. Communication in Ad-hoc Network

The features of sensor network have been presented in previous. The ad-hoc network in which mobile phones works as sensor nodes has the most of features same as sensor network. Sensors in mobile phones are like eyes and ears of users to obtain external sensitive information and respond the useful information to users.

This information can be identified as context information and they can fall into four primary categories: Computing context, Physical context, User context and Time context. Computing context includes communication bandwidth, network delay, and nearby resources. Physical context includes temperature, light, noise and so on. User context includes user position, user profile, user social status, etc [48].

If context information is actively providing, or passively responsive, such as the current user-position information, mobile phones can perform this service with specific devices, such as GPS module. Although from the perspective of users the position query service is transparent which means users just need to click the query button on their mobile phones, the working process of GPS modules involves a sequence of actions, such as searching for satellites, receiving signals from satellites. If the users need to query specific information and this specific information is queried through a sequence of communication, achieving to receive this information over a temporarily existing network could be lucid. In addition to the merit that lucid communication could be over a network, the concerns of security may be addressed more easily. The security design for this communication will be presented soon after. The temporarily exiting network is the ad-hoc network. In this ad-hoc network, users can participate in a variety of activities. For our goal, users must know the presence of their friends in this temporarily existing network.

5.2.2. Friend-Presence Discovery

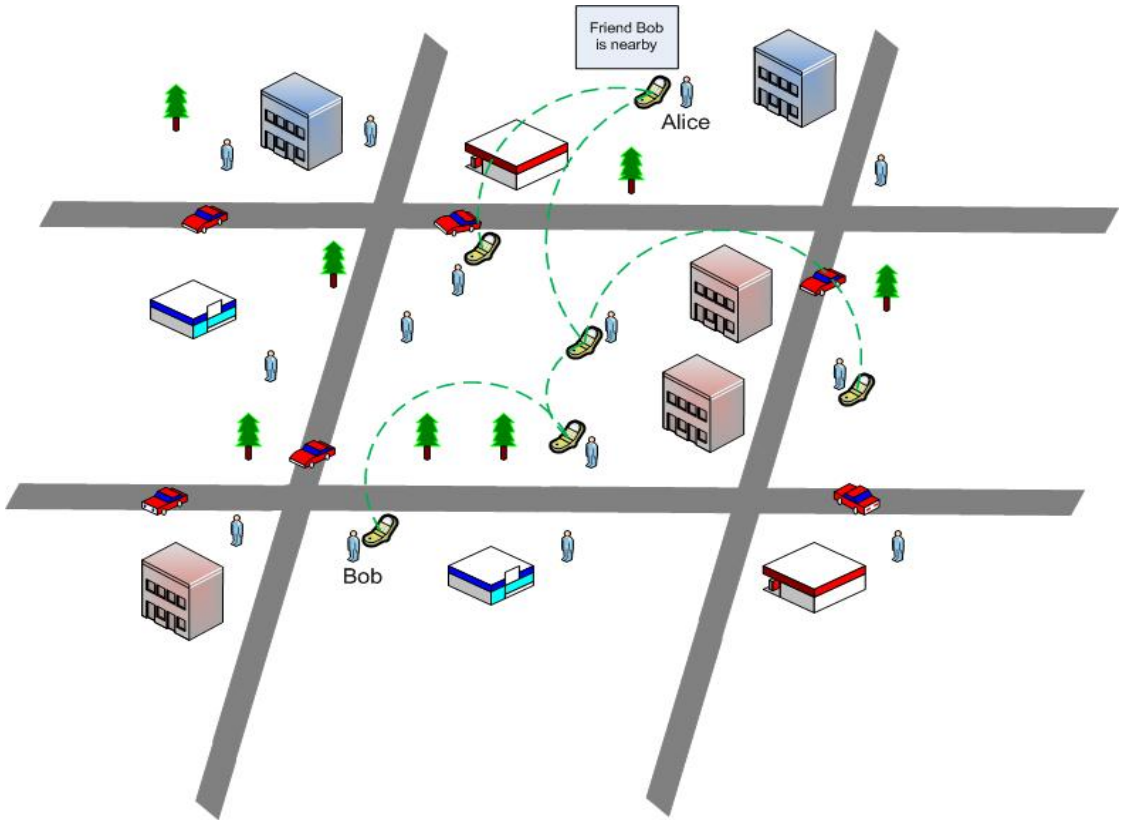


Figure 48 Friend-presence sensing in ad-hoc network

As Figure 48 shows above, six mobile phones form a temporary network based on ad-hoc mode. Possibly, these six users do not know each other. At the very beginning, one of them originates an establishment of an ad-hoc network. Anybody nearby him can join this network. Some subsequent problems must be evident in this kind of network, such as dynamic routing protocol of ad-hoc, ad-hoc network security management, but these aspects are beyond our studying and we will refer to some of these problems later. We assume that this ad-hoc network is visible to everyone and there is no entry limitation in this network.

In Figure 48, user Alice is informed of the presence of her friend Bob. The specific mechanism for friend-presence informing is like that Bob finds there is an ad-hoc network and Bob is interested in this network, therefore he entering this network. As soon as he enters this network, he broadcasts his name Bob to everyone in this network. Certainly, as long as Alice is in this network, she will know Bob is nearby. This friend-presence informing mechanism is easy to understand, but too simple, because this is the most fundamental structure.

5.2.3. Friend-Proximity Definition

In Figure 48, Alice must find whether Bob is present through Bob's name packet forwarding over

two mobile phones, namely two hops. Possibly, this network is very large in which thousands people interact. If a friend is one thousands meters away from you but he can still broadcast his name in this network to inform you of his presence, is it supposed to say your friend is proximate to you? Of course, the meaning of friend proximity may be arbitrarily defined. However, in "Periodical Pattern Recognition", friend-proximity is defined as the physical distance is less than 100 meters between friends, because, at a distance of more than 100 meters, one might be in a location different from another one. Thus, users allow their names to be broadcasted across at most one mobile phone; in other words, their names can be forwarded at most once (The transmission distance of Wi-Fi is ranging up to 100 meters).

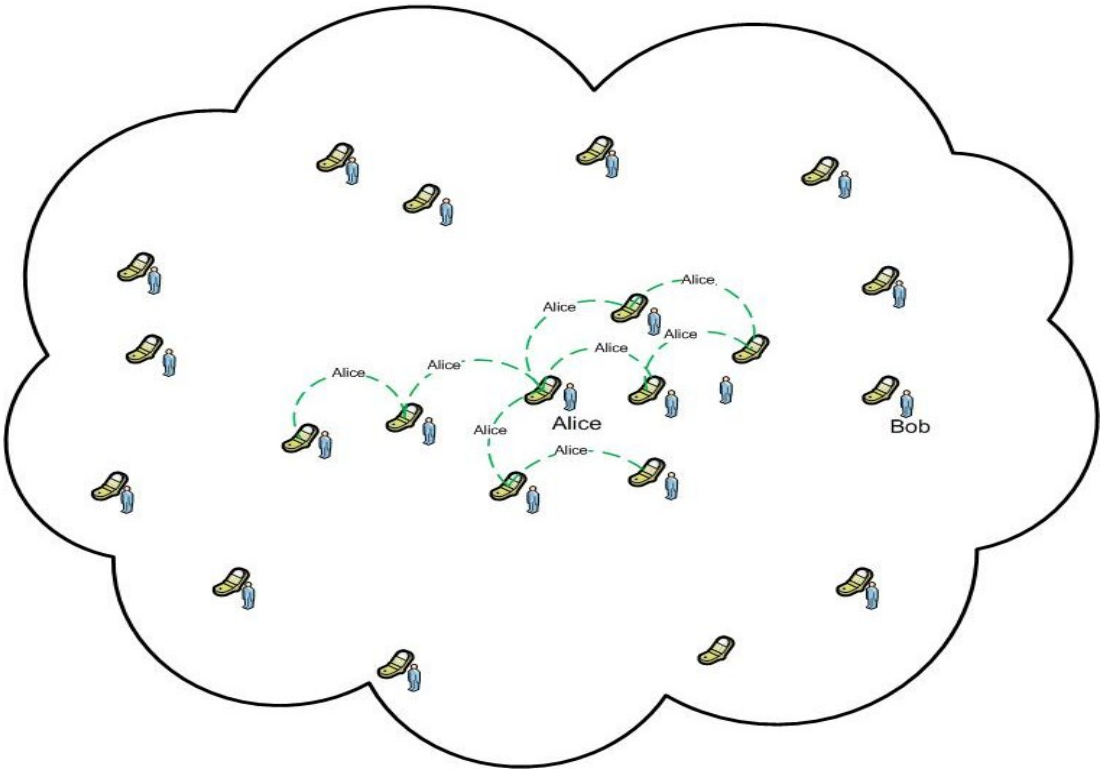


Figure 49 Restricted broadcasting in ad-hoc network

If users frequently broadcast in ad-hoc network, frangible wireless ad-hoc network cannot support great signal load. In [49], the research gives "The broadcast storm Problem in a mobile ad hoc network", and in [50], several broadcasting techniques for mobile ad-hoc networks is introduced and compared. The specific solution to on-demand distance broadcasting in ad-hoc network is beyond our work, but they have been presented in some ad-hoc network related publications.

5.2.4. Collaboration of Localization and Friend-Presence Discovery

There are four schemes to localize users, introduced in the previous section. Since the user location and time information is important to "Periodical Pattern Recognition", the following part

will explain the collaboration of localization and friend-presence discovery on the ad-hoc network. We don't refer to how to obtain time information, because time information can be known directly from our mobile phones.

According to the specification of "Periodical Pattern Recognition", there are three elements that must be provided: time at friend-presence, location at friend-presence, name of discovered friends. A scenario is given below to describe the procedure of collecting these three elements.

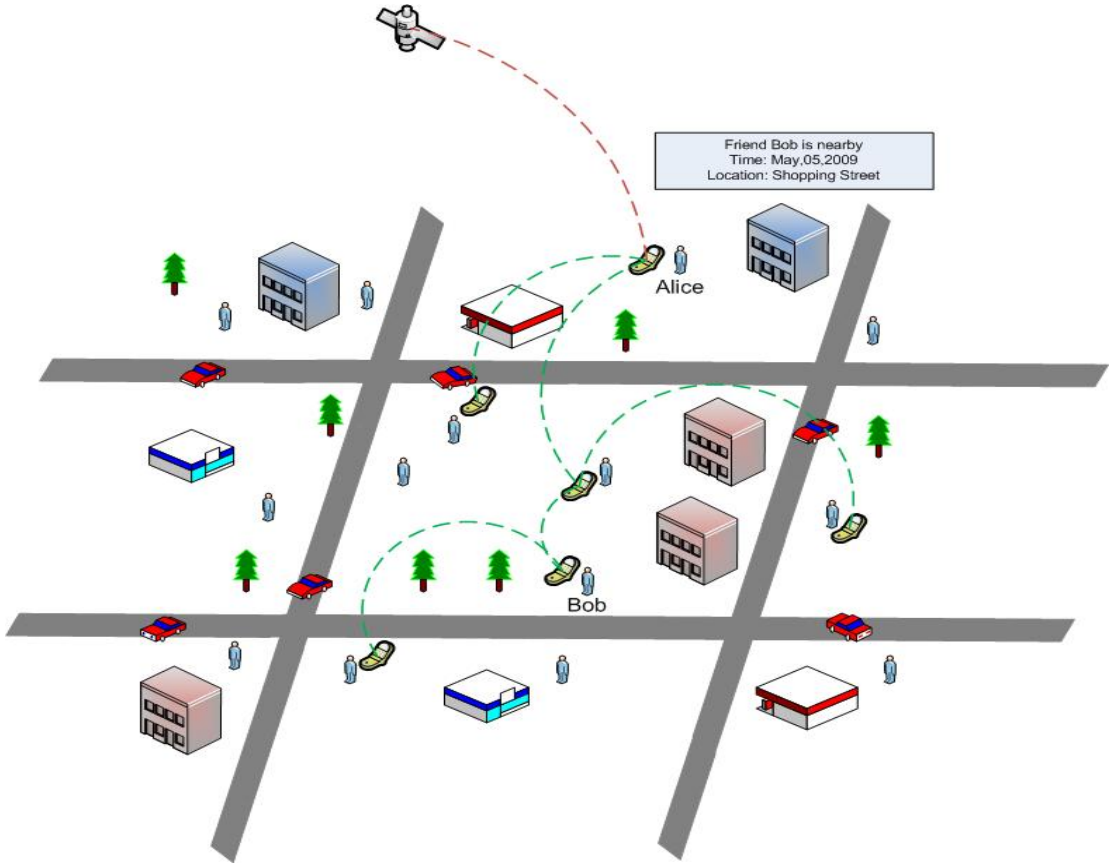


Figure 50 Collecting information

Bob tells everybody who is within 100 meters away from Bob that he is present in this network, and then Alice receives the message from Bob and knows her friend Bob is in proximity. She subsequently obtains her current position from GPS using her mobile phones and records the current time; at last she puts this information as elements to "Periodical Pattern Recognition". This is how it works. Not only can the current user position be obtained from GPS, but also from other means. As we refer this ad-hoc network as a wireless sensor network, people can obtain their current position from external sensors in this ad-hoc network, which is like that in a certain region, some sensors are scattered and facilitate specific information provision; they can also join a ad-hoc network which is held in the way talked about always, and users authorized to control them can leverage them to obtain particular contexts, not only position context, but also various contexts. In [51], it is proposed and implemented to use Bluetooth and a database of user profiles to cue informal, face-to-face interactions between nearby users who don't know each other, which is full of originality. Extending our scheme to support more interesting services will be the future work.

5.2.5. Privacy Controls for Presence-Sharing

In order to prevent user from being tracked, users' identities should not be exposed. In Figure 50, Bob broadcasts his name in an ad-hoc network, which leads to everybody in this ad-hoc network knowing that Bob is present. We will give the solution to resolve the concerns of privacy protection.

According to our goal of privacy control for presence-sharing which makes reference of [47] and declares previously, the first task is to satisfy the "control" requirement. This is not hard, and users can broadcast their names as they like at anytime and anywhere, or stop broadcasting as they like. The second is to satisfy "disclosure" requirement. "SmokeScreen" has proposed a solution which also satisfies our requirement for "disclosure", so that we just apply the method from "SmokeScreen" in our designing. The following is the explanation to the method. As Bob enters an ad-hoc network, he broadcasts his hashed name, which can be interpreted only by his friends. Therefore, nobody knows that Bob is in proximity except for his friends who know Bob's hashed name.

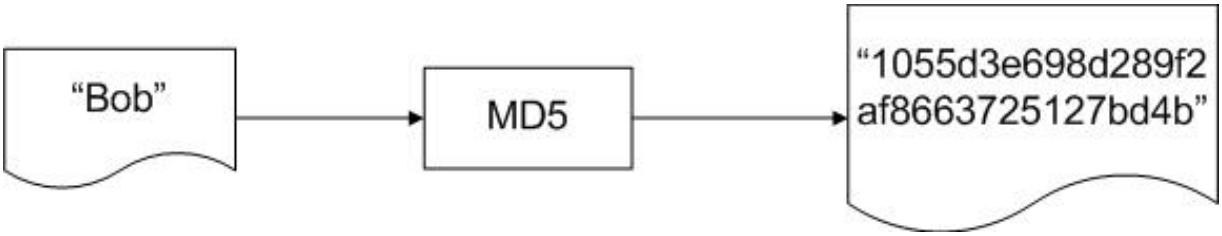


Figure 51 Hashing name

Alice knows "1055d3e698d289f2af8663725127bd4b" means "Bob", so that she can know Bob is nearby as long as she receives this message. Two problems happen here: Consider a scenario in which only Bob, Alice, and a malicious sniffer are in the ad-hoc network and Bob broadcasts his hashed name. The malicious sniffer records the message from Bob, and if next time this malicious sniffer receives the same broadcasted message from Bob, he can infer either Alice or Bob must be in this ad-hoc network where he is. Another problem is that how to guarantee this message is from Alice's friend Bob though anybody can broadcast the same message in the ad-hoc network?

To resolve the first problems given above, Bob must deterministically and independently update his hashed name, which also meet the requirement of isolation. At the same time, Alice must also know Bob's updated hashed name.

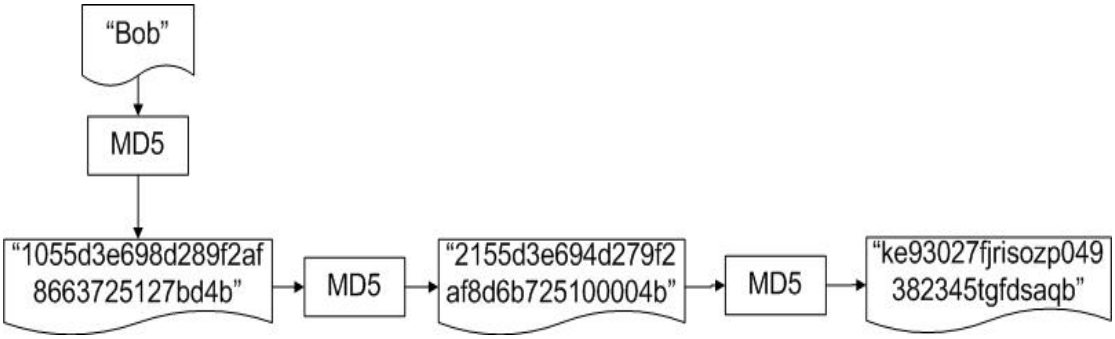


Figure 52 Updating the hashed name

Alice and Bob must be synchronized to independently update the hashed name. For example, if Bob updates his hashed name every one hour starting with a certain time point, Alice must update Bob's hashed name as soon as Bob updates his hashed name. For simplicity, we assume each user's hashed name is updated on the hour.

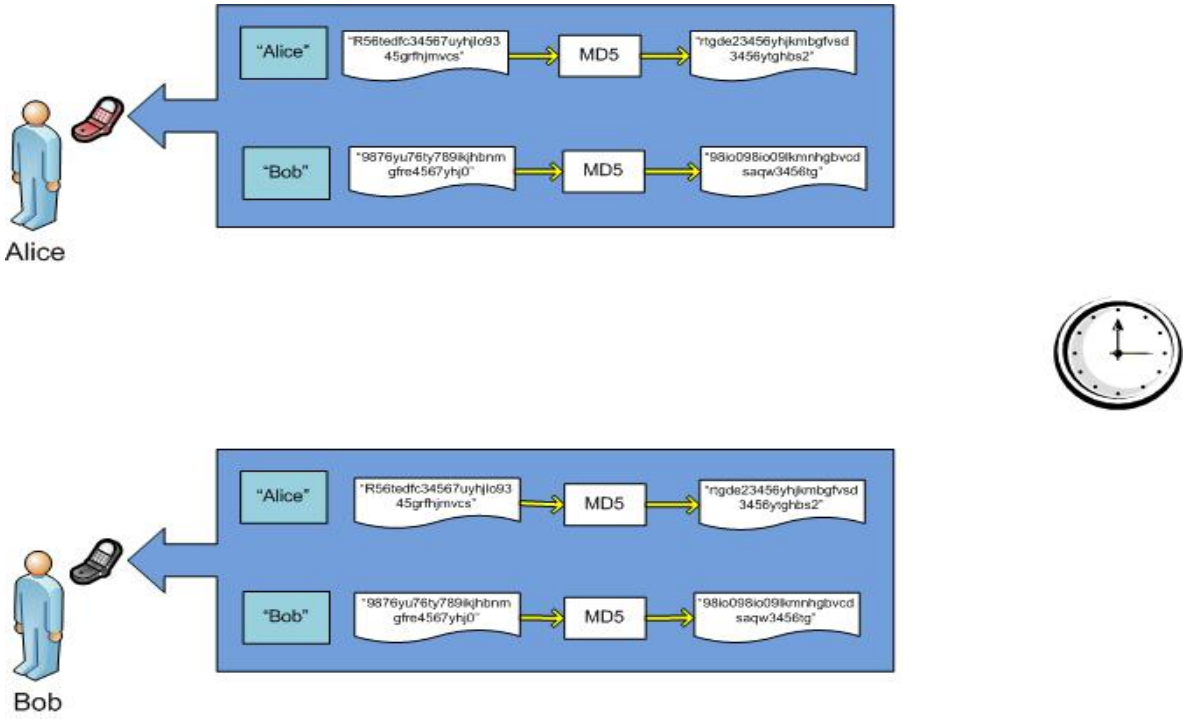


Figure 53 Updating the hashed name on the hour

This approach can prevent the hashed name from being unchanged, and prevent malicious sniffer from inferring the presence of users. In ad-hoc network, the transported messages are visible to users in this network. If Bob and Alice want to go further than knowing the presence of each other, they must negotiate a symmetric key to secure the communication between them. Similarly, the negotiated symmetric key must vary on the hour and be known only to Alice and Bob, in order to prevent malicious users from computing the subsequent hashed name of Bob's. For example: Although Bob's hashed name varies every one hour, his subsequent hashed name could be computed by everyone. Because the hash function MD5 is public to everybody. Consider that Charlie is not Bob's friend, but once he gets a hashed name from Bob, he can know all his updated hashed names by the same hash function MD5. According to "SmokeScreen", the symmetric key is referred to compute the hashed name. In our case, a user's symmetric key is shared with his friends. As to "SmokeScreen", the symmetric key is a clique key which is owned by all clique members and one clique has one symmetric key, while one user has a personal symmetric key in our case.

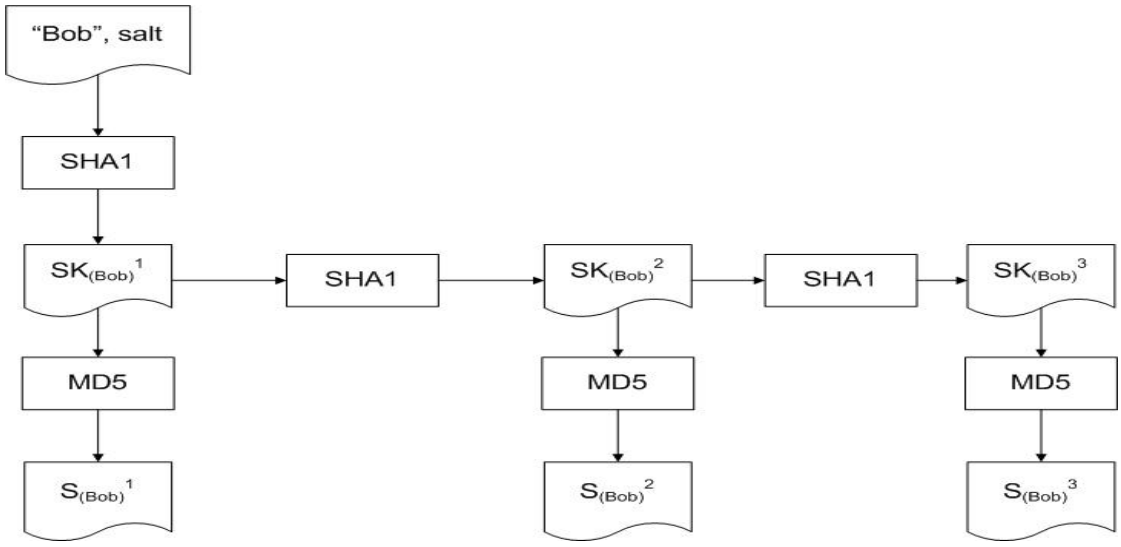


Figure 54 Symmetric key and hashed name generation [47]

Bob uses "Bob" and salt to generate his symmetric key through SH1, and use the symmetric key to generate his identity signal. Only Bob's friends and Bob hold Bob's symmetric keys, so that it prevents malicious sniffers from computing Bob's subsequent identity signals. Because, malicious sniffers do not have Bob's symmetric keys and cannot compute Bob's subsequent identity signals, while for his friends who have his symmetric keys, they can generate Bob's both symmetric keys and identity signals, identifying Bob at anytime and anyplace.

Alice doesn't expose her identity to unintended people in the ad-hoc network, but reveal identity only to Alice's friends. A problem emerging, Alice's friends can generate Alice's identity signal and pass off as Alice. If Alice adds her signature after her identity signal, problem is solved. We suppose everyone has own private key and their public keys are known publicly. Thus, their signatures are generated through their private keys.

For resolving the second problem, let's see a scenario in which Alice decides to notify her friends of her presence in the ad-hoc network at 2:00.14.March.2009. At the first, Alice uses his private key to sign his identity signal $S_{(Alice)}^{2}$ and current time t as $Sig(S_{(Alice)}^{2}, t)$, and Alice's current symmetric key is $SK_{(Alice)}^{2}$. What Alice is broadcasting is a message of $S_{(Alice)}^{2}$,t and $E(Sig(S_{(Alice)}^{2}, t), SK_{(Alice)}^{2})$, which implies $S_{(Alice)}^{2}$ is known only to Alice's friends and only Alice's friends can decrypt $E(Sig(S_{(Alice)}^{2}, t), SK_{(Alice)}^{2})$, then use Alice's public key to prove that this identity signal is definitely from Alice.

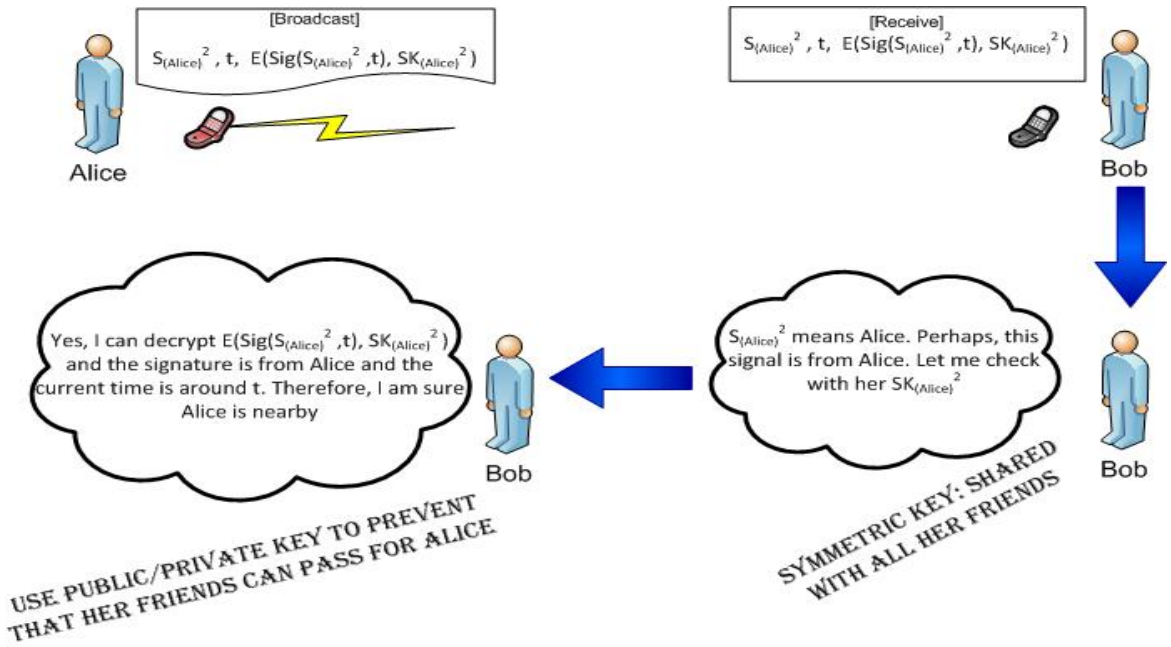


Figure 55 Decrypting the encrypted message

After Bob has verified Alice's message, if Bob is willing to share his presence with Alice, Bob can broadcasts his message in the same way. In last scenario, why do we send E(Sig($S_{(Alice)}^2$,t), $SK_{(Alice)}^2$) along with $S_{(Alice)}^2$? As Alice adds her signature after her identity signal, nobody can pass off as Alice, but sending $S_{(Alice)}^2$ and Sig($S_{(Alice)}^2$,t) without symmetric cryptography is big vulnerable. If David is not Alice's friend but has Alice's public key, he doesn't have to identify Alice's identity signal but knows Alice is nearby with using public key to verify Alice's signature.

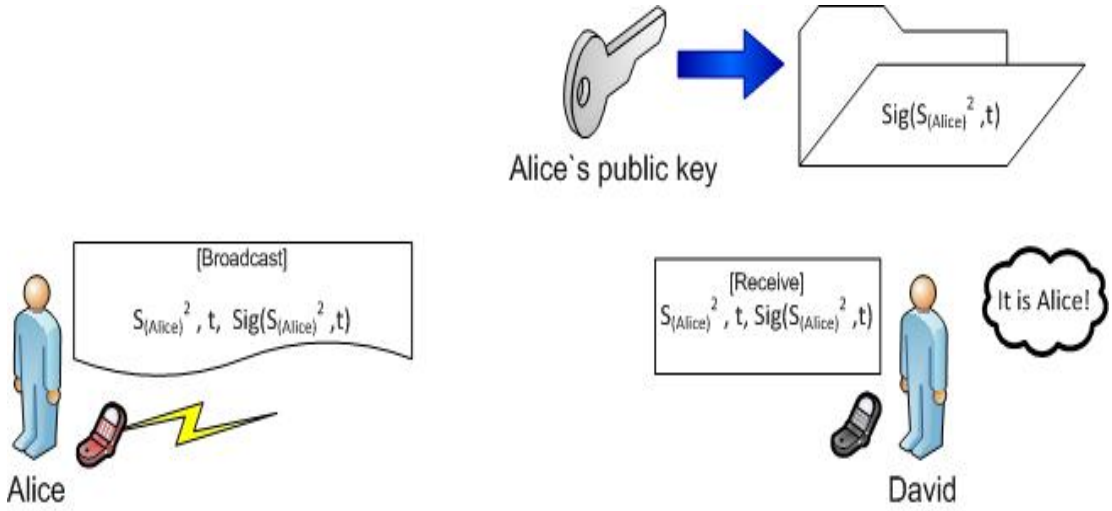


Figure 56 Exposing identities by public key

If Alice encrypts his signature with his symmetric key, only his friends can decrypt the E(Sig($S_{(Alice)}^2$,t), $SK_{(Alice)}^2$) and verify Alice's signature. That is why E(Sig($S_{(Alice)}^2$,t), $SK_{(Alice)}^2$) is so important.

5.2.6. Presence-Sharing Between Specific Friends

We envision that Bob broadcasts his identity signal to notify his friends of his presence, but some friends who receive Bob's notification don't want to inform Bob of their presence. Of course, users can operate their mobile phone not to respond their identity signal when they receive friend's presence information, but it is too verbose.

There is also a situation where Bob wants to conceal himself from some friends who know his IDs and Keys when he is sharing presence with friends.

Another concern is that since mobile phones are resource-limited devices, it is supposed to minimize the CPU consumption. Try to think everyone in the ad-hoc network intends to broadcast identity signals to notify friends, and then each one of the members in this ad-hoc network must check every incoming identity signal by means of table-lookup to find if there is user's friends. In this way, it results in big energy-consuming.

In order to perfect our platform and address those problems mentioned above, bloom filter is applied in our designing, because its implementation can lead to not only reducing space and time cost, but also giving a way to filter unintended friends from sharing presence.

Bloom filter is proposed in [52], 1970 by Burton H. Bloom, and in recent years, has become very popular in the networking literatures. And in [53], the survey makes introduction to some typical application of Bloom filter and several variations of Bloom filters.

"The Bloom filter principles: *Wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated."*[53]

In our case, if false positives happen, non-friend of senders still cannot identify senders, but friends of senders might identify senders. So, we will discuss how to design a Bloom filter for our platform and maximally reduce the probability of false positives.

If Bob needs to inform one hundred friends of his friends, let's look how to use Bloom filter to help specific-friend notification.

According to the method of computing the least false positives

$$"g = -\frac{m}{n} ln(p)\, ln\,(1-p)\ ,$$

When p=1/2 or equivalently k=ln2 · (m/n), then the false positive rate f is $(1/2)^k$=$(0.6185)^{m/n}$. *In practice, of course, k must be an integer, and smaller k might be preferred since they reduce the amount of computation necessary."* [53].

In above illustration, m denotes the number of bits in the Bloom filter, n denotes the number of

elements inserted into the Bloom filter, and k denotes the number of hash functions.

We design two Bloom filters; one resides in user's native mobile phone for deciding if to share presence when receive presence information from friends; the other one is used to spread to every one for filtering unintended friends and strangers.

As to the first Bloom filter, since it resides in mobile phones and the storage capacity in mobile phones could be not a big problem, we pursue the aim of lower probability of false positives and lower amount of computation. Assume that the number of specific friends to be notified is fixed, and then adjust those arguments m, k, and f to realize a desired Bloom filter. For the detailed process under this mechanism, Bob creates a Bloom filter containing his friends intended to share his presence with, and this Bloom filter resides in his mobile phone. When an identity signal comes to Bob's mobile phone, the identity signal will be placed in this Bloom filter to check if it is positive. If it is, it means the sender is one of Bob's friends to be notified of Bob's presence information and then respond Bob's identity signal to this friend. If it shows negative, no response is sent. Of course, identities mentioned here are encrypted in the way designed in the previous section.

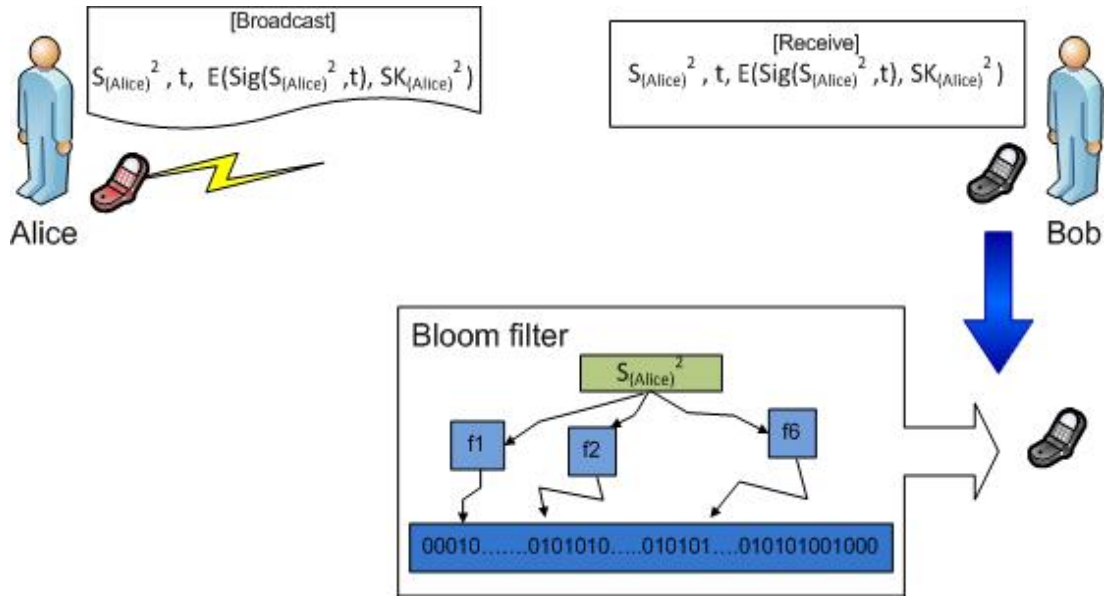


Figure 57 Bloom filter resides in native mobile phones

The latter Bloom filter is transported along with identity signals, which is not secure at a certain extent. We must ensure the program may protect this Bloom filter. This Bloom filter also carries the information of the friends to be shared presence with, but is transported in ad-hoc network. For example, Alice broadcasts her identity signal and a Bloom filter which holds the friends to share her presence with. If her identity signal and Bloom filter reach her friends, then the program will check the identity signal first to determine if it is recognized. If it can be identified, then check if this friend is in Alice's Bloom filter. If it is positive, the program will remind this friend that Alice is proximate. If not, the program will abandon this message from Alice. It is obvious that if the program is falsified, then her friends can know Alice is proximate through avoiding the check of Bloom filter. Since the size of Bloom filter is critical to transportation in

ad-hoc network, the arguments should be adjusted to realize the least length of bits at acceptable conditions of probability of false positives and amount of computation.

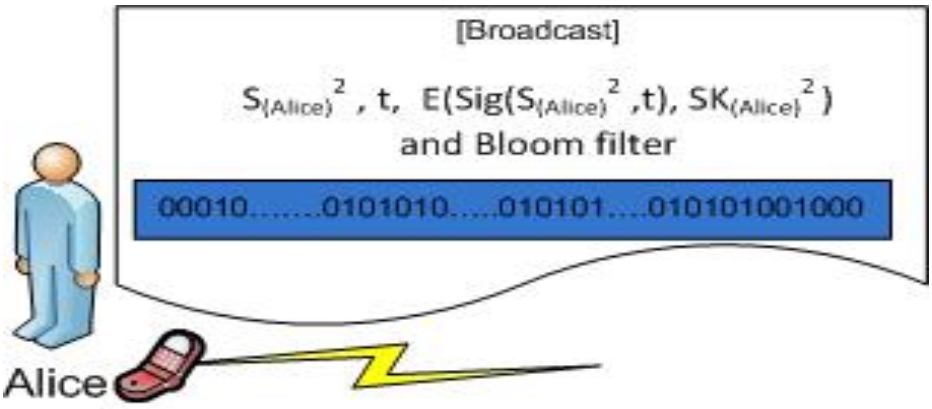


Figure 58 Bloom filter transports in ad-hoc network

5.2.7. Software Architecture

Figure 59 presents the software architecture to be implemented. Communication client contains two sockets, UDP socket for general signals transmission, TCP socket for special information transport, such as exchange of keys and IDs between friends. Service in the below figure refers to program modules, such as “Periodical Pattern Recognition”, Bloom filter. External context information is collected with the specific modules and delivered to Data processing, waiting for related programs to query. Interaction Client manages the coordination of the work of modules and maintains friendly human-computer interaction.

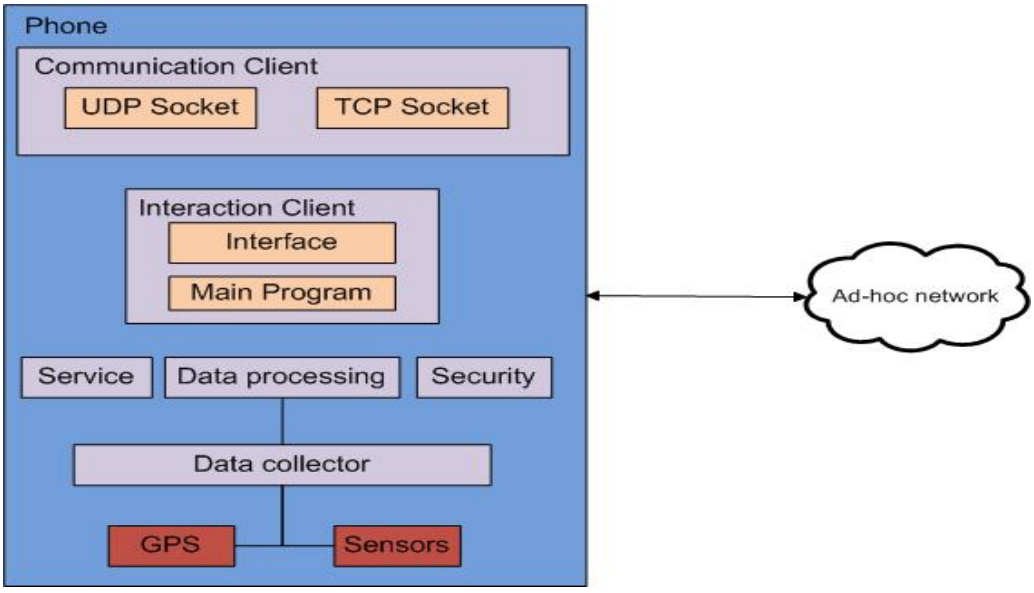


Figure 59 Block diagram of software architecture

5.3. Implementation

Our prototype system consists of two mobile phones: HTC P3300 and Sony Ericsson X1, both of which are equipped with Wi-Fi modules. C# is the programming language to implement this

prototype. As Windows mobile 6 professional is the OS of both of these two mobile phones, Windows mobile 6 SDK in C# can provides us quickly fashioned prototype. Taking advantage of managed libraries, those components that should not be our concerns are able to be implemented easily.

5.3.1. Forming an Ad-hoc Network

These two mobile phones forms as ad-hoc network through setting their connection configuration to be in ad-hoc mode. This network is called “Small World”, and any Wi-Fi enabled device in ad-hoc mode can join this network without authentication request. After this network is established, these two mobile phones actually communicate by the means similar to infrastructure.

One of the mobile phones broadcasts its explicit IP to verify his success of joining this network, as the figure below (Figure 60) shows.

Figure 60 Joining ad-hoc network

By this time, P3300 doesn’t provide user’s ID, but exposes its current IP for verification of having connected to this network. In the real use, this step is not necessary.

5.3.2. Acquiring IDs and Keys

Symmetric key is generated through user’s typing user’s name, and personal ID is generated from

symmetric key. Specifically, users type their names, and system adds a random number of unfixed length after their names. By the means of “SHA1” hashing algorithm, personal symmetric key is generated first, and then through “MD5” hashing algorithm, personal ID is generated, as described in the design section. Users’ current IDs and symmetric keys are saved in the file folder “/SmallWorldFile/MyInfo”. Once users want to either change their IDs and symmetric keys or generate their descendant IDs and symmetric keys, the former IDs and symmetric keys will be replaced with the new ones.

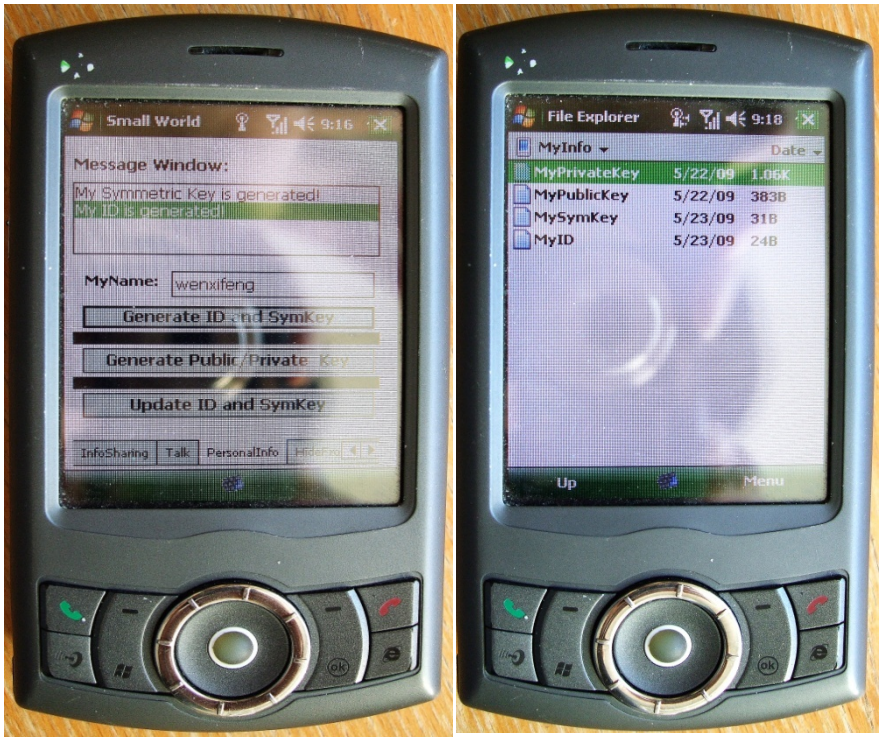


Figure 61 Generating ID and Keys

RSA is the algorithm of public-key cryptography we use for digital signature. Acquire personal private key and public key, and save them in same file folder as where personal IDs and personal symmetric keys are saved. Note that, usually users don’t have to replace their RSA key pair with a new one. If a user decides to renew the RSA key pair, he better informs his friends who hold his public key that his RSA key pair is changed in case that their friends cannot identify him at next time.RSA key pair generator is implemented by C# managed library, which can provide RSA key pair to users.

5.3.3. Exchange of IDs and Keys Between Friends

To exchange IDs and keys between friends, one user should make a private ad-hoc network to share his ID and keys with his friends. Since this network is encrypted and not public, exchange of IDs and keys between friends resembles the common file content sharing, with good security. A specific thread and a port number are assigned to exchange of IDs and keys between friends. After one user executes receiving commands, his friends can send their IDs and keys through his IP and the specific port number. As for the user, the specific thread is opened for receiving until time is out or receiving is done. The program will assemble the received data of byte type to data

of text type or xml type; store them in the file folder "/SmallWorldFile/FriendInfo". We realize exchange of IDs and keys between friends over TCP transportation protocol because of the high demand for data integrity.

Figure 62 Exchange of personal information

5.3.4. Presence Sharing

As users have all his friends' IDs and keys, they can create the two Bloom filters to achieve sharing presence between specific friends. Of course, users can also share their presence with all their friends without creating the Bloom filters. In view of lacking enough users to participate in our evaluation, a set of simulated name data is input for testing the performance of those two Bloom filters.

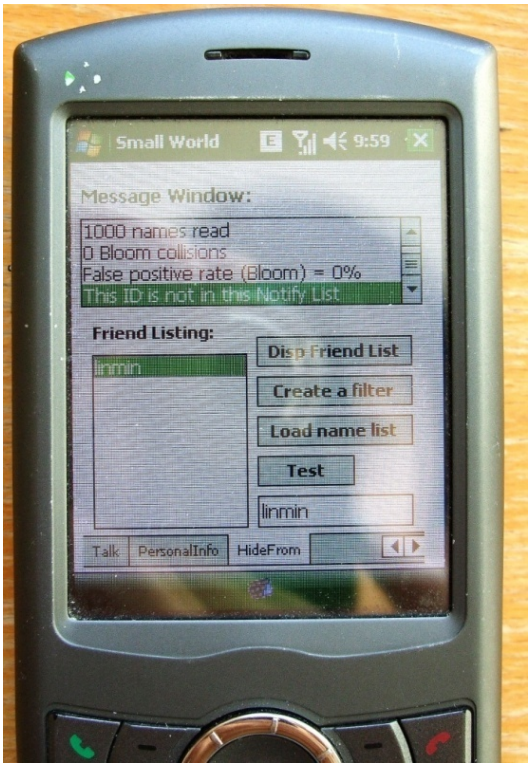


Figure 63 Testing Bloom filter

Although the screen shots of the phone display the simple interface with friends' IP and presence information, the actual process includes a series of encryption, decryption, verification and file manipulation. In the presence sharing process, UDP is the transportation protocol used to broadcast in this unstable ad-hoc network.

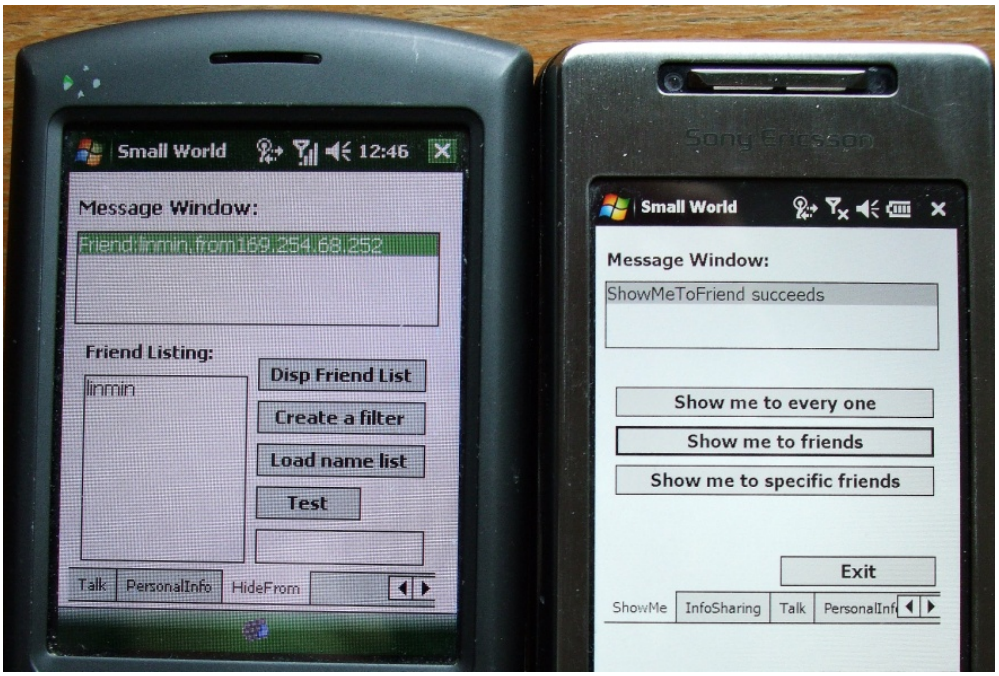


Figure 64 Discovery of friend-proximity

6. Conclusion and Future Work

In the thesis, we investigated a context-awareness issue in a Wi-Fi based ubiquitous computing environment. The key limitations stated in section 1.4 makes the research problem very challenging. We consider this issue as a periodic pattern recognition problem which was mainly addressed by machine learning techniques in the previous studies. However, unfortunately most of the existing techniques in this regard can only work in an offline manner or with a high computational complexity. Considering our solutions would serve for the mobile services, in order to pursue a better user experience, we proposed the light-weight *Finite Learning Automata* based *online* solutions to handle the problem. In order to avoid conceptual confusion, we designed one *Finite Learning automata* for each periodic pattern we considered typical and important. We have done some mathematical analysis work on the proposed solutions. Please note that the design thought of these *Finite Learning Automata* in the proposed solutions can also be applied to other periodic patterns according to the specific needs.

Considering the real-world testing on a mobile phone will take a long time (e.g. several months) for the periodic pattern recognition, we decided to run computer simulations as evaluation experiments. Our extensive experiments investigated some important properties of the proposed solutions. We chose the number of false alerts as the performance criterion for the evaluation experiments. We didn't make judgments on when the performance can be good enough since the required performance level for practical applications should be decided by industrial communities. The significance of presenting empirical results in this thesis lies in that it will help the developers from industrial communities to make decisions for a practical application based on the trends implied by the empirical results. The empirical results demonstrated how different random noises in the environment can affect the performance of the proposed *Finite Learning Automata* solutions. In addition, we have done a few mathematical formulations to supplement the trends in the empirical results. However, there are still some studies we left to the future work:

1) The performances of the *Finite Learning Automata* based solutions are generally scenario-dependent. That means, the accuracy of the solutions is considerably enslaved to the probability of the random noises. In the future, it is important to design solutions which can have relatively stable performance in different noisy environments. In addition, a carefully-designed comprehensive scenario simulation which incorporates multiple and time-variant periodic patterns should be done in the further work.

2) Though our empirical results demonstrate that all these *Finite Learning Automata* in the proposed solutions can converge to a sub-optimal accuracy (number of false alerts) in each different noisy environment. However, the mathematical proofs in this regard have not been formulated. The complicated relationships that how the sub-optimal convergence can be related to the sizes of the state spaces, the probabilities of the *random gaps* noises and *random encounters* noises are currently considered to be scenario-dependent and not well-understood yet.

3) The research problem (stated in section 1.2, analyzed in section 3.2) constraint by the key limitations (stated in section 1.3) is very complicated in nature. We feel the pure online solution is a good starting point but not sufficient to solve the research problem perfectly. The underlying reason is that we are not allowed to make use of the statistical characteristics of historical datasets in the pure online solutions. This may also lead to the *perceptual aliasing* problems discussed in the Chapter 4. We suggest that the finite-size windows which can contain a certain amount of historical data could be used to assist the pure online solution in order to achieve a better performance.

For the prototype part, we have designed an explicit architecture to support operation of our Finite Learning Automata and implemented it on two windows mobile powered mobile phones. In the architecture, similar to the wireless sensor network, the ad-hoc network for communication platform is characterized with exporting information, self-organizing, and service-sharing. In which sensor nodes are deemed being carried by the mobiles phones and mobile phones can obtain a variety of services through these nodes, as long as requestors are authorized. We assume this network is a public ad-hoc wireless network, any node and any mobile phone can join this network without entrance to be permitted, because this ad-hoc network has a fixed network name and no claim for permission. Thus, a public ad-hoc network is constructed, and users are able to communicate in this network through their mobile phones. This implies the security of communication is hard to protect, as to which, the method of "SmokeScreen" is adapted for our architecture and we use Asymmetric Cryptography to improve security mechanism of communication of "SmokeScreen". In addition, we adopt the basic Bloom filter to achieve presence-sharing between specific friends.

For our future work on this architecture, there are several points worth studying:

1) Broadcast protocol and multicast protocol in ad-hoc network is an active study. Since the notification of friend-presence needs to broadcast identity signals in the ad-hoc network, avoiding broadcasting storm in an unstable network structure is crucial.

2) In order to define if friends are proximate, the physical distance between users should be specified. In an ad-hoc network, broadcast and multicast are based on routing protocol, not based on physical environment, which means packets-transporting cannot be restricted within a specified physical region. However, there are some researches for this broadcasting and multicasting in ad-hoc network.

3) Intelligent assistants residing in our mobile phones is our final goal, but the ad-hoc network bears danger of broadcast storm under our architecture, far from providing intelligent services, by now. The key reason is that this ad-hoc network does not have the same management mechanism as infrastructure, such as fixed routers, DHCP servers, and users' mobile phones just perform some logical programs. In [54], it proposes an agent-oriented approach for intelligent sensor network. Intelligent agents [55] offer several features that could help solving the problems faced in our architecture: autonomy, reactivity, proactivity,

reasoning, character, cooperation. An example scenario is like: Friend-presence discovery is done through dynamical interaction between agents instead of broadcasting identity signals. By the advantage of intelligent agent system, the ad-hoc network can be structurally managed, and intelligent agents can live in a virtual social network founded on the ad-hoc network on behalf of users.

Reference

[1]M. Satyanarayanan, “Pervasive computing: vision and challenges,” *IEEE Personal Communications*, vol. 8(4), pp. 10-14, Aug. 2001.

[2]B. Schilit and M. Theimer. “Disseminating active map information to mobile hosts,” *IEEE Network*, vol. 8(5), pp. 22 – 32, July 1994.

[3]G.K. Mostefaoui, J. Pasquier-Rocha, and P. Brezillon, “Context-awareness computing: a guide for the pervasive computing community,” in *Proceedings of the IEEE/ACS International Conference on Pervasive Services*, 2004, pp. 39-48.

[4]A. K. Dey. “Providing Architectural Support for Building Context-Aware Applications,” PhD dissertation, College of Computing, Georgia Institute of Technology, 2000.

[5]Edwin J. Y. Wei and Alvin T. S. Chan, "Towards context-awareness in ubiquitous computing," in *Proceedings of Embedded and Ubiquitous Computing, International Conference,* EUC 2007, Taipei, Taiwan, December 17-20, 2007, Proceedings. Lecture Notes in Computer Science 4808 EUC 2007, pp. 706-717.

[6]G.D. Abowd, A.K. Dey, P.J. Brown, N. Davies, M. Smith and P. Steggles, “Towards a better understanding of context and context-awareness,” *Lecture Notes in Computer Science*, vol. 1707, pp. 304-307, Jan. 1999.

[7]V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: a survey,” *ACM Computing Surveys*, pp. 1-72, Sep. 2009 (to appear).

[8]S. Ma, and J. L. Hellerstein, “Mining partially periodic event pattern with unknown periods,” in *17th International Conference on Data Engineering*, 2001, pp. 205-214.

[9]B. Ozden, S. Ramaswamy , and A. Silberschatz , “Cyclic association rules,” in 14th International Conference on Data Engineering, 1998, pp. 412-421.

[10]J. Han, Y. Yin, and G. Dong, "Efficient mining of partial periodic patterns in time series database," in *15th International Conference on Data Engineering*, 1999, pp.106.

[11]G. Chen, H. Huang, and M. Kim, "Mining frequent and periodic association patterns," Dartmouth Computer Science Technical Report TR2005-550, July 2005.

[12]A. k. Mahanta, F. A. Mazarbhuiya, and H. k. Baruah, “Finding calendar-based periodic patterns,” *Pattern Recognition Letters*, vol. 29(9), pp. 1274-1284, July 2008.

[13]W. Lin, M. A. Orgun, and G. J. Williams, “An overview of temporal data mining,” in

Proceedings of the 1st Australian data mining workshop (ADM02), 2002, pp 83–90.

[14]J. F. Roddick, and M. Spiliopoulou, "A bibliography of temporal, spatio, spatio-temporal data mining research," *ACM SIGKDD Explorations Newsletter*, vol. 1(1), pp. 34-38, June 1999.

[15]J. F. Roddick, K. Hornsby, and M. Spiliopoulou, "An updated bibliography of temporal, spatio, spatio-temporal data mining research," *Lecture Notes in Computer Science*, vol. 2007, pp. 147-164, 2000.

[16]C. H. C. Ribeiro, "A tutorial on reinforcement learning techniques", *Supervised Learning Track Tutorials of the 1999 International Joint Conference on Neuronal Networks*, Washington: INNS Press, 1999.

[17]L. P. kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: a survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237-285, 1996.

[18]M. E. Harmon, and S. S. Harmon (1996), "Reinforcement learning: a tutorial," URL: http://www.nbu.bg/cogs/events/2000/Readings/Petrov/rltutorial.pdf, cited in Apr.5, 2009.

[19]Markov Decision Process, http://en.wikipedia.org/wiki/Markov_Decision_Process, cited in Apr.7, 2009.

[20]Markov Property, http://en.wikipedia.org/wiki/Markov_property, cited in Apr.7, 2009.

[21]Monte Carlo Method, http://en.wikipedia.org/wiki/Monte_carlo_method, cited in Apr.7 2009.

[22]R. S. Sutton, "Learning to predict by the methods of temporal differences," *Machine Learning,* vol. 3(1), pp. 9-44, Aug. 1988.

[23]C. J. C. H. Watkins, and P. Dayan, "Technical notes: Q-Learning," *Machine Learning*, vol. 8(3-4), pp. 279-292, May 1992.

[24]R. S. Sutton, and A. G. Barto, *Reinforcement Learning: An Introduction*, Cambridge: MIT Press, 1998.

[25]O.-C. Granmo, "A Bayesian learning automaton for solving two-armed Bernoulli bandit problem," in *Seventh International Conference on Machine Learning and Applications*, 2008, pp. 23-30.

[26]M. L. Tsetlin, *Automata Theory and the Modeling of Biological Systems*, New York: Academic Press, 1973.

[27]K. S. Narendra, and M. A. L. Thathachar, "Learning automata–a survey," *IEEE Transactions on*

Systems, Man, and Cybernetics, vol. SMC-4 (4), pp. 323-334, July 1974.

[28]M. A. L. Thathachar, and P. S. Sastry, "Varieties of learning automata: an overview," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 32(6), pp. 711-722, Dec. 2002.

[29]B. J. Oommen, and J. K. Lanctot, "Discretized pursuit learning automata," *IEEE Transactions on Systems, Man and Cyberneteics*, vol. 20(4), pp. 931-938, Aug. 1990.

[30] M. A. L. Thathachar, and P. S. Sastry, "Estimator algorithms for learning automata," in *Proceedings of the Platinum Jubilee Conference on Systems and Signal Processing*, Bangalore, India, Dec. 1986.

[31]A. Nowe, K. Verbeeck, and M. peeters, "Learning automata as a basis for multi-agent reinforcement learning," *Lecture Notes in Computer Science*, vol. 3898, pp.71-85, Mar. 2006.

[32]B. J. Oommen, and M. Agache, "Continuous and discretized pursuit learning schemes: various algorithms and their comparison," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 31(3), pp. 277-287, June 2001.

[33]M. S. Obaidat, G. I. Papadimitriou, and A. S. Pomportsis, "Efficient fast learning automata," *Information Science—Informatics and Computer Science: An International Journal*, vol. 157(1-2), pp. 121-133, Dec. 2003.

[34]B. J. Oommen, and M. Agache, "Generalized pursuit learning schemes: new families of continuous and discretized learning automata," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 32(6), pp. 738-749, Dec. 2002.

[35] O.-C. Granmo, and N. bouhamala, "Solving the satisfiability problem using finite learning automata," *International Journal of Computer Science and Applications*, vol. 4(3), pp. 15-29, Dec. 2007.

[36] K. S. Narendra, and M. A. L. Thathachar, *Learning Automata: An Introduction*, New Jersey: Prentice Hall, 1989.

[37]T. M. Mitchell, *Machine Learning*, Illinois: McGraw Hill, 1997.

[38]M. Miquel, and N. Petteri, "A generic large-scale simulator for ubiquitous computing," in *Third Annual International Conference on Mobile and Ubiquitous Systems: Networking & Services*, 2006, pp. 1-3.

[39]E. Miluzzo, N. D. Lane, S. B. Eisenman, and A. T. Campbell, "CenceMe – Injecting sensing presence into social networking applications", *Lecture Notes in Computer Science*, vol. 4793, pp. 1-28, Oct. 2007.

[40]R. K. Ganti, P. Jayachandran, T. F. Abdelzaher, and J. A. Stankovic, “Satire: a software architecture for smart attire,” in *Proceedings of the 4th International Conference on Mobile Systems, Applications and Services,* 2006, pp.110-123.

[41]I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E.Cayirci, “Wireless sensor networks: a survey,” in *Computer Networks*, vol. 38(4), pp.393-422, Mar. 2002.

[42]A. LaMarca et al., “Place lab: device positioning using radio beacons in the wild,” *Lecture Notes in Computer Science*, vol. 3468, pp. 116-133, May 2005.

[43]S. Gaonkar, J. Li, R. R. Choudhury, L. Cox, and Al Schmidt, “Micro-Blog: sharing and querying content through mobile phones and social participation” in *Proceedings of the 6th International Conference on Mobile Systems, Applications and Services*, 2008, pp. 174-186.

[44]J. Hightower, and G. Borriello, “Location systems for ubiquitous computing”, in *Computer*, vol. 34(8), PP. 57-66, Aug.2001.

[45]Herecast, http://www.herecast.com/, cited in Apr. 20, 2009.

[46]WiGLE, http://www.wigle.net/, cited in Apr. 20, 2009.

[47]L. P. Cox, A. Dalton, and V. Marupadi, “SmokeScreen: flexible privacy controls for presence-sharing” in *Proceedings of the 5th International conference on Mobile Systems, Applications, and Services*, 2007, PP. 233-245.

[48]B Schilt, N. Adams, and R. Want, “Context-aware computing applications,” in *the first Workshop on Mobile Computing Systems and Applications*, pp. 85-90, 1994.

[49]S. –Y. Ni, Y. –C. Tseng, Y. -S. Chen, and J. –P. Sheu, “The broadcast storm Problem in a mobile ad hoc network,” in *Proceedings of the 5th annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1999, pp. 151-162.

[50]B. Williams, and T. Camp, “Comparison of broadcasting techniques for mobile ad hoc networks” in *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking & Computing*, 2002, pp. 194-205.

[51]N. Eagle, and A. Pentland, “Social serendipity: mobilizing social software,” *IEEE Pervasive Computing*, vol. 4(2), pp. 28-34, 2005.

[52]B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors” *Communications of the ACM*, vol. 13(7), 422-426, July 1970.

[53]A. Broder, and M. Mitzenmacher, “Network applications of bloom filters: a survey,” in *Internet Mathematics*, vol. 1(4), pp. 485-509, 2004.

[54]B. Karlsson, O. Bäckström, W. Kulesza, and L. Axelsson, “Intelligent sensor networks –an agent-oriented approach,” in *Workshop on Real-World Wireless Sensor Networks*, June 2005.

[55] N. R. Jennings, and M. Wooldridge, “Intelligent Agents: Theory and Practice” *Knowledge Engineering Review*, vol. 10(2), pp. 115-152, 1995.