DISS. ETH NO. 23271

Frequent Episode Mining for Smart Home Wireless Sensor Network

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH ZURICH
(Dr. sc. ETH Zurich)

presented by

LI LI

M.Arch., Southeast University, China
born on 18.08.1984
citizen of P.R.China

accepted on the recommendation of

Prof. Dr. Ludger Hovestadt
Chair of Computer-Aided Architectural Design, ETH Zurich

Prof. Dr. Li Biao
School of Architecture, Southeast University, China

2016

Abstract

By recognizing patterns in daily activity, smart homes are able to personalize their services. To achieve this, established technologies are available for data collection and sequence pattern recognition, but although many studies have been carried out in this field, these technologies all share the drawback that the methodology used for data collection tends to be ill suited for the purposes of pattern recognition. In particular, the specific demands on engineering and power supply for data transmission are not usually taken into consideration. For this research, we have developed a bespoke Wireless Sensor Network (WSN) and combined it with a compact data format for Frequent Episode Mining (FEM) to overcome this obstacle.

The data format encompasses both sensor data about spontaneous events and periodic environmental readings. The format has been optimized for battery use, requiring a minimum of power for data transmission and processing. Two different WSNs are being introduced for data collection: the first solution is a self-organizing mesh network that requires neither planning nor configuration and lends itself especially well to spontaneous, short term monitoring. The second solution is a tree-structured network which is extremely energy efficient and therefore particularly suitable for long-term installations. Schemes and data are stored in a temporary database. The algorithm for sequence pattern recognition has been expanded to include visualization functions.

The proposed framework has been evaluated with both synthetic data from a smart home simulator and with real data from a self-organizing WSN in a student home. We have been able to demonstrate that the framework is capable of recognizing activities and making predictions as a basis for new personalized services. The framework is self-contained, scaleable and energy efficient, and is thus applicable in different smart home settings.

Keywords: smart home, wireless sensor network, frequent episode mining

Zusammenfassung

Indem Smart Homes Muster täglicher Aktivitäten erkennen, können sie die von Ihnen bedienten Services besser personalisieren. Für diese Aufgabe bieten sich etablierte Technologien zur Datenerhebung und Sequenzmuster-Erkennung an. Zwar wurden in diesem Zusammenhang viele Studien durchgeführt, sie alle aber haben das Problem, dass die Verfahren der Datenerhebung den Erfordernissen der Mustererkennung nur schlecht entsprechen. Vor allen Dingen werden in der Regel der Bedarf an Engineering und Energie für die Datenübertragung nicht berücksichtigt. Um diese Hürde zu überwinden, wurde in dieser Forschungsarbeit ein spezielles Wireless Sensor Network (WSN), zusammen mit einem kompakten Datenformat speziell für ein Frequent Episode Mining (FEM), entwickelt.

Das Datenformat umfasst sowohl Sensordaten von spontanen Ereignissen als auch periodische Messungen der Umgebung. Das Format wurde dahingehend optimiert, dass es, um einen Batteriebetrieb zu ermöglichen, möglichst wenig Energie für den Transport und die Verarbeitung der Daten benötigt. Es werden zwei verschiedene WSN für die Datenerhebung vorgestellt. Die erste Lösung ist ein vermaschtes, sich selbst organisierendes Netz, das keine Planung oder Konfiguration benötigt und besonders geeignet ist, für spontanes und kurzzeitiges Monitoring. Als zweite Lösung wird ein baumartiges Netzwerk vorgestellt, das extrem wenig Energie verbraucht und daher für langfristige Installationen geeignet ist. Schemen und Daten werden in einer sogenannten Temporalen Datenbank abgelegt. Der Algorithmus für die Sequenzmuster-Erkennung wurde um Funktionen zur Visualisierung erweitert.

Das vorgeschlagene Framework wurde sowohl mit synthetischen Daten aus einem Smart Home Simulator, als auch mit realen Daten eines selbstorganisierenden WSN in einem Studentenheim evaluiert. Es konnte gezeigt werden, dass das Framework Sequenzen von Aktivitäten erkennen kann, und Vorhersagen als Grundlage für neue personalisierte Services treffen kann. Das Framework ist in sich abgeschlossen, skalierbar und energieeffizient. Es ist daher in verschiedensten Smart Home Umgebungen einsetzbar.

Keywords: smart home, wireless sensor network, frequent episode mining

Contents

Chapter 1. Introduction

Every morning on a turkey farm, the farmer comes to feed the turkeys. A scientist turkey, having observed this pattern to hold without change for almost a year, makes the following discovery: "Every morning at eleven, food arrives." On the morning of Thanksgiving, the scientist announces this law to the other turkeys. But that morning at eleven, food doesn't arrive; instead, the farmer comes and kills the entire flock.

CIXIN LIU, THE THREE-BODY PROBLEM

1.1. Background

Market

Although the smart home has been promised for a long time, it still hasn’t managed to find its way to mainstream consumers (Balta-Ozkan, Boteler et al. 2014),yet there is no lack of smart home products on the market. In the 1970’s, X10 released their power line communication based control system, but it was a very technical and sophisticated system and difficult to install. Since a smart home is a very complex system, to have an overall integrated solution is still very expensive. Some smart home solution providers, like VIA International, Vivint, Creston, Control4, Savant, and AMX Home automation, provide their technologies to construction and architectural firms for wealthy clients. On the other hand, in the consumer market, most of the recent products focus on one specific application, such as light control, auto door locker, and thermostat control. Such kinds of products include Nest and Honeywell Lyric Thermostat, Tado Cooling, Goji and Kwikset smart lock, Piper Wi-Fi Surveillance, Philips Hue light control, and Sonos HiFi. By providing accessibility from the home network, products can be easily connected with smart phones or tablets. Another approach is to provide a hub with some basic functionalities that allow other smart devices to be connected and expended, such as Ninja Sphere, SmartThings, Homey, Revolv, Canary, Sentri, etc. Functionalities provided by these products are limited.

A survey conducted by Icontrol Networks showed the top five most desired smart home devices are: Self-adjusting thermostat (72 percent), doors that can be locked from a remote location (71 percent), a master remote control for all the household preferences (68 percent), home monitoring cameras (65 percent), and automatic adjustable outdoor lighting (65 percent). It is clear that **connectivity** and **adaptability** are two keywords in the consumer market.

Technology

The development of a smart home system largely relies on technologies from the Electrical Engineering and Informatics fields. There are a lot of achievements in these research fields in terms of **Data Mining (DM)**, **Micro-Electro-Mechanical Systems (MEMS)** and **wireless technologies**.

Due to the information explosion, the **Knowledge Discovery in Databases (KDD)** is in high demand for making use of huge amounts of data. **Data mining (DM)**, as a particular step in the KDD process that extracts pattern from data (Fayyad, Piatetsky-Shapiro et al. 1996), has attracted a lot of attention from researchers recently. DM techniques, such as machine learning and pattern recognition, have been successfully applied in web mining, custom baskets analysis and image recognition. With **Big Data**, we can make use of **correlations** found in data, instead of understanding the **causality**. Take **Natural Language Processing (NPL),** for example. The earlier approach relied on **Syntactic Structures** analysis, which tried to analyze sentences with grammar. This approach soon reached a bottleneck, because there are lots of exceptions that do not strictly follow grammar rules. Until the late 1980s, when the first **statistical machine translation** systems were developed, a big leap in performance took place. This approach is based on statistical models instead of grammar.

On the hardware side, with the recent advances in **Micro-Electro-Mechanical Systems (MEMS)** technology. The size and cost of sensors have been brought down significantly, which makes them affordable in the consumer market. The smart phone, composed of dozens of sensors (acceleration, compass, approximate, touch sensor, etc.), is a good example. At the same time, more and more low-energy, short distance **Personal Area Network (PAN)** and **Local Area Network (LAN)** are available in the market, such as Zigbee, Bluetooth LE, and UWB, which facilitates indoor data transmission. Furthermore, major processor manufactures have released their low-energy, low-cost **microcontrollers** specially designed for embedded applications. With adequate calculation power, the battery life can be extended significantly. With the advance and prevalence of low-cost, low-power sensors, computing devices, and wireless networks, **pervasive computing** has evolved from a vision to an achievable and deployable computing paradigm (Chen and Khalil 2011). To install or integrate **Wireless Sensor Networks (WSNs)** into houses is becoming more and more practical and feasible in terms of cost, size, energy consumption, and complexity.

Problem

By mining the data collected with home WSNs, a smart home can learn and adapt to its inhabitants' lifestyles instead of only following preprogrammed tasks. A lot of studies have been conducted on data acquisition and data mining for the smart home system, however, no practical solution has been provided yet. Admittedly, the home environment is much more complex than image recognition or web mining, and more factors should be considered when applying these technologies. But in terms of research methodology, the main cause of the problem is the data acquisition and data mining have only been studied separately. WSN researchers keep improving the network regardless of what kinds of data are needed by mining algorithms, and the algorithm developers develop algorithms without knowing if the input data is available or feasible with current WSN technologies. There is no integrated solution that can combine them together efficiently.

1.2. Major Objective of the Research

The main objective of this research is to provide a framework that can integrate both data acquisition and data mining processes practically and efficiently. The main goal is divided into three objectives:

- **A summary of existing problems in smart home technology**

  By reviewing on the smart home products, markets, and research, this study will summarize the current status of smart home development and the main problems that exist that are preventing smart home technology from going into the main consumer market.

- **A comparison of available data collection and processing techniques for the smart home**

  Data collection and data processing are two major tasks in smart home research. WSNs and DM techniques are widely used in these two tasks, respectively. Since both WSN and DM are very broad definitions, there are lots of variations and different applications for each. By surveying well-known smart home projects, the author will study the pros and cons of different approaches.

  The study on WSN is going to find out (1) the difference in wireless protocols, (2) what kinds of sensors are used and where are they installed, and (3) how the collected data are formatted?

  On the DM algorithms side, these questions will be answered: (1) what kinds of DM algorithms are commonly used? (2) what kind of tasks they are assigned? (3) how are they adapted to smart home research? (4) what are the performance, features, and limitation of each algorithm? and (5) in what kind of experimental environments were the studies conducted?

- **Propose an integrated solution for pattern detection for the smart home**

  Most studies only focused on one aspect, either data collection or data processing, regardless of if they fit with each other. For example, some data mining approaches need a labeled data sample, which means the occupants have to write down what they are doing, such as reading, cooking, making coffee, etc. Although the proposed algorithms have very good performance, acquiring such kinds of data is only feasible in laboratory studies and can’t be applied in the real world.

  In this research, by examining both WSN and DM techniques, the author is trying to determine the most practical combination of them. From the data collection point of view, instead of collecting as much data, only a minimum data sample for the processing step will be collected, which will result in less sensors, smaller size, and less power consumption. On the data processing side, less data means less calculation load and less input noise. A framework will be designed to integrate the whole process.

1.3. Scope and Limit of the Research

This research includes three main parts: literature review, the design of both hardware and software, and the evaluation of the proposed framework.

The literature review was based on (1) papers and books from 1950 to 2015, (2) surveys conducted by authorities, and (3) online articles related to smart home history, smart home projects, wireless sensor networks, and data mining used in smart home technology.

The prototype of the hardware for data collection was designed base on sensors, microcontrollers, and transceivers that are available on the market. The optimization of the power consumption was focused on the software and data collection strategy, rather than hardware. The compactness and efficiency of the circuit design is not deeply investigated here.

The software design includes the firmware on the WSN node, the protocol of the network, the data base manager, the convertor for third party data source, the data mining algorithm, and the synthetic data generator. C++ and Java were chosen as programming languages. MySQL was selected as the database management software.

The data used for evaluation was collected from two different sources: synthetic data and data recorded in a student dorm. Since it is hard to know the properties of patterns in real life data, the performance of the proposed algorithm was evaluated with synthetic data. This research will focus on detecting patterns, rather than prediction.

1.4. Novel Contribution of the Research

In this research, the author proposed a systematic design for sequential pattern mining in the smart home. The design provides a seamless and optimized solution for data collection and data processing. For data collection, two different WSN are provided for short-term and long-term application.

For data processing, a new algorithm based on frequent episode mining is provided for mining the frequent sequential pattern in the sensor data database. Compared with existing algorithms, it has several key advantages. First, it can deal with both discrete and continuous values. Secondly, it can extract both frequent and rare patterns. Thirdly, it preserves occurrence information of patterns, which helps backtrack the patterns in the sequence. Furthermore, a graphic interface is provided.

1.5. Outline of the Thesis

The rest of the thesis is organized as follows. In chapter 2, a literature review on smart homes and related research will be given to understand the current status and existing problems in smart home development. Chapter 3 will focus on the WSN technologies used in smart homes, where two different WSNs for data collection in this research are proposed. One is Self-organizing WSN, which maximizes the flexibility of the network structure and installation. The other one, called Ultra-low power WSN, minimizes the power consumption by introducing an efficient sleeping mode for long-term battery based applications. In chapter 4, frequent pattern mining techniques are deeply investigated and a frequent episode-mining algorithm is proposed, which can extract both frequent and rare patterns from complex event sequences. In chapter 5, the whole framework of the smart home mining process is introduced, from data acquisition to data mining. The evaluation of the framework with both synthetic data and real life data is described.

Chapter 2. Smart Home and Related Research

The term, “Smart home,” was formally introduced by the American Association of Home Builders in 1984. But the smart home’s history can be tracked back to the early turn of the 20th century, when electrical and telephone wiring were installed in houses. However, the definition of smart home is still unclear, because the expectations of a smart home keep changing with the development in technology. According to Aldrich (2003), a “smart home” can be defined as:

A residence equipped with computing and information technology which anticipates and responds to the needs of the occupants, working to promote their comfort, convenience, security and entertainment through the management of technology within the home and connections to the world beyond.

2.1. Evolution of the Smart Home Concept

Domestic technologies experienced several transitions, and finally lead to the idea of the smart home. The smart home has had several different names in history, such as **Intelligent Home**, **Home Automation**, etc. Recently, more and more smart devices have come into daily life. The name “Smart Home” is used more widely than the others.

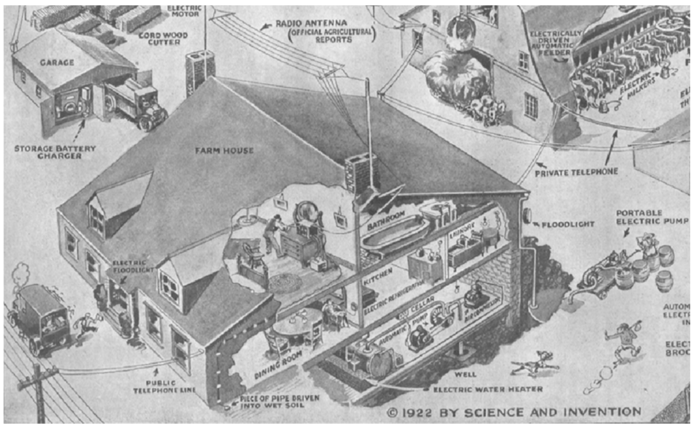


Figure 2-1 A vision of “farm house of tomorrow” published by Science and Invention in 1922.
Source: Retrieved Jun 5, 2013, from http://paleofuture.gizmodo.com/how-the-1920s-thought-electricity-would-transform-farms-510917940

- **The invention of the home appliance**

  In the first 20 years of the 20th century, the vacuum cleaner, dryer, washing machine, iron, and toaster were invented. Although they have nothing to do with being “smart,” they still help to reduce the housework load. About 85% of American homes had electricity by the end of the 1920s (Figure 2-1).

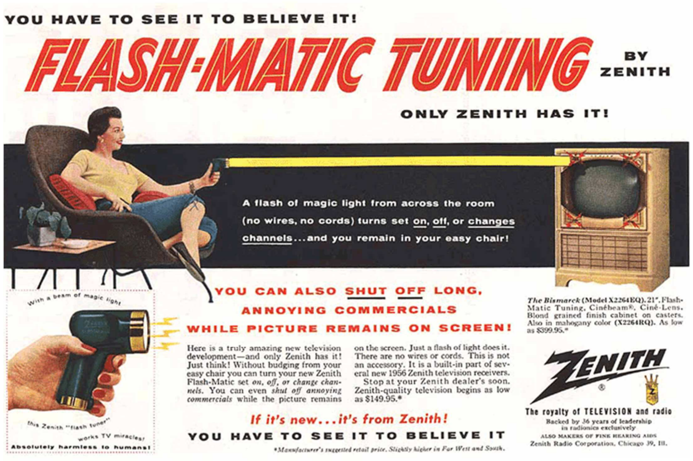

YOU HAVE TO SEE IT TO BELIEVE IT!
FLASH-MATIC TUNING BY ZENITH
ONLY ZENITH HAS IT!
A flash of magic light from across the room (no wires, no cords) turns set on, off, or changes channels...and you remain in your easy chair!
YOU CAN ALSO SHUT OFF LONG, ANNOYING COMMERCIALS WHILE PICTURE REMAINS ON SCREEN!
With a beam of magic light
this Zenith "flash tuner" works TV miracles!
Absolutely harmless to humans!
Here is a truly amazing new television development—and only Zenith has it! Just think! Without budging from your easy chair you can turn your new Zenith Flash-Matic set on, off, or change channels. You can even shut off annoying commercials while the picture remains on the screen. Just a flash of light does it. There are no wires or cords. This is not an accessory. It is a built-in part of several new 1956 Zenith television receivers.
Stop at your Zenith dealer's soon. Zenith-quality television begins as low as $149.95.*
If it's new...it's from Zenith!
YOU HAVE TO SEE IT TO BELIEVE IT
*Manufacturer's suggested retail price. Slightly higher in Far West and South.
The Bismarck (Model X2264EQ). 21". Flash-Matic Tuning. Cinébeam®. Ciné-Lens. Blond grained finish cabinet on casters. Also in mahogany color (X2264RQ). As low as $399.95.*
ZENITH
The royalty of TELEVISION and radio
Backed by 36 years of leadership in radionics exclusively
ALSO MAKERS OF FINE HEARING AIDS
Zenith Radio Corporation, Chicago 39, Ill.

Figure 2-2 The first wireless remote control for televisions by Zenith in 1955
Source: Retrieved May 3, 2015, from http://www.tvhistory.tv/Remote%20Controls.htm

- **Appliance automation**

  Domestic technology took a giant leap in the 1950s with the improvement of remote control devices, air conditioning, television sets (Figure 2-2), and advanced kitchen appliances. General Electric developed a series of firsts that laid the foundation for the contemporary smart home: portable automatic dishwashers in 1954, self-cleaning ovens in 1963, and digital alarm/music clock radios and over-the-range microwave ovens in 1978.

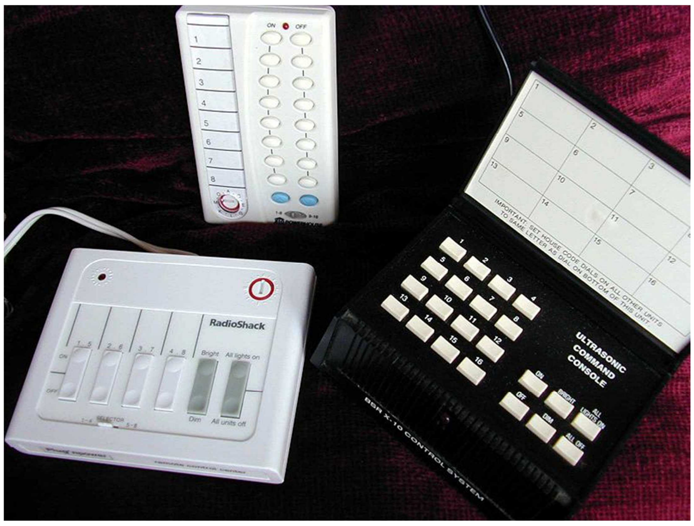


Figure 2-3 X10's home automation kit in 1990s
Source: Retrieved May 3, 2015, from https://en.wikipedia.org/wiki/X10_(industry_standard)

- **Home automation**

  At the end of last century, the Personal Computer (PC) and the Internet came to everyday life. Home automation also began to increase in popularity. With X10 products (Figure 2-3), people were able to control their home appliances through a central controller or computer interface.

Figure 2-4 Nest thermostat in 2012
Source: Retrieved May 3, 2015, http://securityaffairs.co/wordpress/34576/hacking/hacking-nest-home-networks.html

- **Internet of Things**

  With the fast growing internet application and mobile technology in the new century. **Internet of Things (IoT)** has become a new trend. More connectivity and accessibility are provided on smart devices. They can be easily connected to a smart hub or directly accessed through smart phones or tablets (Figure 2-4). The increasing connectivity makes the building of smart home systems much easier than before.

The evolution of these technologies shows that **connectivity** and **adaptability** are two key features of the smart home. With connectivity, people are allowed to connect to home appliances remotely when they are away, get information from the outside world without stepping out the door, or centralize all the available data from their smart devices. While adaptability can make some daily routines operated by the home appliances without additional instructions; Device can adapt to the environment without preprogrammed rules. An appliance is called "smart" if it provides at least one of these two features. For example, a smart bulb provides connection to the home Wi-Fi network and can be controlled by the app on a smartphone. A smart robotic vacuum cleaner can clean the floor automatically without preprograming.

There are two different approaches to setting up a smart home with current smart home products. The first way is the **integrated approach**, or **centralized** approach (Figure 2-5). Home system integrators help their customers install a home automation system, like Control4, Crestron, Savant, AMX, etc., when they construct or renovate their house. It has a systematic structure and all devices are connected to a central controller. The second is the **DIY approach**, or **decentralized** approach. Customers buy the smart devices that can connect to the home network, such as a smart bulb, smart lock, etc., and control these devices from the apps that are provided together with the devices on a smart phone or tablet. In this approach, all smart devices are interconnected and independent. Both of these approaches have pros and cons. With the integrated approach, the customer can have a full control of the smart home. However, the upfront investment is still high. And if the system only uses its own communication protocol, then it might be incompatible with products for other producers. The DIY approach is much more flexible. Customers can choose their own smart products and install them incrementally. However, only partial control of the home appliance can be achieved, and the compatibility among these different products is not ensured.

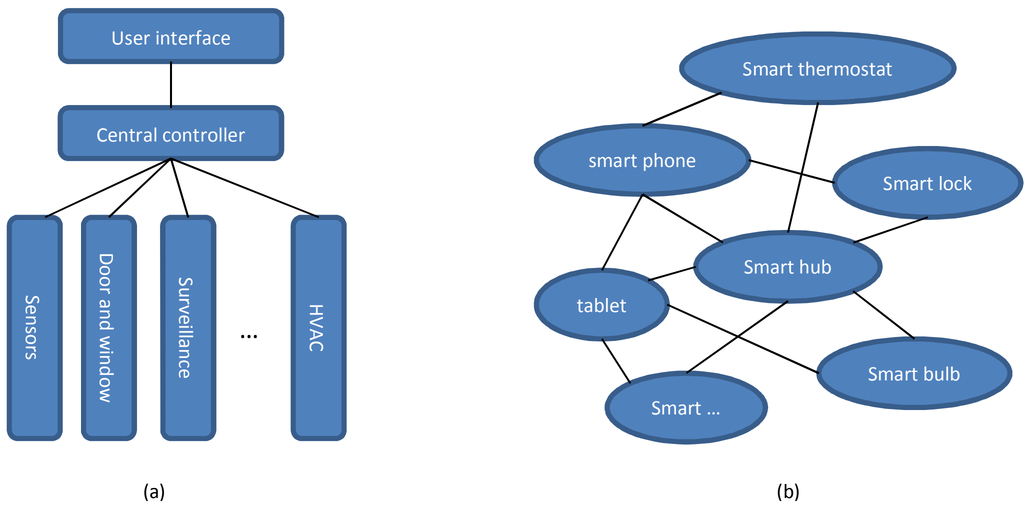


Figure 2-5 Two different approaches to building a smart home: a) integrated, centralized b) DIY, decentralized

2.2. Composition of a Smart Home

A complete smart home consists of three main components: the interface with the physical world, such as appliances, sensors and actuators; the communication component, such as wired or wireless transceivers; and the information processing system, which is for decision making based on the collected information.

- **Appliance, sensor, and actuator**

  Only the appliance with connectivity and/or adaptability will be considered as a smart home device. Most of the home appliance manufacturers and network companies have launched their own smart appliances and platforms, such as Samsung, LG, GE, ABB, Bosch and Cisco. Their smart appliances are ranging from smart coffee machines, smart washing machines and dryers, smart refrigerators, to smart vacuum cleaners. The selling point of these products is that they can report and take instructions from apps on smart phones or tablets.

  Most of the integrated smart home systems have heterogeneous sensors installed to monitor the environment and occupants. In some applications and research, sensor data are collected for more sophisticated usages, such as context awareness, activity recognition, and prediction. There are thousands of different sensors available and different ways to classify them. In this research, sensors are divided into two groups: **ambient sensor** and **event sensor**. Ambient sensors measure the parameters of the environment, such as air and water temperature, humidity, and light and sound intensity, constantly or intermittently. The event sensor, such as passive infrared sensor (PIR), tilt sensor, smoke alarm, or simple switch, can be triggered by occupants when they are doing their daily activities.

In some applications that need direct monitoring of the occupant, wearable sensors, such as an accelerometer and rotation sensor, are commonly used for recognizing activities that are primarily defined by ambulatory movements (such as walking, running, sitting, standing, lying down, and falling). More recently researchers are exploring smart phones equipped with accelerometers and gyroscopes to recognize ambulatory movements and gesture patterns (Krishnan and Cook 2014). With EEG, ECG, EMG, blood pressure sensor, and pulse oximetry sensors, even physiological conditions of the occupant can be monitored.

An actuator is a device that can change its state or the state of the appliance it is attached to. In a smart home, an actuator could be a step motor that can turn a heater up/down or a relay that can turn on/off the ventilation system.

- **Communication**

  Smart homes need some communication mechanism for exchanging data among devices. Such communication can be implemented either using wired or wireless communication system. Bus-line and power-line based communications are available wired solutions. However, the wireless system provides more flexibility in terms of the deployment and network topology. **Local Area Network (LAN)** and **Personal Area Network (PAN)** are commonly used for indoor communication. Some wearable devices will also utilize the **Body Area Sensor Network (BASN)**. More detailed information will be provided in Chapter 3.

- **Information processing**

  Information processing provides the functionalities for managing and analyzing the collected data, and controls appliances and actuators according to the input. There are mainly two approaches to processing the data: **centralized** and **distributed**. In the centralized approach, all data are aggregated to a central controller with data storage and processing power. It could be a gateway node, sink node, or hub. Most of the processing and controlling tasks are done on this central controller. In the distributed approach (Sun, Yu et al. 2013), the calculation power is separate on each device. Each device has a certain level of autonomy to react on the input data. The centralized approach can provide more complex functions, such as data mining, while the distributed approach has better scalability and less data transmission. But, since more calculation will lead to more power consumption on each device, it is not ideal for battery-based applications.

2.3. Contemporary Research Topics

- **Wireless network architecture**

Although there are already some wireless communication systems designed for smart home applications, such as ZigBee, Z-wave, etc., the architecture of the network, network configuration, communication protocol, and deployment still need be studied for different circumstances.

- **Context-aware**

  In the ubiquitous computing scenarios, context-aware refers to devices that must be imbued with an inherent consciousness about their current location and surrounding environment (Cook, Augusto et al. 2009). In the smart home system, context-aware means that the smart home can recognize occupants' daily lives, or it can predict the location of the occupants (Roy, Das Bhaumik et al. 2003), or it can estimate the power consumption of the next time period, or it can detect abnormal behaviors. Since, it is impossible for the system designer to envision all possible contexts forehead; such functionalities largely rely on data mining techniques. More detail will be described in section 2.5.

2.4. Communication Technologies for Smart Home

There are several communication technologies designed for, or have been adapted to, smart home applications. Power-line and bus-line communication are wired solutions. There are also some low-rate wireless personal area networks that could be used. None of these technologies have dominated smart home communication. They all have their pros and cons.

2.4.1. Power-Line Communication

Power-Line Communication (PLC) systems are made of devices that can be connected directly into the main power supply. PLC systems operate by impressing a modulated carrier signal on the wiring system. So, devices can use the normal wiring to send data to devices to activate or deactivate them. There are two categories of PLC: narrowband and broadband PLC. Narrowband PLC offers low data rates and is typically used for home automation and metering purposes. Broadband PLC is popularly used for home networking (Dange and Gondi 2011). There are several important players in the PLC field, such as **Konnex**, **X10**, **CEBus**, and **LONWorks**.

- **Features**

  The advantage of the PLC is that it can build on the existing power-line system, such that no additional wiring or intrusive installation is required. It doesn't interfere with other radio communication devices, and it doesn't need an extra power source for the communication device(Mainardi and Bonfè 2008).

- **Limits**

  Since the power wiring system was originally intended for transmission of AC power, the power wire circuits have only a limited ability to carry higher frequencies. The propagation problem and electrical noise (vacuum cleaners, light dimmers, kitchen appliances and drills) are some of the limiting factors for power line communication. Due to the presence of numerous elements on a power-line network, data attenuation is likely to be an issue.

2.4.2. Bus line Communication

Bus-line uses a separate 12-volt cable to transmit data to devices, which runs in parallel to the main cable. The bus-line device can be configured to adhere to stricter operational parameters and, therefore, systems that are more complex can be envisaged. There are several bus-line protocols, such as the **European Home Systems Protocol (EHS)**, **BatiBUS**, and the **European Installation Bus (EIB or Instabus)**, while **KNX** is the successor to, and convergence of, three previous standards (Dewsbury, Clarke et al. 2002).

- **Features**

  Bus-line has been proven to be the most effective and reliable form of communication for the smart home. It was developed for large buildings, such as offices and factories, and, therefore, uses high quality components that have been tested rigorously, which is ideal for use in high dependency systems.

- **Limits**

  This high specification for products is also reflected in the price of items that are expensive and not always easy to obtain, due to the small number of providers for the technology. (Dewsbury, Clarke et al. 2002).

2.4.3. Wireless Communication Protocols

Wireless communication network usually rely on the radio frequency. Infrared is also used in some cases, but needs a line of sight. Wireless systems are becoming increasingly more popular with users as there are no wires so no modification to the home is required for installation. There is a standard called **IEEE 802.15.4** that offers the fundamental lower protocol layers of a type of **Wireless Personal Area Network (WPAN),** which focuses on low-cost, low-speed ubiquitous communication between devices. However, not all protocols used in smart homes follow this standard. There are several well-known wireless protocols for the smart home environment, such as **ZigBee**, **6LoWPAN**, **DigiMesh**, **Wi-Fi**, **Bluetooth LE**, **Z-wave**, and **EnOcean**.

2.4.3.1. ZigBee

ZigBee is a wireless networking technology developed by the ZigBee Alliance for low-data rate and short-range applications based on IEEE 802.15.4.

- **Features**

  ZigBee defines three device roles: **coordinator**, **router,** and **end device**. Coordinator is responsible for forming the network, handing out addresses, and managing the other functions that define the network, secure it, and keep it healthy. Each network must be formed by one and only one coordinator. A router is a full-featured ZigBee node. It can join existing networks, send information, receive information, and route information. They must be turned on all the time. End devices are essentially stripped-down versions of a router. They can only join networks, send and receive information, so they can use less expensive hardware and can power themselves down intermittently, saving energy by going temporarily into a nonresponsive sleep mode. End devices always need a router or the coordinator to be their parent device. The parent helps end devices join the network, and stores messages for them when they are asleep. ZigBee can form different network topologies, such as pair, star, mesh, and cluster tree (Faludi 2010).

- **Limits**

  ZigBee has several versions and variations, such as ZigBee 2004, ZigBee 2006, ZigBee 2007, and ZigBee PRO. However, it is reported that they are not compatible with each other (Rovsing and Toftegaard). If two vendors' products are based on different versions of ZigBee, their products probably won't work together. Although ZigBee is designed for low-power applications, only the end device can go into the sleep mode. Both coordinator and router still need to rely on an external power source.

2.4.3.2. 6LoWPAN

The idea of 6LoWPAN, or **IPv6 over Low powered Wireless Personal Area Network**, is to offer an IPv6-based solution for the critical embedded wireless requirement of high reliability and adaptability and long lifetime on limited energy and within highly constrained processing resources to minimize cost. 6LoWPAN can be regarded as the IPv6 version ZigBee.

- **Features**

  Most WSN networks cannot be easily integrated with the Internet since the protocols based on IEEE 802.15.4 are not compliant with the IP. Therefore, sensors cannot easily communicate with web-based devices, servers, or browsers. Instead, gateways are required to collect the information from the WSN and communicate with the Internet (Akyildiz and Vuran 2010). 6LoWPAN uses the same RF-chips and low-level protocols as ZigBee, and it has similar specifications except the overhead of using IP, which reduces throughput.(Rovsing, Larsen et al. 2011)

- **Limits**

  The 6LoWPAN protocol is based on IPv6 and operates in a fully asynchronous way. It adopts a mesh topology and uses a routing algorithm, which does not take care of the sleeping node, thus requiring approaches such as low-power listening for energy saving purposes (Tascano, 2012). The greatest challenge to 6LoWPAN is the lack of applications that utilize 6LoWPAN, because it require extensive training as it is complicated to work with and requires extensive knowledge of IPv6 protocol (LSR 2015).

2.4.3.3. XBee DigiMesh

XBee modules leverage multiple types of wireless protocols, which are suitable for all sorts of different network architectures. Open Standards include ZigBee, 802.15.4, and Wi-Fi. Digi has also developed a proprietary protocol called DigiMesh, a mesh networking protocol that reduces some of the complexity of the ZigBee protocol.

- **Features**

Unlike ZigBee, which uses hierarchical based network architecture, DigiMesh uses peer-to-peer architecture, where nodes are not subjected to obligatory relationships as in parent-child constraints and all the nodes are identical (Digi 2008). The route discovery function enables the network to discover the route without maintaining a routing map. Node association or dissociation does not affect the network performance. Low power sleep modes and synchronized wake up are supported by this protocol (Qandour, Habibi et al. 2012).

- **Limits**

  DigiMesh is efficient for the network whose topology or nodes size will frequently be changed. However, in the smart home, the network does not always change. The route discovery will cause more power consumption than a fixed routing table solution.

2.4.3.4. Wi-Fi

Wi-Fi is the chosen communication standard for multimedia applications in the home. However, it is also used to provide access to the home automation system from Wi-Fi enabled devices as an alternative to the ZigBee.

- **Features**

  Wi-Fi implements the IEEE 802.11 standard and offers wireless networking through the use of radio frequency. There are different versions of this protocol. The dominant protocol in use today is IEEE 802.11g, which operates in the unlicensed 2.4 GHz band and provides a maximum raw data rate of 54 Mbps (Gill, Yang et al. 2009). Homes increasingly covered with Wi-Fi networks and Wi-Fi enabled devices, such as smart phones and tablets, are everywhere in daily life. Moreover, the high data rate nature of Wi-Fi allows for greater flexibility in interface design.

- **Limits**

  Wi-Fi is not originally designed for low power applications. The high data rate is redundant for most WSN tasks. Wi-Fi only works with an external power source or with a battery frequently recharged.

2.4.3.5. Bluetooth Low Energy

Bluetooth Low Energy (BLE) is an emerging wireless technology developed by the Bluetooth Special Interest Group (SIG) for short-range communication.

- **Features**

  BLE constitutes a single-hop solution applicable to a different space of use cases in areas such as healthcare, consumer electronics, smart energy and security. It follows a star topology. BLE defines two device roles at the Link Layer for a created connection: the master and the slave. A master can manage multiple simultaneous connections with different slaves, whereas each slave can only be connected to one master. In order to save energy, slaves are in sleep mode by default and wake up periodically to listen for possible packet receptions from the master. BLE may benefit from the widespread use of Bluetooth technology, since BLE easily integrates into classic Bluetooth circuitry, and hence it is likely that future Bluetooth devices will be dual-mode devices. Cambridge Silicon Radio has demonstrated a mesh network for Bluetooth, and the Bluetooth SIG is working on IPv6 support and longer range.

- **Limits**

  The range of Bluetooth is very limited. Moreover, it only supports single-hop. In home environments the coverage is not enough. Therefore, it requires a hybrid topology where some client nodes act as server nodes for other star networks. In Bluetooth-specific terminology, this is called "scatter net", which yields high network complexity in real deployments. For instance, BLE is essentially asynchronous, such that this hybrid topology (mix of star and mesh) causes increased interference and increased power consumption, even inside a single network (iotee 2015).

2.4.3.6. Z-Wave

Z-Wave is a proprietary technology developed by the private company Zensys, which is the only supplier of the chips implementing the physical layer.

- **Features**

  The radio mainly operates in the 900 MHz ISM bands (868 MHz in Europe, 908 MHz in the United States), which often is an advantage, because the 2.4 GHz RF band is typically subject to significant interference due to 802.11 and 802.15.1 devices. The first generations of Z-Wave chips allow transmission at 9.6 and 40 kb/s data, but the recent Z-Wave 400 series supports the 2.4 GHz band and offers bit rates up to 200 kb/s. The transceivers from Zensys allow up to 30 meters indoor range (100 meters outdoors), and the protocol supports the Mesh network topology, which enables a wider range, however, the address space allows only a maximum of 232 devices in a network. In practice, only main powered devices are cable of routing, so a network of only battery powered devices will be unable to route packets.

- **Limits**

  Z-Wave works smoothly and is easily installed, but Z-Wave products are more expensive; they are around triple the cost of an X10. PC controllers that interface to Z-Wave networks use a standardized serial protocol proprietary to Zensys. In order to program your own custom software, you need to sign an NDA and buy a developers' kit, which is roughly worth $10,000.

2.4.3.7. EnOcean

EnOcean is designed to create an ultra-low-power wireless communication technology that is powered by energy harvesting.

- **Features**

EnOcean is not very complex, feature rich, or capable to handle ad-hoc network topologies like other wireless communication protocols. Instead, it is rather simple and tailored in every point to be energy efficient. But, EnOcean also had to give up established approaches in wireless communication that increase communication reliability, such as message acknowledgments or regulated medium access. This makes the protocol basically less robust to the other eminent issues of wireless communication, which are message collisions and transmission errors. EnOcean addresses these issues by using very short messages that reduce the probability of message collisions and by repeating messages several times. This caused EnOcean to propagate their technology also as a protocol with a very low collision probability, indicating a high reliability (Anders 2011).

- **Limits**

  EnOcean sacrifices the complexity and bandwidth of the network for power efficiency. Its network is good for simple applications, such as controlling a light with a wireless switch. Otherwise it needs to rely on the other building backbone such as the Lonworks, KNX to connect its central unit in different area together.

Protocol	ZigBee	6lowpan	DigiMesh	Wi-Fi	Bluetooth LE	Z-wave	EnOcean
Frequency (Hz)	868/915 /2.4G	2.4G	2.4G	2.4G	2.4G	868/915M	314/868/ 902/928 M
Range (Meter)	10~100	10~100	30~90	20~100	10	30~100	30~200
Data rate (kbps)	250	250	250	11000~ 54000	2100	40	125
Max. Node	255~ 65535	100	65535	~50	8~255	232	>4000
Network topology	Star, Mesh, Cluster Tree	Star, Mesh, Cluster Tree	Star, Tree, P2P, Mesh	Star, Tree, P2P, Mesh	P2P, Star	Star, Tree, P2P, Mesh	P2P, Star
Standard (IEEE)	802.15.4	802.15.4	802.15.4	802.11	802.15.1	proprietary	proprietary
Power consumption (mA)	20~35	20~35	45	220+	15~20	30~40	5
Features	Low energy mesh	IPv6 Support	homogenous network	High speed	Low energy	Low energy mesh	Self-powered

Table 2-1 a comparison of different wireless protocols

2.4.3.8. Comparison of Different Wireless Protocols

In Table 2-1, it is shown that the major differences are in the network topology and power consumption.

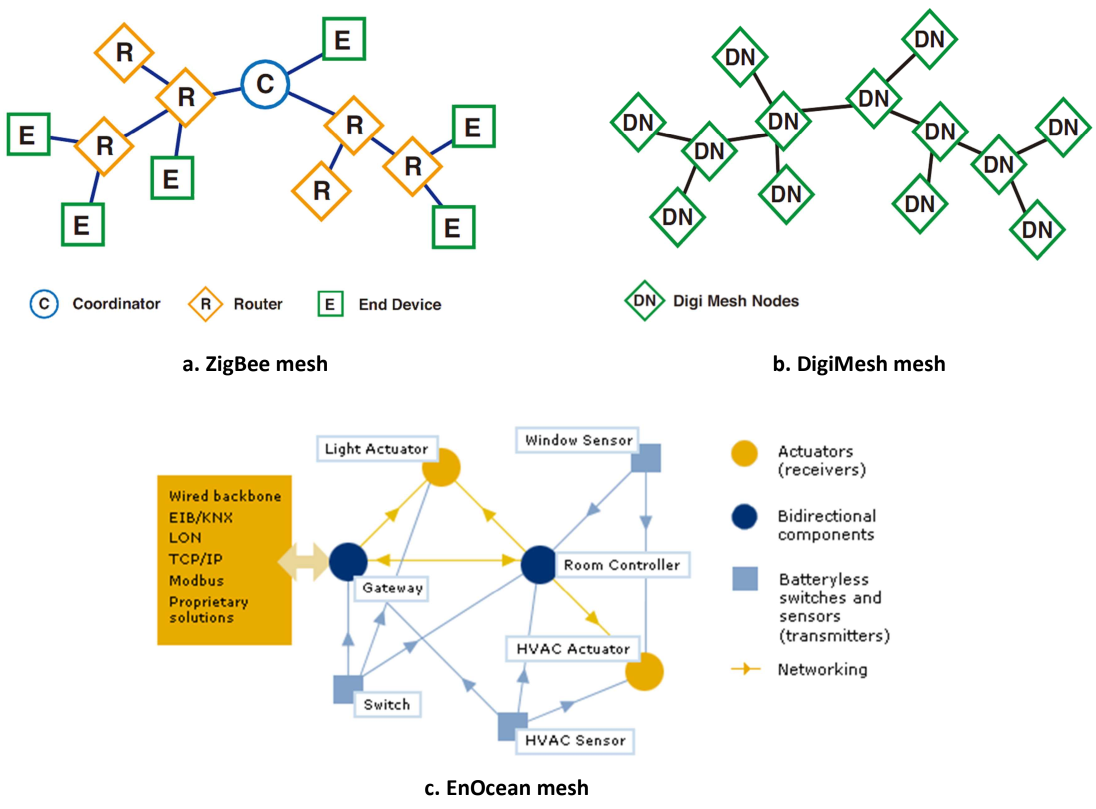


Figure 2-6 Different implementations of mesh network
Source: (Digi 2008) and {Anders, 2011 #2864}

- **Network topology**

  Network topology not only decides how the nodes are interconnected but also the power consumption and reliability of the wireless communication. Although both ZigBee and DigiMesh support mesh network based on IEEE802.15.4, their implementations are different. ZigBee network is a heterogeneous network, which has three different types of nodes, while DigiMesh is a homogeneous network, which has only one kind of node. DigiMesh is easier to setup, has more flexibility to expand the network, and is especially suitable in environments where routers may come and go due to interference or damage. On the other side, ZigBee has reduced function end nodes, which can save more energy if the layout of the network is well planned. EnOcean doesn’t support any mesh network function, but it is also possible to do signal repeating or simple mesh networking by manually connecting the bidirectional components (Figure 2-6).

- **Power consumption**

  Power consumption is a very crucial aspect of the wireless network. There is always a tradeoff between power consumption, sending distance, and functionalities. EnOcean has the best power consumption control. As a consequence, it can only provide very simply functionalities. In contrast, Wi-Fi has the highest bandwidth, but its power consumption is not affordable for most battery based applications. ZigBee, 6lowpan, Digimesh, and Z-wave manage to reach a balance, but none of them performs much better than their competitors. For example, 6lowpan added the support for IPv6 protocol on ZigBee, but it can't handle as many nodes in one network as ZigBee.

2.5. Machine Learning for Smart Home

As mentioned before, the "smart" in smart home is more about context awareness in current research fields. It's an **Artificial Intelligence (AI)** topic. **Machine Learning (ML)** algorithms can give computers the capability to learn without being explicitly programmed (Simon 2013). **Data Mining** utilizes ML and some other algorithms to extract valuable information from apparently unstructured data. The ML problems can be divided into several different subgroups such as **classification**, **clustering**, **frequent pattern mining**, **structured prediction**, etc. Each problem has corresponding algorithms. In order to let the smart home discover and recognize the patterns in occupants' daily lives, a lot of ML algorithms have been tested. In this section, some commonly used algorithms will be investigated. The main features of these algorithms and how they are adapted to smart home systems will be discussed.

2.5.1. Classification

Classification is learning a function that maps (classifies) a data item into one of several predefined classes (Weiss and Kulikowski 1991). In smart home research, algorithms of this group are usually used for recognizing high level activity based on low level sensor data.

2.5.1.1. Artificial Neural Network

2.5.1.1.1. Introduction of ANNs

Based on Warren McCulloch and Donald Hebb's discoveries in neuron structure and function (McCulloch and Pitts 1943) (Hebb 1949), artificial neural networks are originally developed to mimic the basic biological neural system. ANNs are composed of a number of interconnected simple processing elements called neurons or nodes (Figure 2-7). Each node receives an input signal from other nodes or external stimuli, processes it locally through an activation or transfer function, and produces a transformed output signal to other nodes or external outputs. (Reilly and Cooper 1990). ANNs learn relationships between the inputs and outputs from examples presented to the networks **without any prior knowledge** about the underlying relationships that exist within the data sets (Begg and Hassan 2006). In this way it is able to synthesize **implicitly** a certain model of the problem. In other words, the neural network builds up alone an algorithm to solve a problem. The capacity of the neural network to solve complex practical problems using a multitude of samples gives them a highly large potential of applicability (Vintan, Gellert et al. 2004).

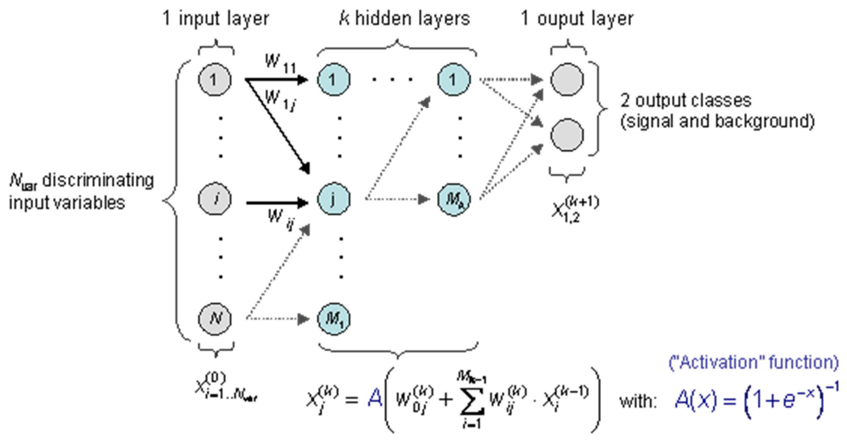

1 input layer
k hidden layers
1 ouput layer
2 output classes (signal and background)
N_{var} discriminating input variables
w_{11}
w_{1j}
w_{ij}
$x^{(0)}_{i=1..N_{var}}$
$x^{(k+1)}_{1,2}$
$x_j^{(k)} = A\left(w_{0j}^{(k)} + \sum_{i=1}^{M_{k-1}} w_{ij}^{(k)} \cdot x_i^{(k-1)}\right)$ with: $A(x) = \left(1+e^{-x}\right)^{-1}$
("Activation" function)

Figure 2-7 Multi-layer perceptron with k hidden layers, N input, 2 output
Source: Retrieved Jan 3, 2014, from http://tmva.sourceforge.net/

2.5.1.1.2. Variations of ANN

The first single-layer perceptron, a type of linear classifier, was invented in 1957 by Frank Rosenblatt (Rosenblatt 1957). Since then, a lot of different variations of ANNs with different features have been developed.

Generally, An ANN can be characterized in three different aspects. First, the way the neurons are **connected** to each other; second, the method that determine the connection strengths or weights, i.e. the **training algorithm**; third, the **activation function** that determine the mapping relationship between the input and output of each node. Based on the way they learn, all artificial neural networks can be divided into two learning categories - supervised and unsupervised. Meanwhile, according to the structure and connection of the network, they can also be divided into feed-forward networks and feedback networks.

In 1959, Bernard Widrow and Marcian Hoff of Stanford developed models called ADALINE (ADAptive LINear Elements) based on the single-layer perceptron, which takes multi input and single output. By combining a number of ADALINEs, MADALINE (Multiple Adaline) is created to help counter the problem of non-linear separability (Widrow and Hoff 1960). In 1972, Kohonen used matrix mathematics to describe the neurons, which are supposed to activate a set of outputs instead of just one ADALINE (Kohonen 1972). In 1982, Hopfield network is invented by John Hopfield, which provided a model for understanding human memory (Hopfield 1982). In the same year, Reilly and Cooper built a multiple layers neural network, with each layer using a different problem-solving strategy (Reilly, Cooper et al. 1982). In the 1980s, back propagation, a supervised learning method which was first described as a multi-stage dynamic system optimization method, was introduced in training multilayer neural networks by some researchers (Rumelhart, Hintont et al. 1986). It led to a renewed interest in neural networks. Different kinds of neural networks, includes Boltzmann machines, competitive learning models, and adaptive resonance theory models, have been invented. In 2006, Hinton proposed a new type of ANNs, called AutoEncoder, which promotes research into the Deep Learning stage (Hinton, Osindero et al. 2006).

2.5.1.1.3. Applications in Smart Home Research

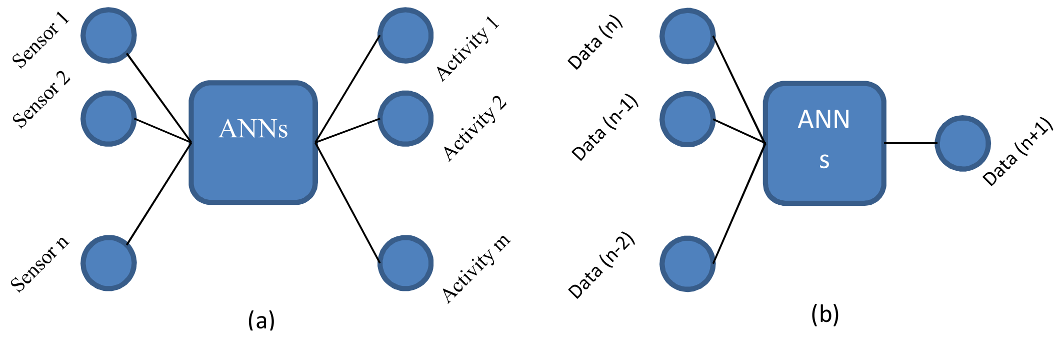


Figure 2-8 Two paradigms for using ANNs in smart home research

In the smart home realm, researchers also did a lot of experiments to improve the building control system by integrating ANNs, such as the control of home appliances and the prediction of power consumption. ANNs are mainly used in two different ways in smart home research. The first way, ANNs work like a classifier to map the input low level sensor data to high level daily activities, which needs a supervised training process (Figure 2-8 (a)). In the second way, ANNs are fed with historical data, and asked to make predictions on the next step, which could use unsupervised training (Figure 2-8 (b)).

Early in 1994, (Curtiss, Kreider et al. 1994) tried to use a feed-forward network with the delta rule back propagation learning algorithm to predict the load of an HVAC system in the next time period. Sliding window technique was introduced to provide the network not only the latest data, but also a short period of history data. Similarly, Yang (Yang, Yeo et al. 2003) applied a multi-layer perceptron with several hidden layers to determine the optimum start time of HVAC equipment.

A smart home research project was undertaken by Mozer in 1998 (Mozer, Vidmar et al. 1997), named Adaptive Control of Home Environments (ACHE). To anticipate the inhabitant’s needs as well as to conserve energy, a multi-layer perceptron network was used to estimate the probability that the occupant would be home, so the controller can switch on the furnace and preheat the room in advance.

In 2000, a more complex ANN, multilayer recurrent network using the standard back-propagation learning algorithm was applied to model the thermal behavior of the building. This network architecture holds the contents of one of the layers as it existed when the previous pattern was trained. In this way, the network sees the previous knowledge it had about previous inputs. Recurrent neural networks are particularly suitable for prediction of sequences, so they are excellent for time series data as in the present case(Kalogirou and Bojic 2000).

Table 2-2 ANNs application in Smart home

Authors & project name	Application	Network type & size	Input vector	Output	Data source
(Badlani and Bhanot 2011)	Control the home appliance with temperature and humidity	Single perceptron	2, Temperature , humidity	4, On/off of fan, A/C, bulb, table lamp	Synthetic 150 training examples
(Machado and Mendes 2009) Automatic Light Control in Domotics	Predict the on/off state of the hall light based on the state of other lights	Multi-layer perceptron with one hidden layer: Input: 26; Hidden: 26 ; Output: 1;	26, one for each of the 24 light sources and one for the time attribute and other for the weekday attribute	1,On/off state of the hall light in the instrumented home	a total of 178028 records from which 3592 record are ON state and 174436 are OFF state for the hall light
(Rivera-Illingworth, Callaghan et al. 2005)Activity Detection in AmI Environments	Recognize different high level activities ,such as sleeping, working at computer, eating, and identify abnormal behaviors.	Adaptive Neural Architecture, 3 layers: the input layer, the evolving hidden layer and the output layer, plus a memory layer		8, represent Sleeping, Out of room, Desk work, Reading, Computer work, Listen music, and other activities	Collected by 18 sensors over a period of 4 days. 8 different activities labeled. A total of 471 instances were recorded.
(Vintan, Gellert et al. 2004)Person Movement Prediction	Anticipate a person's next movement in the office	Multi-layer perceptron with one hidden layer	(4~32) + 2,The code of the person and the code of the last rooms visited by that person.	5,code for predicted room	4 people , 100-450 movements per person during the summer 2003 and 1000 movements per person during fall
(Yang, Yeo et al. 2003) predict the optimal start time for heating system	Determine the optimum start time of HVAC equipment	Multi-layer perceptron with several hidden layers	4, room air temperature, varying rate of room air temperature, outdoor air temperature, varying rate of outdoor air temperature	1, start time of the HVAC	Data collected from a thermal Simulation system
(Kalogirou and Bojic 2000) prediction of the energy consumption of a passive solar building	Use ANNs, to model the thermal behavior of the building	A multilayer recurrent architecture using the standard back-propagation learning algorithm has been applied.	5, Season, Insulation, Thickness, Function, Time	1,energy consumption	Data generated by the program ZID have been used for the training of a neural network.
(Mozer, Vidmar et al. 1997) Neurothermostat	Adaptive controller that regulates in-door air temperature in a residence by switching a furnace on	Multi-layer perceptron with one hidden layer	4, Current time of day, day of week, average proportion of time home was occupied	1,estimation of the probability that the occupant will be home.	The real data is opening and closing of outside door of the house with single resident and artificial data
Curtiss, Kreider et al. 1994)	Predict the load at the next timestamp	feed forward network	80, valve actuator control and cooling load 20 time steps, entering air temperature, air flow, hot water temperature rate and Hot water branch flow rate 10 time steps.	Prediction of the load at the next time step.	Collected in the laboratory.

Vintan (Vintan, Gellert et al. 2004) proposed neural prediction techniques to anticipate a person’s next movement. They focused on neural multi-layer perceptron with back-propagation learning. The optimal configuration of the neural network is determined by evaluating movement sequences of real persons within an office building. The simulation results, obtained with one of the pre-trained neural predictors, show accuracy in next location prediction reaching up to 92%.

(Rivera-Illingworth, Callaghan et al. 2005) aimed to distinguish different human behaviors inside a domestic environment such as “sleeping,” “relaxing,” “eating,” etc. Once the “normal” activities are detected, different kinds of abnormal activities can be spotted. They used an Adaptive Neural Architecture derived from the ECoS paradigm proposed by Kasabov. This network can grow dynamically. New nodes can be added to the hidden layer whenever an example is not found to fit in the already existing structure.

(Machado and Mendes 2009) presented a complete autonomous method of using ANN to determine the users’ patterns concerning the usage of the lights with the purpose of predicting its state and act on them automatically. The method presented will not require any users’ direct input by UI. It only uses the patterns acquired from the normal usage of the lights from the users and, therefore, it doesn’t require the task-labeling step.

Most research only focuses on a specific sub topic of Smart Home technology to keep the problem simple, and most of the training data are collected from laboratory, or under special instruction. Some applications already have very clear mathematic models. The ANNs are just used to fit the model. Thus, although some of the research claimed accuracy already higher than 90%, it is still far away from the real application.

2.5.1.1.4. Limits of ANNs

- **Setup process**

  There are several parameters that can significantly affect the final performance of the network. The size of the network, the amount of the hidden layers and nodes, the type of transfer function at each node, and the training algorithm, all require a substantial amount of engineer in designing the network.

- **Black box**

  The mechanism of ANN is generally considered a black box. With the same setup and same training dataset, the outcome performance could be totally different. The researchers can repeat the training until they get a good result, but in real time condition, it is hard to find a proper stopping criterion. Besides that, the size of nodes in the hidden layer is hard to determine. It is partially related to the amount of patterns in the dataset that we don’t know in the real time application.

- **Lack of accuracy**

The algorithm and training process decides that ANNs are not competent for the tasks that require high precision. To keep the performance in generalization and predictive ability, the network will not be trained 100% fit to the training set. Otherwise, the network is over fitting (Gurney and Gurney 1997). Good prediction accuracy could be around 85%. It is enough for some applications that don't need to be very accurate, such as temperature control (Khalid and Omatu 1992), HVAC system control of commercial building (Curtiss, Kreider et al. 1994), and water heater control (Mozer 1998). But, in some other applications, a wrong prediction can be annoying or detrimental if the inhabitant must undo the action executed by the house or repair damage caused by a faulty decision, like the on-off of the lights and cookers.

- **Supervised learning process**

  As a supervised learning algorithm, ANN first needs to be trained with a training dataset before making any predictions. With a clean, labeled dataset preprocessed by researchers, the training process can convergent very quickly and the recognition rate could be high. However, in real application, it is impossible to label the activities for each user manually to create the training dataset. And with the noise in the raw data, the training may never be convergent.

- **The rare pattern**

  The network has to be trained until the output of the error function is lower than some threshold. So, it will tend to fit the high frequency patterns by sacrificing the rare patterns in the training dataset to get a lower global error. However, in real life, a low frequent activity does not mean it is not important.

- **Hard for on-line learning**

  It also has been stated (Tapia, Intille et al. 2004) that adapting neural networks to the continuous changes in an intelligent space might require retraining the whole network, which can be a computationally expensive process especially in cases that require on-line adaptation.

2.5.1.2. Support Vector Machine

2.5.1.2.1. Introduction of SVM

The complete theory of **Support Vector Machine (SVM)** was first introduced by Cortes in 1995 (Cortes and Vapnik 1995). As a competitor of ANN, it also widely used in classification problems. The basic idea of SVM is converting non-linearly separable data into linearly separable data by mapping them into higher dimensional space and finding the **hyper plan** which can separate them with the **maximum margin** (Cristianini 2001) (Figure 2-9). Since the calculation load will be increased largely in the high dimensions, the **kernel trick** is introduced to implicitly compute the dot product of the vectors (the major part of the calculation) in a higher dimension without explicitly transforming vectors to a higher dimension (Kim 2013). In fact, a linear SVM works like a single layer ANN and a nonlinear SVM can be regarded as a single hidden layer neural network

(Zanaty 2012). ANNs try to solve the nonlinear problem with multiple hidden layers (multi perception), while SVMs try to linearly separate the data in a higher dimension.

2.5.1.2.2. Applications in Smart Home Research

Some researchers have reported their applications in smart home research. In the paper (Choi, Shin et al. 2005), linear SVM is utilized to recognize the patterns of appliances used by the occupant in the smart home. The training data consists of six context data from the occupant and smart home and one set of context data from occupant commands. The six context data includes pulse, body temperature, facial expression value, room temperature, time, and occupant location in the smart home. The occupant commands include the on/off of the TV, air conditioner, audio, projector, and Light. In Fleury's project (Fleury, Noury et al. 2009), the author used the percentage of time spent in the different posture (stand, sit, lie) and walking, number of events per class and number of events per microphones, percentage of time in each room and number of events for each PIR sensor, percentage of time the door was in "open" position and predominant state (open/close) in the considered time slot, and differential measure for the last 15 minutes for temperature and hygrometry as selected features to recognize seven different activities: sleeping, resting, dressing, eating, elimination, hygiene, and communication. In Nguyen's work (Quoc Cuong, Dongil et al. 2009), motion of the occupants were recognized with SVM. Four linear-SVMs were used to recognize four different actions: lie down, sit, stand up, and walk. The edge information extracted from black and white images recorded by camera was used as input data.

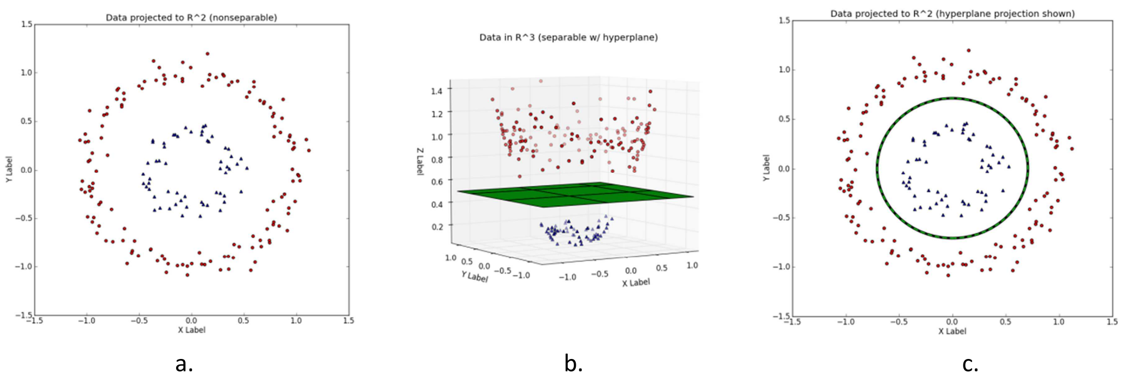


Figure 2-9 a) A dataset in 2D space, not linearly separable. b) The same dataset is transformed into 3D by adding a Z dimension, where $z_i = x_i^2 + y_i^2$, and is able to be divided by a hyper plan. c) hyper plan becomes a non-linear division when transformed back to 2D space.
Source: Retrieved Feb 20, 2015, from http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html

2.5.1.2.3. Limits of SVM

Although the theoretical base of SVM is more solid than ANN, which makes it not only a black box as ANN, it still suffers from some problems. The biggest drawback is in choosing the kernel function. Non-linear SVM needs some prior knowledge of the target to choose a good kernel (Cristianini 2001). Another problem is, unlike ANNs that can have several outputs, SVM can only have one Boolean output. The way to create an n-ary classifier with SVM is to create n SVMs and train each of them one by one.

2.5.2. Clustering

Clustering is a common descriptive task where one seeks to identify a finite set of categories or clusters to describe the data (Jain and Dubes 1988). Most of the clustering algorithms work in the unsupervised manner, such as K-means, DBSCAN, and Self-organizing Map.

2.5.2.1. Self-organizing Map

2.5.2.1.1. Introduction of Self-organizing Map

The **Self-organizing Map (SOM)** algorithm was first presented by Kohonen in 1982 (Kohonen 1982). It is one kind of unsupervised competitive learning neural network. It is an effective software tool for the visualization of high-dimensional data (Kohonen 2001). A SOM is a single layer neural network, where the neurons are set along an n-dimensional grid (Figure 2-10). In most applications, this grid is 2-dimensional and rectangular; some applications use hexagonal grids, and 1, 3, or more dimensional spaces (Bação and Lobo 2004). These nodes are connected with mutual lateral connections. Unlike the connections to the input layer, these lateral connections do not have weights, but are used in updating the weights (Trajanoski 2008). Neighboring cells compete in their activities by means of mutual lateral interactions, and develop adaptively into specific detectors of different signal patterns. In this category, learning is called competitive, unsupervised, or self-organizing (Kohonen 1990). It has been used for clustering, dimensionality reduction, classification, sampling, vector quantization, and DM. With respect to any other classification methods, the main characteristic of the SOM classification is the conservation of the topology (Oja and Kaski 1999).

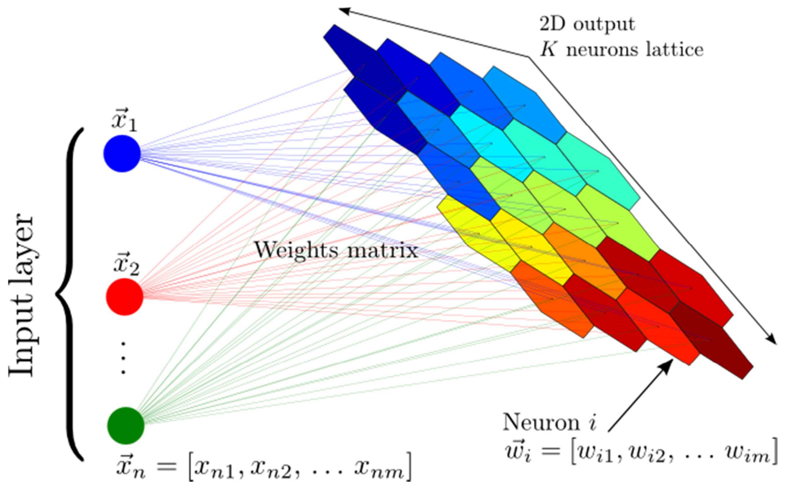

2D output
K neurons lattice
$\vec{x}_1$
$\vec{x}_2$
Input layer
Weights matrix
Neuron i
$\vec{w}_i = [w_{i1}, w_{i2}, \dots w_{im}]$
$\vec{x}_n = [x_{n1}, x_{n2}, \dots x_{nm}]$

Figure 2-10 A schematic representation of a self-organized map. The color of the map encodes the organization of groups of objects with similar properties
Source: Retrieved Jun 12, 2015, from http://www.lcdm.astro.illinois.edu

2.5.2.1.2. Applications in Smart Home Research

In the paper (Mayrhofer, Radi et al. 2003), the author proposed an architecture for recognizing and predicting context by learning form data collected from a user's mobile device. The architecture consists of five steps: sensor data acquisition, feature extraction, classification, labeling, and prediction. SOM is suggested for use in the classification step, which classifies the feature vector extracted from raw data and creates the cluster for assigning the label in the next step. In Zheng's research (Huiru, Haiying et al. 2008), he presented an approach to cluster analysis of human activities of daily living within smart home environments based on a self-adaptive neural network called **Growing Self-Organizing Map (GSOM)**. Different from the normal SOM, the GSOM has the ability to add new nodes into its output layer when new clusters are found. This is a useful feature when the amount of clusters is unknown. In this test, 22 different activities were discovered in the data samples collected from 76 sensors during a 16-day period. The research (Vazquez and Kastner 2011) shows that SOM outperforms other cluster algorithms, like Fuzzy C-means, K-means, K-means with Repeated Bisection, Graph Clustering, and Support Vector Clustering with their sensor data.

2.5.2.1.3. Limits of SOM

SOM is not good at dealing with categorical values, because it is unclear how to calculate the distance with categorical values. Especially when the input data is a combination of numerical and categorical values, it is hard to decide if the on/off of a light creates more distance than a temperature change. Another problem of SOM is that there will be some randomness in its cluster as a result caused by its random initial state.

2.5.3. Structured Prediction

The goal of structured prediction is to build machine learning models that predict relational information that itself has structure, such as being composed of multiple interrelated parts (Nowozin, Gehler et al. 2014). The main application is in fields including computer vision, speech recognition, natural language processing, and computational biology. Hidden Markov Model, Bayesian Network, and Conditional Random Field are algorithms widely used in this field.

2.5.3.1. Hidden Markov Model

2.5.3.1.1. Introduction of Hidden Markov Model

The Hidden Markov Model (HMM) is a generative probabilistic model that's used for generating hidden states from observable data, i.e., to determine the hidden state sequence that corresponds to the observed output sequence (Eunju, Helal et al. 2010). Andrei Markov gave his name to the mathematical theory of Markov processes in the early twentieth century, but it was Baum and his colleagues that developed the theory of HMMs in the 1960s. HMMs have found applications in many areas in signal processing, and in particular speech processing, but have also been applied with success to low level NLP tasks such as part-of-speech tagging, phrase chunking, and extracting target information from documents (Blunsom 2004).

In smart home scenarios, the hidden state sequence is the high level daily activities performed by occupants and the observed sequence is the low-level sensor events. With HMM we are able to model the daily activities sequence by analyzing the sensor events. As showed in Figure 2-11, these ovals represent hidden states (i.e., activities) and rectangles represent observable states (i.e., sensor data). Values on horizontal edges represent transition probabilities a_{ij} between activities. Values on the vertical edges represent the emission probability b_{kl} of the observable state given a particular current hidden state. There are several situations. When the observation sequence and all parameters are given (transition probabilities, emission probability), try to discover the hidden state sequence that was most likely to have produced this given observation sequence, also called Decoding. Another situation is that when only the state sequence and observation sequences are given, we would like to be able to estimate the model parameters that best describe that process. Then a supervised training process is needed. However, in most case, only the observation sequence is provided, to guess a model that may have produced those observations, the unsupervised training algorithm, called Baum-Welch algorithm is needed.

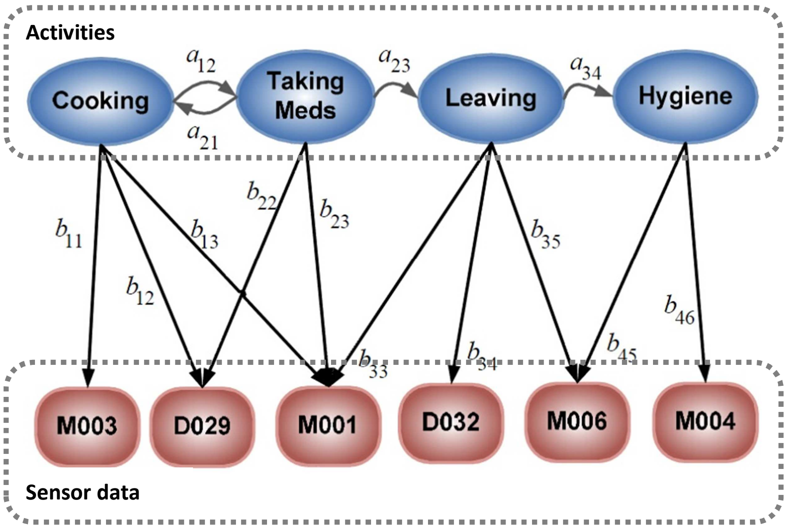


Figure 2-11 A Hidden Markov Model describing for smart home activity recognition
Source: (Rashidi,2011)

2.5.3.1.2. Applications in Smart Home Research

In the paper (Lu, Sookoor et al. 2010), the author tried to train a HMM to control the thermostat. The hidden state is a distribution over the home state: away, active, and sleep. The HMM transits to a new state every five minutes. The observed states are a vector of three features of the sensor data: 1) the time of day at 4-hour granularity, 2) the total number of sensor firings in the time interval, and 3) binary features to indicate presence of front door, bedroom, bathroom, kitchen, and living room sensor firings in the time interval. In Kasteren's project (Kasteren, Noulas et al. 2008), researchers tried to recognize activities, namely: leave house, toileting, showering, sleeping, preparing breakfast, preparing dinner, and preparing a beverage, from sensor readings (14 state-change sensors) in a house with labeled training data set.

2.5.3.1.3. Limits of HMM

Due to their Markovian nature (the conditional probability distribution of future states of the process depends only upon the present state), they do not take into account the sequence of states leading into any given state, i.e., the context information. HMM have the disadvantage of resulting in very complex networks and their learning algorithms have proved difficult to apply to problems involving numerous low-level sensors(Rivera-Illingworth, Callaghan et al. 2005).

2.5.4. Frequent Pattern Mining

Frequent Pattern Mining (FPM) is aimed at finding out the frequent pattern, i.e. itemsets, subsequences, or substructure that appear in a dataset with frequency no less than a user-specified threshold (Han, Cheng et al. 2007). More details will be provided in Chapter 4.

Algorithm	Learning strategy	Data type	Input vector size	Parameters
ANN	Supervised	Numerical & Categorical	fixed	Amount and size of hidden layers, activation function, training algorithm
SVM	Supervised	Numerical	fixed	Size of the input layer, support vector, kernel function
SOM	Unsupervised	Numerical	fixed	Size and topology of the output layer, distance function
FPM	Unsupervised	Categorical	Non-fixed	Min-support, constrain window size
HMM	Supervised or unsupervised	Categorical	fixed	Training algorithm, tolerance, iterations

Table 2-3 Comparisons of different ML algorithms

2.5.5. Comparison

All of these techniques have their own merits and limitations when applied to a smart home. Here we compare the popular algorithm in four aspects:

- **Supervised vs. unsupervised**

  An unsupervised learning algorithm only makes use of the information contained in the training dataset to extract the optimal set of model parameters. Supervised learning methods require that the data are augmented with **priori information**, or **labels**, to guide the learning process. The labels usually consist of input data associated with the expected model output, defined by a subject matter expert.

  Supervised methods have the drawback that they require labeled data for training the recognition algorithm. Annotating human activity data from low-level sensors is a very time consuming and laborious task and can limit the scalability of the system. (Rashidi and Cook 2013). In some applications, like speech recognition, it's not too hard to assign the text contents to the voice. However, in smart home research, since people are performing their daily activities consciously or unconsciously and always interrupted or interleaved with each other, it is really hard to define what activity they are exactly doing even in experimental environment. For example, in Tapia's research (Tapia, Intille et al. 2004), the occupants were asked three questions every 15 minutes, i.e., "what are you doing now ?", "For how long have you been doing this activity?", "Were you doing another activity before the beep?". In another project, the occupant was asked to report the activity he was doing through a

Bluetooth headset, and later processed with speech recognition engine. A more common approach is to record everything by video camera, and researchers manually assign labels depending on the records. However, in real life, all of these approaches are not feasible. Results gained in the experiments are not reusable and scalable due to the variation of the individual's behavior and their environments. (Chen and Khalil 2011).

- **Input data and output data type**

  The data acquisition and preprocessing needs a large amount of effort, even more than the mining process. There are mainly two kinds of data types that might be recorded: **numerical** (discrete and continuous value), **categorical** (like the weather: windy, snowy, sunny, etc.), and **ordinal** type can be regarded as the mixture of those two (like room id: room 1, room 2…). Most of the algorithms don't support all data types. Sometimes these data types are interchangeable, but additional knowledge of the dataset is required. The ANNs can input and output both types with normalization. The frequent pattern mining can only deal with categorical data type, while SOM can only deal with numerical data.

  Another constraint of the input is the input data size. Such as ANN, SVM, and SOM, they only accept a fixed length feature vector. In time series analysis, if the pattern is longer than the input vector, then it can't be detected. However, frequent pattern mining doesn't have this problem. It takes the whole database as input.

- **Consistency**

  Since there is some randomness in the training process, such as the random initial weights in ANN and SOM, the same dataset and settings may lead to different results from time to time. Researchers tend to pick the best result to show their achievement. However, when there is a lot of noise in the training dataset, the correctness of classification of the same algorithm may float in a very large range. Such inconsistency in performance also will be caused by the order of the training dataset. Training with a different ordered training dataset may converge at different convergent points. However, the frequent pattern mining algorithms don't have this kind problem. Once a transaction is set T, a **min_sup** and window constraints are given, the set of frequent itemsets that can be found in T is uniquely determined. Any algorithm should find the same set of frequent itemsets. This property about frequent pattern mining does not hold for many other data mining tasks, e.g., classification or clustering, for which different algorithms may produce very different results (Agrawal, Imieli et al. 1993).

- **Over trained problem**

  The introduced supervised algorithms are running an optimization process, which gradually reduce the error between the output and expected output in the training dataset in a gradient descent manner. Such an approach doesn't ensure the result will get to a globally optimal solution. On the other hand, the error can't be too low on the training dataset, otherwise the algorithm will be over fitting to the training set, and lose the generalization on the other dataset. The error is also related to the noise in the input data. If there is a lot of noise and little pattern in the dataset, the error will remain high. So the control of the training process also needs expertise and a lot of experiments to reach a balance between accuracy and generalization. For the unsupervised algorithm, the parameters of the algorithm will have a large impact on the result. The setting of parameters also needs prior knowledge of the dataset.

2.6. Problems in Existing Smart Homes

Although there are a lot of achievements in smart home products and research, its market share is still negligible when compared to overall consumer electronics. Several surveys and studies have been conducted to address the problems and challenges (Edwards and Grinter 2001, Chan, Estève et al. 2008, Eckl and MacWilliams 2009, Holroyd, Watten et al. 2010, Rovsing, Larsen et al. 2011, hgfdSuryadevara and Mukhopadhyay 2015). From the technical side, the existing problem in smart home systems can be summarized into these points:

- **Interoperability**

  Smart homes must evolve and adapt to changing preferences, demands, and needs. Thus, the system should be able to easily assimilate new devices gradually added to the smart home network, and its devices need to be able to communicate with each other. Since none of the existing protocols is widely accepted as the standard for smart homes, the market is fragmented. Many competing manufacturers are developing disparate smart home systems using different protocols. It’s easy to integrate the same series of devices made by the same vendor. However, life cycles of the home appliances are different and new technologies keep emerging, people are more likely to buy the new appliance from different vendors. Then the compatibility will be an issue. Although there are more and more smart devices starting to support the commonly used Wi-Fi connections, they still need their own control interface or App.

- **Power consumption and supply**

  The devices based on power-line and bus-line communication don’t have a power issue, but the installation will be intrusive. In contrast, the wireless solution makes the installation less intrusive. However, for long-term application, power supply is still a problem. If a wireless sensor still relies on cable power supply, it will lose its wireless advantages. For the battery-based solution, although there are some low energy and short distance wireless transceiver, the actual power consumption also needs to count in consumption of the other sensors on board and the sending frequency. The available indoor energy harvesting techniques are still not enough for complex network structure.

- **Learning curve**

  The installation, maintenance, and upgrade of centralized smart home systems need expertise. Users also need time to get familiar with the user interface and configurations. The decentralized smart home system will be easier for normal customs, because smart phone and tablet apps are commonly used these days. But for elder people who are not so keen on new technologies and hard to adapt to complex interfaces, user-friendliness is still a concern (Demiris, Rantz et al. 2004).

- **Rate of return**

  Fully integrated custom systems are expensive and often require a consultant to install them, and structural changes to the home, both costs tacked on to the price of the system itself. But the smart home won't take an immediate effect. Its benefits will be revealed gradually in costumers' daily life. People are tending to scarify a bit of power consumption and comfort, instead of installing the entire system in their house. However, smart devices are gaining more attention recently, such as smart door locker, smart hub. They have very specific features and limited functionalities. With affordable prices, people just pay for the service they actually need.

- **Security**

  Another serious concern is the potential for criminals to hack into a smart home system. This has serious implications as smart home systems generally integrate home security systems in addition to others. Although most of the wireless smart home networks encrypt their messages, it is still a concern, especially when cameras and microphones are integrated.

- **Lack of intelligence**

  Most of smart home systems are still based on user specified thresholds and triggers to automate the home appliance. However, the needs of the inhabitants still changing from time to time. Unless the smart home is able to adapt to its inhabitants, it is far away from "smart".

In the academic field, researchers are trying to address the problems separately. The WSN developers are trying to give the network more features, regardless what the types of data and measuring frequency are needed. The ML researchers focus on improving the accuracy of the algorithm, without telling if the training data is available in the real life condition. Thus, an integrated solution is needed to combine these technologies in a more practical and efficient way.

Chapter 3. The Design of WSN for SPM

Because of the limited resources on the WSN node, one network can't have all features integrated. There is always a tradeoff between the functionally and power consumption. In this research, two different WSNs are proposed for different applications and deployment environments. The first one is called **Self-organizing WSN** for short term monitoring, which is a mesh network easy for deployment and maintenance. The second one is **Ultra-low power WSN** for long term monitoring, which is a tree cluster network based on battery power supply. In this chapter, some fundamental WSN knowledge will be briefly introduced, and then the design of the proposed networks will be introduced.

3.1. Introduction of Wireless Sensor Networks

A wireless sensor network is a network of devices denoted as **nodes** that can sense the environment and transmit the information gathered from the monitored field through **wireless links**; the data is forwarded, possibly via **multiple hops** relaying, to a **sink** locally, or a **gateway** which provides the connection to the outside network (Malfa 2010)(Figure 3-1).

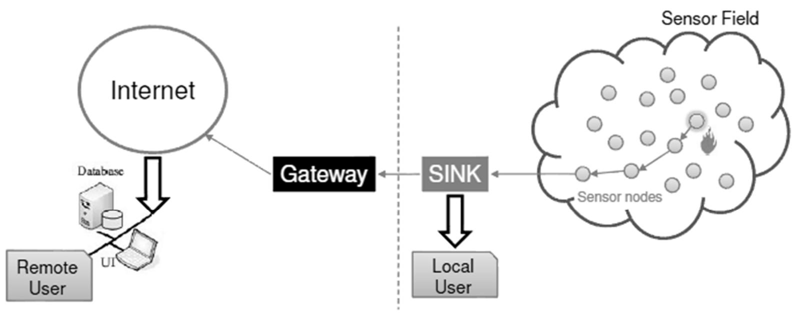


Figure 3-1 Wireless sensor network,
Source: (Malfa, 2010)

Initial research into **wireless sensor networks** was mainly motivated by military applications, like the Distributed Sensor Networks (DSN) programs around 1980 and the Sensor Information Technology (SensIT) programs (Kumar and Shepherd 2001). Recently, WSN also find its applications in physiological monitoring, environmental monitoring (air, water, soil chemistry), condition based maintenance, **smart spaces**, precision agriculture, transportation, factory instrumentation, and inventory tracking (Estrin, Girod et al. 2001, Chee-Yee and Kumar 2003).

The **specific application** and the **deployment environment** play the key roles in determining the design of the WSN. Besides design issues in traditional networks, a WSN has its own design space and resource constraints, such as deployment, mobility, cost, size, resources, energy efficiency, heterogeneity, communication modality, infrastructure, network topology, scalability, coverage, connectivity, fault tolerance, and other QoS (Quality of Service) requirements(Lewis 2004, Romer and Mattern 2004).

3.1.1. Components

The primary unit of WSN is sensor node. The terms sensor node, wireless node (WN), Smart Dust, mote**,** and COTS (commercial off-the-shelf) mote are used somewhat interchangeably in the industry (Kazem Sohraby, Daniel Minoli et al. 2007). A basic sensor node comprises five main components: **controller**, **memory**, **sensors and actuators**, **communication component** and **Power supply** (Karl and Willig 2007) (Figure 3-2)**.**

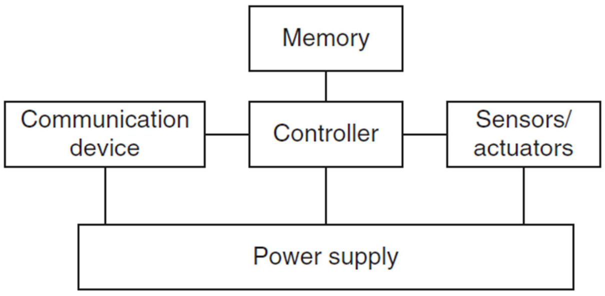


Figure 3-2 Overview of main sensor node hardware components
Source: (Karl, 2007)

The communication component refers to **transmitter** and **receiver**. A combined device is called **transceiver**. For the transmission medium, the usual choices include radio frequencies (RF), optical communication (Kahn, Katz et al. 1999), and ultrasound. It provides the functionality for exchanging data wirelessly among nodes.

Sensors and **actuators** are the actual interface to the physical world: devices that can observe or control physical parameters of the environment. There are many different kind of sensors: mechanical sensors**,** magnetic and electromagnetic sensors**,** thermal sensors**,** optical transducers**,** chemical And biological transducers**,** the electromagnetic spectrum**,** acoustic sensors (Kovacs 1998).

The controller is the core of a sensor node. It collects data from the sensors, processes those data, decides when and where to send it, and controls the actuator's behavior. It has to execute various programs, ranging from time-critical signal processing and communication protocols to application programs (Vieira, Coelho Jr et al. 2003). The most widely used controller is **microcontrollers**, like ARM, AVR, and C51 processer.

Memory unit is needed to store programs and intermediate data. Usually, different types of memory are used for firmware and data. Most of the **microcontrollers** have their own memory space for firmware. For keeping data, such as intermediate sensor readings, packages from other nodes and some other data, there are some choices, like Random Access Memory (RAM), Read-Only Memory (ROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), flash memory, or SD card.

The power supply can be a battery, cable power, or energy harvested from the environment. WSNs research has predominantly assumed the use of a portable and limited energy source, i.e. batteries, to power sensors. However, battery technologies haven't achieved too much in last 20 years, the power-to-weight ratio is still the limit of the battery to provide long term power supply. However, there are emerging WSN applications where sensors are required to operate for long durations, like years or even decades. Examples include environmental/habitat monitoring and health monitoring of critical infrastructures and buildings, where batteries are hard to be replaced or recharged. Lately, only cable power can provide reliable power supply, but the network has to rely on some infrastructures, and will lose the flexibility. An alternative to powering WSNs is being actively studied, which is to convert the ambient energy from the environment into electricity to power the sensor nodes (Seah, Eu et al. 2009). The challenge is that the energy harvesting device has to be comparable in size with the sensors (Seah, Eu et al. 2009). Solar cell is mostly used for harvesting, and already commercialized. Some other approaches like heat difference and vibration are also available.

3.1.2. Network Topology

Network topology refers to the way different nodes inside the network are interconnected. There are mainly three kinds of nodes: **gateway node**, **route node** and **end node**.

From the data collection point of view, gateway node is also called sink node. It gathers data from the network, and stores or relays these data. Some gateways also provide multiple interfaces such as Ethernet, Wi-Fi, USB, or serial ports for connecting the local network to outside network or terminal. Some networks, like ZigBee, also need the gateway node to manage the topology of the network. To fulfill these tasks, the gateway node should have relatively higher computing power then the other node, and also higher power consumption.

The sending distance is limited by the hardware and power of the WSNs. To extend the distance, the route node is used to relay messages. A message can be relayed several times before it reaches the destination, and this is called **multi-hop**.

End node will keep the minimal functionality, such as reading and sending sensor data, to save energy. A network doesn't necessarily have both end node and route node. The network that has both end and route node with different functionalities is called **heterogeneous** WSN, while, if all nodes are identical, it is called **homogeneous** WSN.

There are three typical network topologies: **star**, **cluster tree,** and **mesh** (Figure 3-3). All nodes of star networks are connected directly to the gateway node. No routing is needed; it is the simplest topology. However, the coverage of the network is limited by the sending distance of the transceiver. In cluster tree topology, all nodes are arranged in a tree structure, and each node can have only one **parent node** and several **child nodes**. The gateway node is the root of the tree, and the rest of the nodes are either on the branch or leaf. The data from the end node can be relayed by the nodes along the branch to the gateway node. In mesh networks, a node is allowed to communicate with its neighbor nodes. When the target node is not adjacent, messages will be relayed by the nodes along the route. If the route can be generated dynamically, it is called **ad hoc network**.

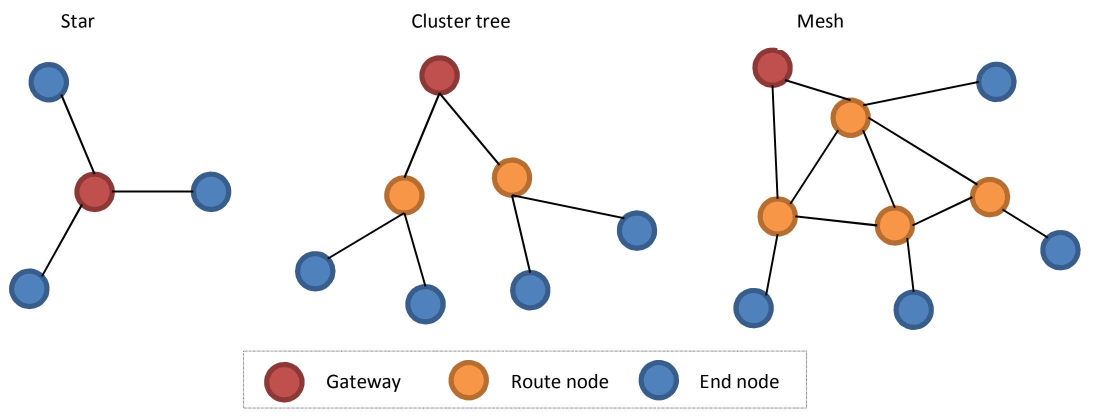


Figure 3-3 Three types of WSN topologies

3.1.3. OSI Stack Layers

The communication between two nodes needs to follow a certain **protocol**. The protocol is a defined set of rules and regulations that determine how data is transmitted. A lot of information has to be included in the protocol, such as how a message is encoded and decoded, ID of the node, the forward route, etc. To reduce the design complexity, most protocols are organized as a stack of layers, each one built upon the one below it. The purpose of each layer is to offer certain services to the higher layers, shielding those layers from the details of how the offered services are actually implemented. There is a conceptual model called **Open Systems Interconnection model (OSI model)**, which defines 7-layer stack protocol for general communication. In the real application of WSNs, it is simplified into 5 layers: **physical layer**, **data link layer**, **network layer**, **transport layer** and **application layer**.

The physical layer decides the actual transmission medium for the communication. RF communication techniques are generally used in WSNs. **Electromagnetic waves** are used for carrying the data. The frequency spans from 3 kHz to 300GHz. The protocol has to specify the facts like which frequency is used and how data is represented on the wave. The RF communication for WSNs can be classified into three as narrow-band, spread-spectrum, and ultra-wide-band (UWB) techniques (Oppermann, Stoica et al. 2004).

The data link layer ensures the reliable transmission of data frames between two nodes connected by a physical layer. A unique hardware address will be assigned to each node, and included in the head of the frame to make sure the message is sent to the right target.

The network layer constructs and manages the topology of the network. Routing and traffic control are main tasks for this layer. It determines how packets are routed from source to destination node.

The transport layer controls the reliability of a given link through flow control, segmentation and error control. However, based on Akyildiz's (Akyildiz, Su et al. 2002) research, there is no real implementation in the WSN.

Normally, the application layer is the one developer has to deal with. The developer decides the way to encode and decode the information in message.

The key success of the protocol stack is the modularized design, where the design of each protocol layer is decoupled from other layers. However, when the efficiency becomes the major concern, the multilayer stack might not desirable. These unique characteristics of WSNs have motivated cross-layer protocols that include the functionalities of two or more layers in a single coherent framework layers to minimize energy consumption.

There are quite a number of actual nodes available for WSN research and development. They are **Mica motes** (Hill and Culler 2002), **EYES nodes** (Dulman and Havinga 2002), **BTnodes** (Beutel, Kasten et al. 2004), and **Scatterweb** (Schiller, Liers et al. 2005).

3.2. The Design of Self-organizing WSN

In most short term experiments, the sensor network is just installed in the house for a period of time. When enough data is collected, the network will be removed. The self-organizing WSN is designed for this kind application. It is designed to minimize the settings and configurations in the installation process and dependence on infrastructure, so it can be easily deployed and removed from the experiment environment. With the self-organizing ability, the network can form the mesh network automatically when nodes are installed or new node be added. The database will also be generated and maintained correspondingly.

3.2.1. Features of Self-organizing WSN

- **Self-organizing**

  In the network, the wireless mesh network can be formed dynamically, which means whenever a node is added, removed or changed position, the topology of the network will be reformed automatically, i.e. the network can be setup at once or incrementally. This feature also gives the network self-healing ability. Whenever a node in the network is down, the data can be rerouted to a working path. It also enables the network to monitor moving node. According to the composition of sensors in the network, the database will automatically generate corresponding data tables for nodes. Self-organizing features make the deployment very convenient. A user just needs to put the node in the right place and turn it on.

- **Both event and ambient data**

The network is designed to record both **ambient data** (light, temperature, etc.) with a certain interval and the **event data** (open/close door, on/off light, etc.) instantly when it happens. This feature is specially designed for the pattern mining algorithms. Some research project only records with a fixed interval. The interval has to be set as short as possible, otherwise some events that happened in the interval will be missed. However, some data, like the room temperature that is changing slowly, doesn't need high frequency measurement. With this feature, we can keep the sensing frequency as low as possible without missing any event.

- **Independence**

  The system is capable of data acquisition and logging without relying on other computer and Internet connections. All the data collected during the experimental period is recoded on the SD card on the gateway node. So the network doesn't rely on the infrastructure of the experimental environment. With the database managing software provided with the sensor network, those data can be easily logged into the database, visualized, or converted to other format afterward.

- **Upgrade on the air**

  During the experiment, changes and adjustments are always needed. This network is able to upgrade the firmware on each node on the air, which means the upgrade can be done on the fly and no physical connection is need during the process.

3.2.2. The Architecture of Self-organizing WSN

The whole system is composed of three parts: hardware, software, and communication protocol. The hardware includes three types of sensor node: **gateway node**, **active sensor node** and **passive sensor node**. The software includes the **firmware** on board of the sensor node and the **database managing** and **firmware update program** on desktop side.

3.2.2.1. Hardware

All of the sensor nodes are equipped with an ATmega328P micro controller for processing and XBee DigiMesh 2.4 Wireless RF Module for communication. Different types of nodes have their own additional components for their special tasks.

The ATmega328P micro controller is an 8-bit AVR micro controller with the 32kb readable and writable flash rom, plus 1kb EEPROM and 2k SRAM(Atmel 2013). It has 23 general I/O pins, and six 10-bit A/D converters input ports, supports SPI, I2C, and serial communication. The Operating frequency is set at 16 MHz

XBee DigiMesh 2.4 Wireless RF Module works in the 2.4G band. It adopts the DigiMesh communicating protocol, and it can form mesh networks automatically and realize dynamic peer-to-peer communications. Each node has a unique 64 bits network address. Besides, it also supports the broadcast mode and multi-cast mode, with a transmission power of only 1 mw, a transmission distance about 90m without obstacles, and a highest transmission rate of 250kb/s. The module exchanges data with the micro controller through serial ports and the data exchange mode includes the transparent mode and API mode (Digi 2013). The system adopts the API mode.

The gateway node is mainly used to collect and record data from the network. Moreover, it is also responsible for the management of network operation, including node discovery, adding, deleting, and error control. When an abnormal event occurs, such as a node that doesn't respond to the request, the gateway node will record the error into the log file. Apart from the micro controller and wireless transceiver, the data logger node is also equipped with the real-time clock module which provides the time stamp of every sensor data, the SD memory card which records sensor data and log files, and the LCD display which shows the operating state of the network. The real-time clock module adopts the DS1307 chip and provides year, month, day, hour, minute, and second data. The module is connected to the micro controller with the I2C protocol. The SD card adopts the 512M mini SD card and connected to the micro controller with the SPI protocol. The LED display adopts the 96*64 OLED and connected to the micro controller via serial port (Figure 3-4).

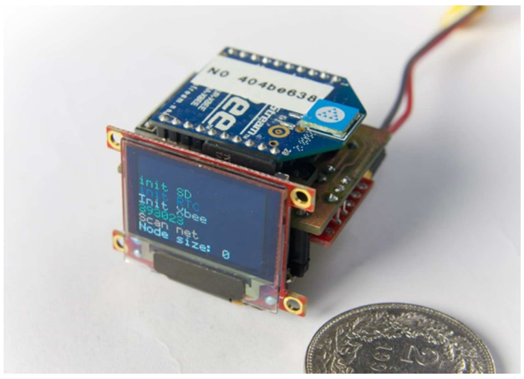


Figure 3-4 Gateway node

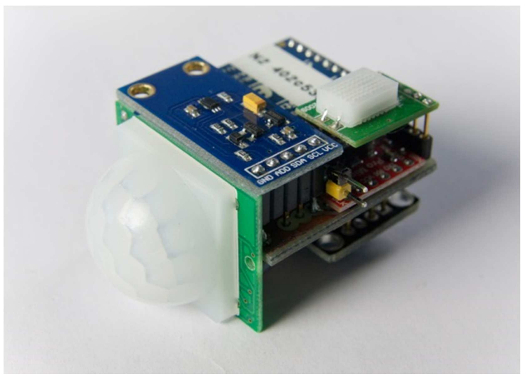
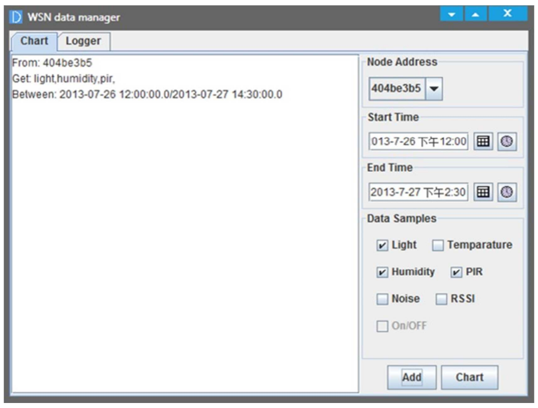


Figure 3-5 Passive sensor node

The sensor node has several different kinds of sensors on board and is used to collect sensor data and send them to the gateway node. There are also two kinds of sensor nodes: the passive node and active node. The passive node will send data after it receives an upload request from the gateway node. It is used to monitor continuously changing ambient values like temperature and humidity. The active node will send data when specific events occur and it is used to monitor state changing events like opening and closing of doors and windows. This design can, on one hand, reduce data transmission and its energy consumption by running on a relative low sampling rate, and on the other hand, capture instantaneous events occurring at sampling intervals. In this experiment, the passive node is equipped with temperature and humidity, luminous intensity, sound, and passive infrared (PIR) sensors. Among them, the temperature and humidity sensor adopts the MTH02A single-interface digital chip and luminous intensity sensor adopts the BH1750FV1 digital sensor and is connected via I2C. The PIR sensor adopts the RE200B passive infrared sensor, and the sound sensor adopts the BCM-9765P electronic microphone (Figure 3-5). By contrast, the active node is only equipped with a reed switch to detect the opening and closing of doors and windows.



```
From: 404be3b5
Get: light,humidity,pir,
Between: 2013-07-26 12:00:00.0/2013-07-27 14:30:00.0
```

Node Address: 404be3b5

Start Time: 013-7-26 下午12:00

End Time: 2013-7-27 下午2:30

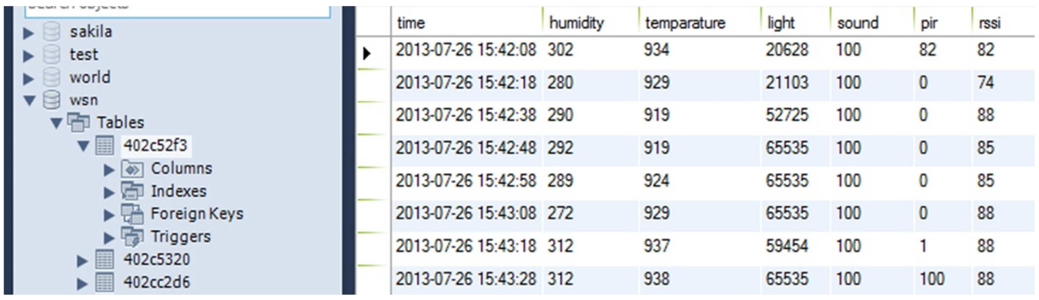
Data Samples: [✓ Light] Temparature [✓ Humidity] [✓ PIR] Noise RSSI On/OFF

Figure 3-6 User interface of the database managing program

3.2.2.2. *Software*

The firmware on the sensor node is written in C++, and it controls microcontroller to fetch data from those sensors on board, encode and decode the messages with the format defined by protocol.

The database managing program is a JAVA program communicates with database management system called MySQL. It parses the raw data stored in the SD card on gateway node, and separates them into according table of each sensor node that are named by the low 32-bit address of the node. Each row in the table saves a sample recorded by the sensor node. The first column is the timestamp of the sample, and the rest columns respectively save the sensor values returned by each sensor on board (Figure 3-7). Meanwhile, it can also read the historical data from the database on demand and visualize the data in different time series chart. In the user interface (Figure 3-6), the user is able to choose the time period and sensor types to create a multi axis time serial chart for one node. This kind of chart can help to provide an intuitive understanding of the correlations between different sensor data.



time	humidity	temparature	light	sound	pir	rssi
2013-07-26 15:42:08	302	934	20628	100	82	82
2013-07-26 15:42:18	280	929	21103	100	0	74
2013-07-26 15:42:38	290	919	52725	100	0	88
2013-07-26 15:42:48	292	919	65535	100	0	85
2013-07-26 15:42:58	289	924	65535	100	0	85
2013-07-26 15:43:08	272	929	65535	100	0	88
2013-07-26 15:43:18	312	937	59454	100	1	88
2013-07-26 15:43:28	312	938	65535	100	100	88

Figure 3-7 Sensor table in the database

Figure 3-8 is a chart for a sensor node in one experiment, placed next to the bed, which can monitor the ADLs like getting up, sleeping, or taking a nap. There are three sensor data overlapped in the chart, light, temperature, and PIR sensor. The record starts from 12:00 A.M. The PIR sensor was triggered when the occupant took a nap after lunch. At 6:30 P.M. when the occupant opened the window next to the bed, the room temperature dropped a lot, and the PIR was also trigged when the occupant passed by. At 9:00 P.M. the occupant closed the window. The room temperature went back. Around 9:00 P.M., when it was already very dark in the room, the occupant turned on the room lights, and the light value became constant. The occupant went to the bed at 11:30 P.M. Since the PIR sensor is very sensitive to movement, it is constantly triggered, but around 2:00 A.M. the occupant went into deep sleep, and no movement was detected. At 6:00 A.M., the occupant got up and opened the window again, the room temperature dropped correspondingly. Around 9:00 A.M., when the room temperature was very low, the occupant closed the window again. We can see that, with multiple sensor data, it is not so difficult to reconstruct the occupant's daily life. Here a pattern appeared twice, that is when the temperature is as low as 680 (raw data), the occupant will close the window.

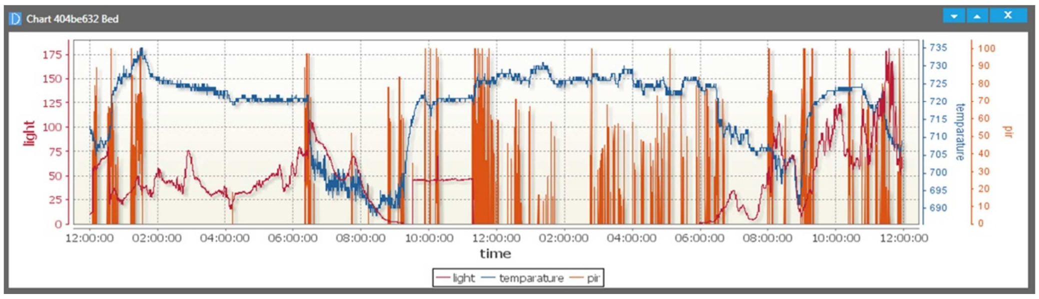


Figure 3-8 Multi sensor data chart for one node

Figure 3-9 is a chart that overlaps the light sensor data on different node in the same time period. Three of them were place in the room (green, yellow, and red), and there is very strong correlation between them. And the day-light has a strong effect on those sensor values. While the node, in the other room without direct natural illumination (blue line), just has every small variation in value.

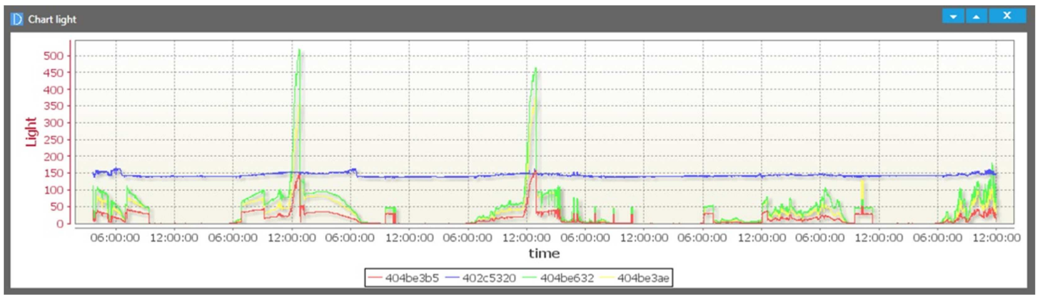
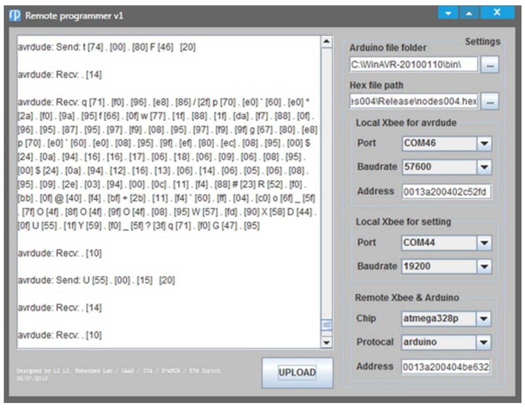


Figure 3-9 Multi sensor datachart for 4 nodes

The firmware update program is also a Java program and runs on a desktop (Figure 3-10). With an XBee node attached on the serial port, the controlling command and new firmware are sent to the target node wirelessly via the attached XBee.


Remote programmer v1
Settings
UPLOAD
Designed by LI LI. Embedded Lab / CAAD / ITA / D-ARCH / ETH Zurich
06/07/2015

```
avrdude: Send: t [74] . [00] . [80] F [46]   [20]

avrdude: Recv: . [14]

avrdude: Recv: q [71] . [f0] . [96] . [e8] . [86] / [2f] p [70] . [e0] ` [60] . [e0] *
[2a] . [f0] . [9a] . [95] f [66] . [0f] w [77] . [1f] . [88] . [1f] . [da] . [f7] . [88] . [0f] .
[96] . [95] . [87] . [95] . [97] . [f9] . [08] . [95] . [97] . [f9] . [9f] g [67] . [80] . [e8]
p [70] . [e0] ` [60] . [e0] . [08] . [95] . [9f] . [ef] . [80] . [ec] . [08] . [95] . [00] $
[24] . [0a] . [94] . [16] . [16] . [17] . [06] . [18] . [06] . [09] . [06] . [08] . [95] .
[00] $ [24] . [0a] . [94] . [12] . [16] . [13] . [06] . [14] . [06] . [05] . [06] . [08] .
[95] . [09] . [2e] . [03] . [94] . [00] . [0c] . [11] . [f4] . [88] # [23] R [52] . [f0] .
[bb] . [0f] @ [40] . [f4] . [bf] + [2b] . [11] . [f4] ` [60] . [ff] . [04] . [c0] o [6f] _ [5f]
. [7f] O [4f] . [8f] O [4f] . [9f] O [4f] . [08] . [95] W [57] . [fd] . [90] X [58] D [44] .
[0f] U [55] . [1f] Y [59] . [f0] _ [5f] ? [3f] q [71] . [f0] G [47] . [95]

avrdude: Recv: . [10]

avrdude: Send: U [55] . [00] . [15]   [20]

avrdude: Recv: . [14]

avrdude: Recv: . [10]
```

Arduino file folder: C:\WinAVR-20100110\bin\
Hex file path: es004\Release\nodes004.hex

Local Xbee for avrdude
Port: COM46
Baudrate: 57600
Address: 0013a200402c52fd

Local Xbee for setting
Port: COM44
Baudrate: 19200

Remote Xbee & Arduino
Chip: atmega328p
Protocal: arduino
Address: 0013a200404be632

Figure 3-10 Interface of the firmware updata program

3.2.2.3. Communication Protocol

Communication protocol on the application layer is used to control the request sending and data exchange between the nodes. The protocol is built on top of the Digi-Mesh protocol. Since the addressing and routing information is already tackled in these lower layer protocols, it is mainly used for formatting network request, includes adding nodes, deleting nodes, collecting data, inserting data, joining network, and exiting network, etc.

3.2.3. The Workflow of the Sensor Node

To minimize the configuration, the network deployment and data collection process are all automated. Generally, a node has initializing and looping states. In the initializing state, the microcontroller checks all the devices on board, and gets its network address form the XBee module. In the looping state, the node keeps operating the programmed instructions in a loop (Figure 3-11).

3.2.3.1. The Workflow of the Gateway Node

- Connect to the power, initialize all devices, and acquire current date and time.
- The XBee wireless transceiver automatically forms the network, and joins the wireless mesh network.
- Scan the network, record the number of found sensor nodes, and acquire the 64-bit addresses of all nodes. Write them into EEPROM if their addresses are not recorded yet.
- Start the timer, and enter the loop state after the initialization ends.
- Monitor XBee, and analyze the 64-bit addresses and instruction contents of the source nodes once receiving instructions. If it is the instruction of joining nodes, reply the acknowledge instruction and add the 64-bit addresses into EEPROM. If it is the instruction of inserting data, record the data into the SD card.
- Check the timer. If it arrives at the specified time, start to collect data from the nodes, extract the 64-bit addresses one by one from EEPROM, and send the upload data instruction to this node. After the target node returns data, record them into the SD

card if it receives the data. If the target node fails to return data for three consecutive times, then delete its node address from EEPROM.

- Reset the timer and prepare for the re-loop.

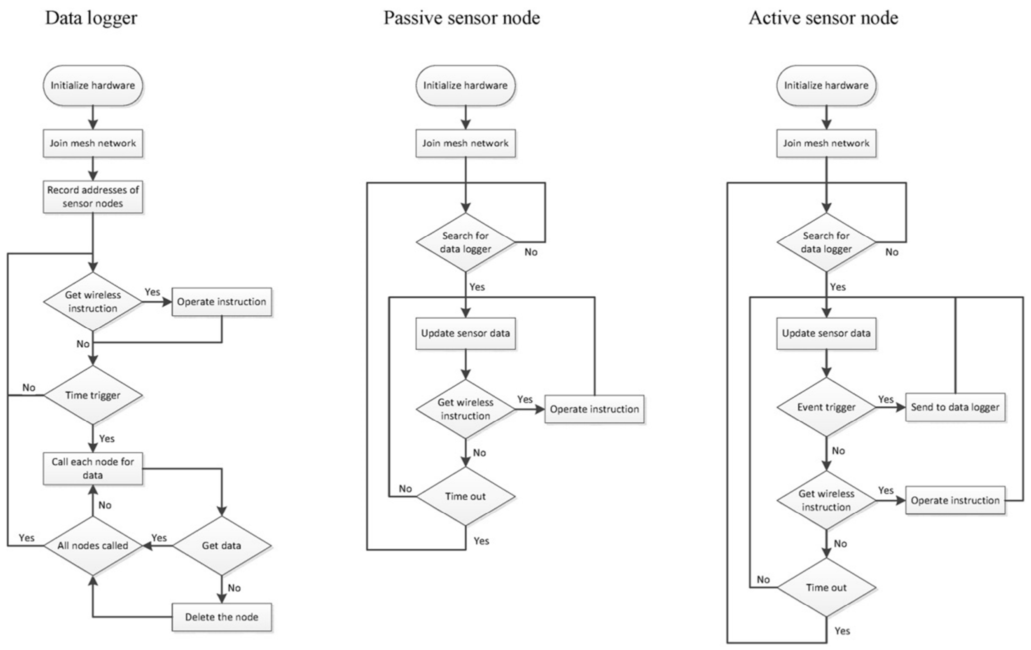


Figure 3-11 Workflow for each type of node

3.2.3.2. *The Workflow of the Passive Sensor Node*

- Connect to the power and initialize all devices.
- The XBee wireless transceiver automatically forms the network, and joins the wireless mesh network.
- Send the instruction of joining nodes in the broadcast mode and wait for the reply from the data logging node. If it receives the reply, then record its 64-bit address.
- Initialize the timer and enter the loop state.
- If the timer time-out occurs, restart the node.
- Monitor the wireless data received by XBee. If it receives the upload data instruction, read the data of every sensor and send them to the data logging node. Reset the timer.

3.2.3.3. *The Workflow of the Active Sensor Node*

- Connect to the power and initialize all devices.
- The XBee wireless transceiver automatically forms the network, and joins the wireless mesh network.

- Send the instruction of joining nodes in the broadcast mode and wait for the reply from the data logging node. If it receives the reply, then record its 64-bit address.
- Initialize the timer and enter the loop state.
- If the timer time-out occurs, restart the node.
- Monitor the sensor. If jump change occurs, send the instruction of inserting data to the data logging node and send the data as well.
- Monitor the wireless data received by XBee. If it receives the upload data instruction, read the data of every sensor and send them to the data logging node. Reset the timer.

By taking advantage of mesh network, the deployment of the network is very flexible. The sensor node can either be passively discovered by the gateway node at the beginning or join the network later by sending the join network request. There is no need to preset all the information of the sensor node and network topology in the gateway node. The data logging node can automatically remove the sensor nodes with abnormal events based on the package loss of the data transmission, thus preventing ineffective data transmissions. Also the sensor nodes will automatically restart themselves after noticing abnormal events of their own and try to rejoin the network, which ensures that the network possesses certain self-healing abilities

3.2.4. System Evaluation

In the real test, the data-logging node collected data from a network with 5 passive nodes and 2 active nodes every ten seconds and the data volume recorded every day amounted to 1M bytes. Every passive node produced about 170K bytes and every active node produced about 80k bytes. Plus the space occupied by the log file, a 512M SD card was enough to store a year's data volume.

As for the stability of wireless data transmission, the received signal strength indication (RSSI) of a node 5 meters away in sight line could keep the range within -60dbm~-65dbm. Even if the node received disturbance from other electromagnetic signals, it could still ensure the received signal strength is more than the lowest receive power of XBee, namely -95dbm, without losing data packages. The node blocked by wall kept the received signal strength indication between -76dbm~-83dbm under normal circumstances. The package loss can be controlled fewer than 2%, which did not create great influence on the data integrity.

3.3. The Design of Ultra-low Power WSN

For the short-term application the simplicity of deployment is the major concern. While for the long-term application, the power consumption is the key issue. The design of the ultra-low power WSN is trying to provide a battery based multi hop wireless sensor network for long term monitoring the indoor environment. This design provides efficient network layer protocol and sleeping mechanism to bring down the power consumption to the minimum. It adopts a cluster tree topology, and although it is not ad-hoc network as self-organizing WSN, the network topology can still be changed on the fly, which is enough for most of indoor application where most of the nodes are stationary.

3.3.1. Features of the Ultra-low Power WSN

- **Sleeping and synchronizing**

  Most energy is wasted by idle listening. Since a node does not know when the message will come, it must keep its transceiver in receive mode all the time. However, in smart home application, the messages are fairly short: they take less than several milliseconds to transmit. There is no better way to conserve energy but to put the nodes to sleep. All the nodes inside the network only wake up for several milliseconds for sending messages, and the rest of time they are in deep sleep mode, which only cost only several micro amps. In this proposed network, such duty cycle is implemented and synchronized in the whole network with a power management scheme that does not demand high accuracy of timing synchronization.

- **Multi-hop network**

  The network layer protocol enables the message being delivered in the multi-hop manner. Although there are several mesh network allows multi hop or ad hoc, but there is no sleeping function integrated or only end node can set to sleep. However, in this network, except the gateway node, the other entire node can be set to sleep, no matter it is route node or end node. Due to the changeable route table, when a node is out of function, the developer is able to remotely rearrange the path for the multi-hop to bypass the fault node.

- **Remote access**

  A GSM module is integrated on the gateway node, which enables Internet access from anywhere via GPRS network. Bi-directional communication can be established. Gateway node can upload the collected real time data onto the remote server. Remote server can also send commands to control the network.

3.3.2. The Architecture of the Ultra-low Power WSN

The whole system is composed of three parts: hardware, software, and communicating protocol. The hardware includes two types of sensor node: gateway node and sensor node. The software includes the firmware on board of the sensor node, the API for communicating with the gateway node from the outside network, and the network protocol for internal routing and communication.

3.3.2.1. Hardware

All of the sensor node are equipped with an ATmega 2560 micro controller for processing, XBee-PRO 868 Wireless RF Module for communication and additionally DS3231 RTC module for synchronizing the wake up time of each node. Different types of node have their own additional components for their special tasks.

Since handling the routing protocol needs more computational power and RAM space, a more powerful microcontroller ATmega2560P was chosen. It is an 8-bit AVR micro controller with the 256kb readable and writable flash rom, plus 4kb EEPROM and 8k SRAM .It has 86 general I/O pins, and 16 channel 10-bit A/D converters input ports, supports SPI, I2C, and 4 serial communications. The Operating frequency is set at 16 MHz (Atmel 2014).

The DS3231 is a RTC driven by a temperature compensated 32 kHz crystal oscillator. The TCXO provides a stable and accurate reference clock, and maintains the RTC to within ±2 minutes per year with an accuracy from -40°C to +85°C. With its alarm function, it can be attached to the interrupt pin of the microcontroller, which wakes it up periodically.

On the gateway node, there is a SIMCom SIM908 GSM module installed. It is a Quad-Band 2G network module, which support the GPRS data transfer. On the sensor node, there are two additional sensors. One is a MAX1704 Lipo fuel gauge, which can measure the voltage and the percentage of the battery. Another is an ADS1115 16-Bit ADC converter, which can provide more precise measurement for external analog sensor.

3.3.2.2. Network Topology

The network consists of two types of nodes, gateway node (Figure 3-13) and sensor node (Figure 3-14). They are interconnected in a cluster tree topology (Figure 3-12). Each node has one parent node and several child nodes, the node at the bottom without a parent node is the root node, and the node at the top without a child node is the leaf node. A node that is connected to all lower-level nodes is called an ancestor. The connected lower-level nodes are descendants of the ancestor node.

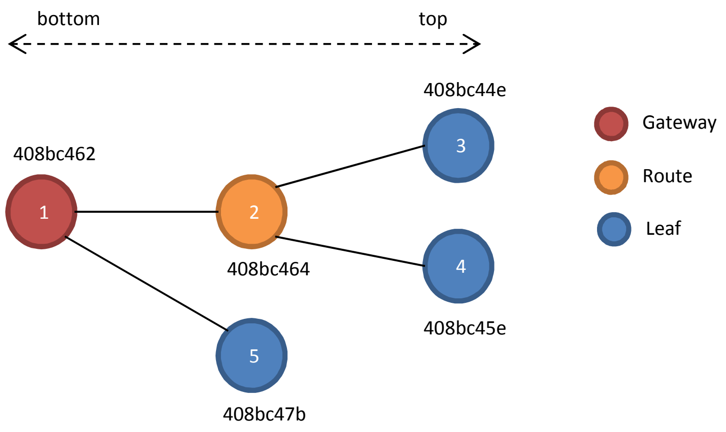


Figure 3-12 Diagram of the tree cluster topology of a 5 node network

There is only one gateway node at the root of the cluster tree. The gateway node has two tasks. One is providing bi-directional communication with the outside network. The gateway provides an API (Application Programming Interface) that developers can send POST requests to control or query data from the WSN. The gateway can also upload sensor data to the remote server. The other is maintaining the local network, like setting or changing the routing table, synchronizing the network, querying the data, etc.

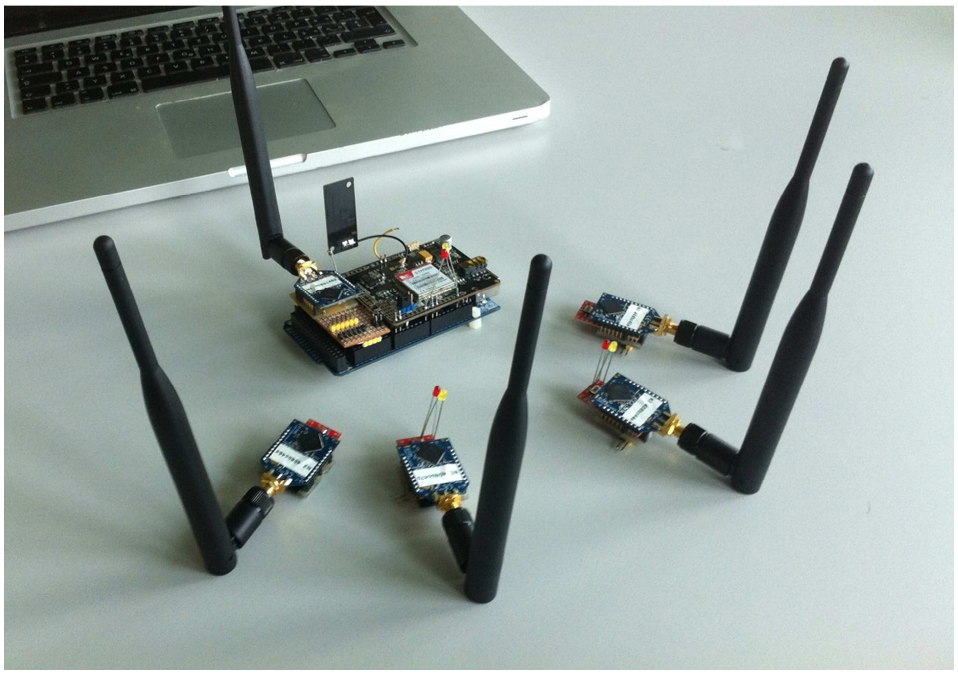


Figure 3-13 Gateway node and end node of ultra-low power WSN

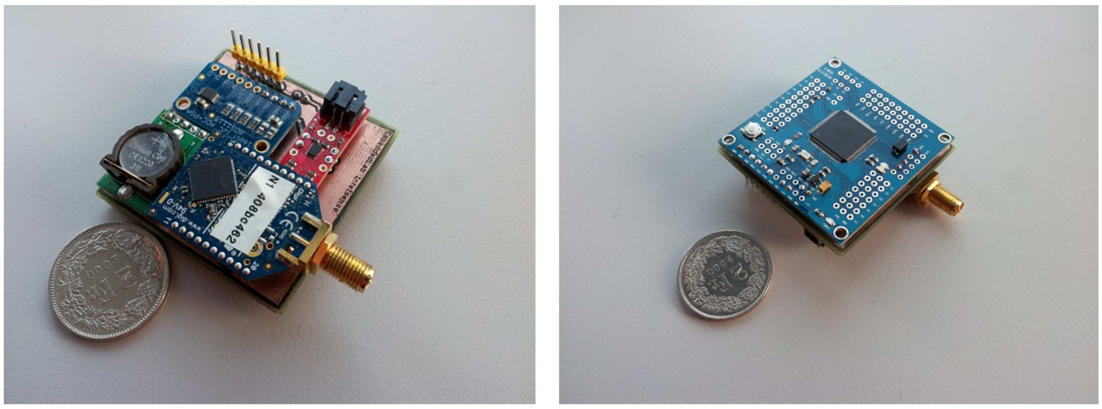


Figure 3-14 Sensor node of ultra-low power WSN

The hardware and firmware of the sensor nodes are identical, but they act differently when they are in different locations of the tree structure. The node along the branch has to do both sensing and routing tasks, while the node on the leaf only needs to do the sensing. Depending on the sensing task, nodes can also be divided into passive sensors and active sensors. The passive is periodically woken up by the gateway node and senses the ambient data, while the active senor can be triggered by certain events detected by the sensor attached. The event sensor node doesn't send the message right after the event is detected, since all the other nodes are probably sleeping. The event and its timestamp are buffered in the node, and sent together with the other data when the gateway queries the data.

3.3.2.3. The Protocol

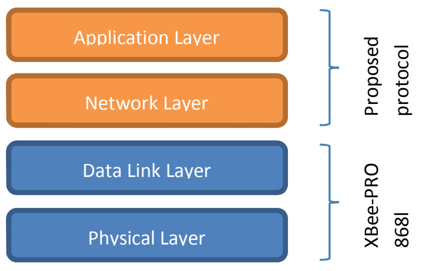


Figure 3-15 Protocol stack of ultra-low power WSN

There are already a lot of studies on the routing protocols for WSNs (Singh, Singh et al. 2010). However, most of them are designed for a mesh network using a reactive protocol that discovers the route in run time. Such features are redundant for a cluster tree network. When sensor nodes are static, it is preferable to have **table driven routing protocols** rather than using reactive protocols (Al-Karaki and Kamal 2004). In this network, a routing table based protocol is designed and implemented, which **allows multi-hop communication in a cluster tree network with sleeping mode**. From the protocol stack point of view, the protocol is built on top of the physical and data link layer provided by the XBee transceiver (Figure 3-15). They handle the point to point communication among the transceivers. The proposed protocol covers the network layer and application layer. The network layer handles the duty cycle and multi-hop routing and the application layer formats the sensor data. The network protocol consists of three parts: **a power management scheme** for synchronizing the sleeping nodes, **routing tables for multi hop communication** and **routing message**.

To synchronize the duty cycle of all nodes in the network is the essential problem. Different methods have been tried (Sadler and Swami 2006), such as GPS signal and Network Time Protocol (NTP). However, most of them are not energy-aware. The synchronization with RTC is a low power consumption and low cost solution, and selected for the proposed network. But, physical clock synchronization is always imperfect (Sichitiu and Veerarittiphan 2003). All commercial RTC products suffer from drift problems. When drifts accumulate, nodes can't wake up simultaneously. Many studies have been done to deal with the drift issue. But they are either for mobile network or single hop solution.

Here, a more simplified solution is provided. The drift of the RTC is 2 minutes per year, which means 200 μs per minute or 1ms per 5 minute. The worst case of the error between two nodes is the double of the value. So, the node has to be resynchronized periodically, and a tolerance for the drift needs to be added to the wake up time. The more often the network is resynchronized, the less tolerance is needed. Since the querying order of each node is fixed and each query will create a certain delay, nodes do not need to wake up at the same time, they just need to wake up a little earlier before their term to be queried. Take the network in Figure 3-12 for example. Figure 3-16 shows the wake up scheme for each node. Here, the gateway node (node 1) wakes up the entire network layer by layer. Each node tries to wake up its child nodes, node 1 wakes up node 2 and 5, while node 2 wakes up node 3 and 4. There are two kinds of ACK messages, one is the wake up ACK, which says the node itself is awake, and the other is descendants wake up ACK, which says all the descendant nodes are awake. After sending the wake up requests, the parent node will wait first the wake up ACK from its child nodes to ensure they are awake, and then wait for the descendant wake up ACK from these child to make sure their descend nodes are awake. The line in Figure 3-16 represents the sleep period and the block represents the working period. The arrow represents the message. In this process, the gateway node first waits until the next wake up time. The wake up time of its child nodes, node 2 and node 5, are delayed a bit compare to the gateway node. The gateway node sends wake up requests to both nodes and waits for ACK. Node 5 has no child node, so it directly sends the descendants wake up ACK to inform the gateway node that all nodes on this branch are awake. Node 2 has two child nodes, node 3 and node 4. Therefore, node 2 first sends a wake up ACK to stop the gateway's calling, and then it sends a wake up request to its child node. Both node 3 and node 4 don't have child nodes, so each of them returns a descendants wake up ACK. When node 2 receives theses ACKs, it knows all its descendant nodes are awake. It sends the descendants wake up ACK to the gateway node. The gateway node also gets notice that all its descendants are awake, and the whole wake up process ends.

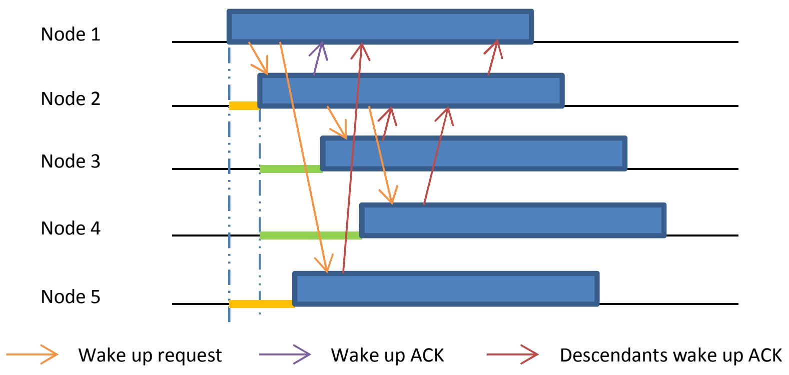
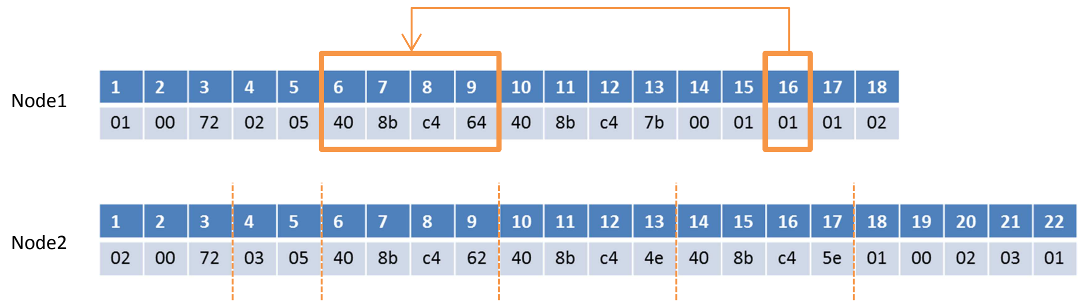


Figure 3-16 Wake up process of the ultra-low power WSN

The routing of the messages relies on the routing table distributed in each node. Given a destination node of a message, a route node should find out which neighbor node should be the router or destination for the message. Take the network in Figure 3-12 for example. When node 1 has to send a message to node 3, it should know that the next router is node 2. Since the physical address of the node is 4 bytes long, it will take a lot of memory space to store the address of the route node, and most of them are repeated. For example, sending from node 1 to node 3, 4 are both using the node 2 as the router. A more **compact routing table** is adopted. The routing table has one record of each physical address of its neighbor nodes, and assigns each address with a 1-byte ID. Then the route node for each destination is only represented with the 1-byte ID. For example, Figure 3-17 shows routing tables in node 1 and 2. In node 2's table, the first 3 bytes is the header, byte 4 and 5 recodes the amount of neighbor and total amount of destination nodes respectively. From byte 6 to byte 17 are the physical addresses of its neighbor nodes: node 1, node 3, and node 4. The rest of the 5 bytes are 1-byte IDs of the route nodes. Their positions indicate the node ID of the destination node. For example, node 1 is going to send a message to node 3. The third byte in the router node IDs is 1, which means it is the first neighbor node that should be used to route the message, i.e. the address of node 2. If a node has m neighbor nodes and n destinations, the size of the normal routing table is $(4 \times n)$

bytes, while the size of proposed compact routing table is $(4 \times \mathrm{m} + \mathrm{n})$ bytes. In the multi hop network, the amount of neighbor nodes is much smaller than the total number of destination nodes. Even In this small-scale case, node 1 saves $4 \times 5 - (4 \times 2 + 5) = 7$ bytes, and node 2 saves 3 bytes. The larger the network, the more efficient the compact routing table is.



Node1

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
01	00	72	02	05	40	8b	c4	64	40	8b	c4	7b	00	01	01	01	02

Node2

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
02	00	72	03	05	40	8b	c4	62	40	8b	c4	4e	40	8b	c4	5e	01	00	02	03	01

Figure 3-17 The routing table for node1 and node2 in ultra-low power WSN

The actual message is wrapped in the XBee package (Figure 3-18). The package begins with the XBee header, which contains information like the start byte, message length, frame ID and hardware address of the destination node. The next byte is the destination node ID. Combined with the routing table, the route in the tree topology can be found. The next byte is the source node ID. It is needed when the destination node has to send an ACK back to the source node. The next byte is the command ID, which specifies the function of the message, and the rest of the bytes are the payload according to the command. At the end, there is a checksum for verifying the content. If it is a data collection command, and the formatted sensor data will be filled in the payload.

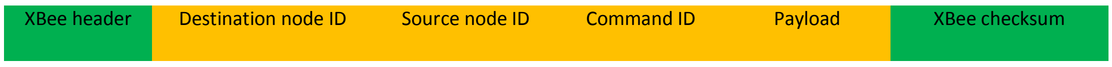


Figure 3-18 The package of the ultra-low power WSN message

3.3.3. The Workflow of the Ultra-low Power WSN

Unlike the self-organizing WSN, where every node is working autonomously, in this network the gateway node conducts most of the network operations. To collect the sensor data from the network, there are three main steps: First, the gateway wakes up the whole network; secondly, it queries data from each node; lastly, it sets the whole network sleep again.

The waking up process of the network has been discussed with the power management scheme before. To collect the sensor data there are two different ways, the first way is **node-wised** and the second way is **level-wised**. In the node-wised approach (Figure 3-19 a), the gateway sends data requests to each node and collects the return data one by one. Since each time only one node is sending, so collisions can be avoided, the communication is more reliable. While the drawback is the working load of the route node is times than the end node, because it has to forward the request and return data for every descendant node. In the level-wised approach (Figure 3-19 b), each node collects all data from its child nodes, packs them together and sends to its parent node. In this way the parent node only needs to send the request to its child nodes once instead of all descendent nodes. However, this approach also has some drawbacks. If a network fault happens when returning the sensor data, not only the data of this node, but also data for all its descendent nodes will be lost. Another problem is that all nodes are sending simultaneously, which increase the chance of collision. The choosing of the approach is really depending on the network size and structure. The level-wised approach is ideal for the network structure with many levels, which will save a lot of routing tasks for the router nodes. For the network with more nodes but fewer levels, the node-wise approach is better choice, which can avoid the collisions during the collections.

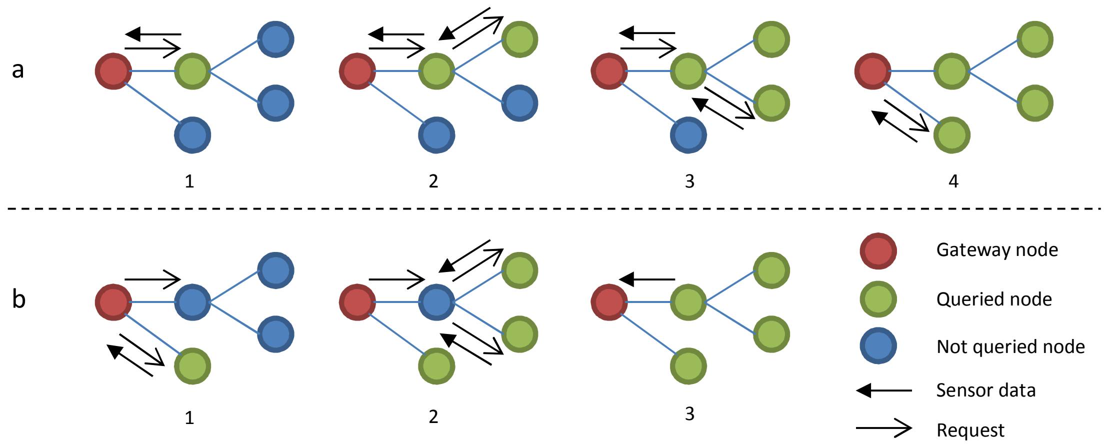


Figure 3-19 Node-wise and level-wise way to collect data in ultra-low power WSN

3.3.4. Estimation of the Power Consumption

The real power consumption of the WSN is hard to calculate precisely. It can be affected by many factors, such as the amount of obstacles, the signal interference, the sensor attached, etc. Only an estimation can be given. The power consumption consists of two parts: the sleeping period and the data transmission period. Most of the time, the node is just sleeping. Reducing the consumption during the sleeping mode will help to extend the lifetime of the network. While, decided by the characteristic of RF, the sending distance is in proportion to the sending power. Reducing the sending power needs more sensitive receivers, or more routers, that will not help in bringing down the power consumption. So the best way to reduce the consumption is to reduce the wireless communication by either reducing the messages or using shorter messages. Both microcontroller and RTC will create the consumption during the sleeping period. The current of the sleeping microcontroller with interrupts on is 7μA. During the working time, it draws 15 mA. The RTC draws 70μA. The TX Current of the XBee module is 85mA, and the RX Current is 65mA. It needs around 25-30 ms between sending an API packet to receiving the ACK. For an end node, it needs to send at least 3 messages during one data collection period. The wakeup period to fulfill one data collection depends on the amount of nodes and layers in the network. Take the network in Figure 3-12 for example. The whole awake period is around 3 seconds. So the power consumption for the awake period is 85 mA * 30 ms * 3 + (15 mA + 65 mA) * 3000 ms = 247650 mAms = 0.0688 mAh. With a 5-minutes data collection interval, the power consumption for sleeping is 0.077 mA * 5 * 60 * 1000 ms = 23100 mAms = 0.006 mAh. So it is 0.0694 mAh. With a 6000 mAh battery, the node can last around 300 days with 5-minute sending intervals. The power consumption of the router will be larger since it has additional routing tasks, which decided by the amount of its descendants.

Chapter 4.
Frequent Pattern Mining for ADLs

In 1993, Agrawal invented the **Apriori algorithm** to find all co-occurrence relationships, called **associations**, among data items (Agrawal, Imieli et al. 1993). He first applied **downward-closure property** in finding **frequent itemsets** in the transaction database. The downward-closure property guarantees that if an itemset is frequent, all its ***subsets*** are also frequent. Accordingly, for an infrequent itemset, all its ***supersets*** must be infrequent. This property narrows down the search space drastically.

Since then, it has attracted a great deal of attention. Many derived efficient algorithms, extensions and applications have been proposed. From the database view, there are mainly three different variations: **Association Rule Mining (AR)**, **Sequential Pattern Mining (SPM)**, and **Frequent Episode Mining (FEM)**. All of these are based on the downward-closure property, but the input database and searching target are slightly different. Association Rule Mining works on a **transaction database** and searches for **frequent itemsets.** It does not consider the order of transactions. Sequential pattern mining discovers **frequent sub-sequences** in a **sequence database**, while frequent episode mining tries to find out the repeated **episode** in a **temporal database** (Table 4-1).

Algorithm	Association rule mining	Sequential pattern mining	Frequent episode mining
Database	Transaction database	Sequential database	Temporal database
Target	Frequent itemset	Sequential pattern	Frequent episode
Rule	Downward-closure property		

Table 4-1 Three different frequent pattern mining algorithms

The concept of **Activities of Daily Living** (**ADLs**) was originally proposed by Dr. Sidney Katz (Katz, Ford et al. 1963). It is a term used in healthcare to refer to people's daily self-care activities, such as feeding, bathing, dressing, grooming, working, homemaking, and leisure. Humans usually perform ADLs in a sequential manner, such as brushing teeth followed by washing face, and preparing a meal followed by eating the meal.

However, in real life, there exists many complex situations since people often multitask when performing their activities. Such multitasking may occur in an interleaved or concurrent manner(Tao, Liang et al. 2011). Usually, there will also be a lot of randomness in ADLs caused by the change of the mood or the environment. There are no clear boundaries between consecutive activities. As a result, the length of the patterns can be highly variable. Practically, the ADLs have to be recorded by sensors. Noise can't be avoided because of the fault of the network or the individual differences in sensors. Moreover, with current sensor technology, we can only get the low level sensor data, such as open/ close of the bedroom door or on/off of the lights in the living room. It is not possible to get the high level ADLs directly for the sensors. All these reasons make it very hard to apply some of machine learning algorithms on the raw data collected in a smart home. However, FEM algorithms can deal with such kinds of problems. It can extract frequently sequential patterns that are randomly distributed in data set within a certain tolerance.

Figure 4-1 shows the whole concept of applying FEM in ADLs mining. The ADLs consist of both regular behaviors and irregular behaviors of inhabitants. The mining target is the regular portion. A sequence of regular behaviors can interleave with other regular behaviors or be interrupted by some irregular behaviors. These behaviors are detected by sensors installed indoors or integrated in the appliances and recorded in the form of discrete or continuous values. Some fault data caused by data transmitting errors or wrong measurements will also be mixed into the data sequences. These sensor data sequences with noise are fed into the frequent mining algorithm as input data. The algorithm will find out the frequent sensor data sequence. The frequent sequences can be interpreted back into ADLs or be further processed to find out the sequential rule among them.

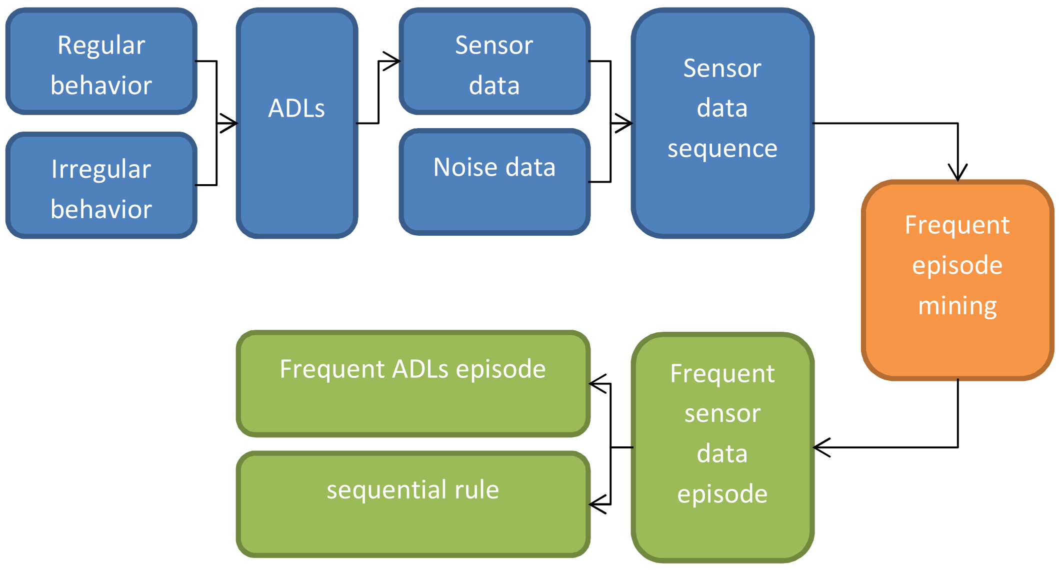


Figure 4-1 Workflow of FEM for ADLs

Since FEM is derived from SPM, it borrows most of the concepts from SPM. Most of the optimization techniques used in SPM can be applied easily in FEM. The main difference between them is the **frequent metric**. So, both of FEM and SPM are investigated in this research.

4.1. Introduction of SPM

There are many applications involving sequence data, such as customer shopping sequences, Web clickstreams, and biological sequences. A sequence database consists of ordered elements or events, recorded with or without a concrete notion of time. Sequential pattern mining was first introduced by Agrawal (Agrawal and Srikant 1995), and was defined as follows:

"Given a database of sequences, where each sequence consists of a list of transactions ordered by transaction time and each transaction is a set of items, sequential pattern mining is to discover all sequential patterns with a user-specified minimum support, where the support of a pattern is the number of data sequences that contain the pattern."

The mining of frequently occurring ordered events or subsequences as patterns has become an important problem in data mining. The preliminary concept about sequential patterns will be first introduced.

4.1.1. Preliminaries

There are some common concepts shared by all the SPM algorithms: Item is the minimum unit. A group of unordered items is called itemset. A list of ordered itemsets is called a sequence. A sequence database is group of sequences.

Definition 1 **(Item, Itemset)** Let $I = \{i_1, i_2, \dots, i_m\}(m \geq 1)$ be a finite set of m distinct attributes. Each of these attributes is called an **item**. An **itemset**, denoted as $a = \{i_{x1}, i_{x2}, \dots i_{xn}\}(1 \leq x1, x2,,, xn \leq m)$, is a nonempty subset of I. An item can occur at most one time in the same itemset, but multiple times in different itemsets. $|a|$ represents the amount of the items in the itemset, called the size of the itemset.

Definition 2 **(Sequence, Simple sequence, Complex sequence)** A **sequence** S, an ordered list, is denoted as $s = \langle(e_1, t_1)(e_2, t_2) \dots (e_n, t_n)\rangle(n \geq 1)$, where $e_j \subseteq$ I is a **data element**, also called an **event** in FEM. A data element can be an itemset when $|e_j| > 1$, or an item when $|e_j| = 1$. $t_j \in \{1,2, \dots\}(j = 1, \dots, n)$ is the timestamp of e_j in s, $t_j < t_{j+1}(\forall\, j = 1, \dots, n-1)$. If the sequence is consecutive, i.e., $t_j = j$, then the sequence is abbreviated as $\langle(e_1)_1(e_2)_2 \dots (e_n)_n\rangle$ or without the timestamp $\langle e_1 e_2, \dots e_n\rangle$. If all data element sizes in the sequence equal 1, then the sequence is a **simple sequence**, otherwise, it's a **complex sequence**. The size of the sequence, denoted as $|s|$, is the total amount of ***data element*** s contains. The length of the sequence, denoted as $l(s)$, is the total amount of its items, i.e., $l(s) = \sum_{i=1}^{n} |e_i|$. A sequence with length l is called an $\boldsymbol{l-sequence}$. When it is a *single-item sequence*, $|s| = l(s)$.

Example 1. If $I = \{a, b, c, d, e\}$, is a sets of letters. each letter is an item. (abc) is an item set. $s_1 = \langle(ab)(a)(cd)\rangle$ is a complex sequence. $s_2 = \langle(a)(b)(d)\rangle$ is a single-item sequence. Both $s_3 = \langle(ab)(a)\rangle$ and $s_4 = \langle(a)(a)(cd)\rangle$ are sub-sequences of s_1. s_1 containing s_3 and s_4.

Definition 3 (Sequence database, Transaction database, Temporal database) A **sequence database**, $D = \{s_1, s_2, \dots, s_n\}$, is a set of sequences. Each sequence is associated with an id. $|D|$ represents the amount of sequences in the database D. A transaction database, $T = \{x_1, x_2, \dots, x_n\}$, is a set of itemsets. Each itemset is associated with an id. $|T|$ represents the amount of itemsets in the database D. Temporal database is a single long sequence.

Transaction ID	Transaction
1	$(abce)$
2	(bce)
3	(cde)

Table 4-2 Transaction database

Sequence ID	Sequence
1	$\langle(a)(b)(ce)\rangle$
2	$\langle(b)(ce)\rangle$
3	$\langle(cd)(e)\rangle$

Table 4-3 Sequence database

Timestamp	1	2	3	4	5	6	7	8	9
Event	(a)	(b)	(ce)	(b)	(ce)	(cd)	(e)	(a)	(b)

Table 4-4 Temporal database

Definition 4 (Sub-sequence, Super-sequence) Given two sequences: $\alpha = \langle a_1, a_2, \dots, a_m\rangle$ and $\beta = \langle b_1, b_2, \dots b_n\rangle$. α is a sub-sequence of β, if and only if $\exists i_1, i_2, \dots, i_m (1 \leq i_1 < i_2 < \cdots i_m \leq n)$ makes $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \dots, a_m \subseteq b_{i_m}$, denoted as $\alpha \sqsubseteq \beta$ or $\alpha \sqsubset \beta$ when $\alpha \neq \beta$. At the same time β is the super-sequence of α, and β **contains** α. If β contains α, and they have same support in the database, β **absorbs** α.

Definition 5 (minimum support, frequent sequence) The support of a sequence α in a sequence database D is the number of sequences containing α, denoted as $support_D(\alpha) = |\{s|s \in D \text{ } and \text{ } \alpha \sqsubseteq s\}|$.Given a minimum support, $\min\_sup$, frequent sequence set includes all the sequences whose support is no less than $\min\_sup$.

Definition 6 (Downward-closure property on SPM) if the support of a sequence α in a sequence database D, $support_D(\alpha) < \min\_sup$. Then, in the database, the support of any super-sequence will be smaller than $\min\_sup$, i.e., $\nexists\beta\ (\alpha \sqsubseteq \beta)$ has $support_D(\beta) \geq \min\_sup$.

Example 2. Table 4-2 represents a transaction database T. $|T| = 3$. The support of itemset $x = (ce)$ is $sup_T(x) = 3$. Given $\min\_\mathrm{sup} = 2$, the frequent itemsets are $FS = \{(b){:}\,2, (c){:}\,3, (e){:}\,3, (bc){:}\,2, (be){:}\,2, (ce){:}\,3\}$, where the notation "$\langle pattern\rangle$: count" represents the pattern and its associated support count.

Example 3. (Table 4-3) represents a sequence database D. $|D| = 3$. The support of sequence $= \langle(b)(c)\rangle$, $support_D(s) = 2$. Given $\min\_\mathrm{sup} = 2$, the frequent sequence is $FS = \{\langle(b)\rangle{:}\,2, \langle(c)\rangle{:}\,3, \langle(e)\rangle{:}\,3, \langle(b)(c)\rangle{:}\,2, \langle(b)(e)\rangle{:}\,2, \langle(b)(ce)\rangle{:}\,2\}$. Let $s_1 = \langle(a)\rangle$, $s_2 = \langle(a)(b)\rangle$, $s_3 = \langle(b)(c)\rangle$, and $s_4 = \langle(b)(ce)\rangle$. Based on the downward-closure property, because $support_D(s_1) = 1 < \min\_sup$, $s_1 \sqsubseteq s_2$, therefor, $support_D(s_2) < \min\_sup$. Correspondingly, because $support_D(s_4) = 2 \geq \min\_sup$ and $s_3 \sqsubseteq s_4$, therefore, $support_D(s_3) \geq \min\_sup$.

Example 4. Table 4-4 is a temporal database s. $|s|$ = 9, and $l(s) = 15$. The calculation of the support will be discussed in next section.

4.1.2. Categories of SPM Algorithms

Mining sequential patterns is computationally expensive. A major part of the variations of the algorithm were focused on reducing the computational cost by adding constrains or providing more efficient searching techniques. Another problem of SPM is the redundancy of the mining results. Researchers also target on mining smaller subsets of the overall frequent sequence. To understand a SPM algorithm, three aspects should be considered: (1) the searching algorithm used in optimizing the searching space, (2) The set of mining target, and (3) the structure of the input database.

4.1.2.1. By Searching Algorithm

There are mainly two approaches in searching the frequent sequence. The first is **Apriori-like** method which is based on the candidate set generation and test approach. Second is the **Pattern Growth** approach which grows the frequent sequence step by step in the projected database. Besides these two, there is also a third approach based on the **Minimum Description Length** (MDL) principal borrowed from data compressing algorithms.

4.1.2.1.1. Apriori-like Approach

With the candidate set generation and test approach, to mine frequent$(k + 1) - sequences$, the algorithm needs to find all the candidate $(k + 1) - sequences$ generated from their previously derived frequent $k - sequences$, scan the database one more time to collect their counts. The essential idea is to iteratively do this process to find longer frequent sequences.

In the pseudo code (Figure 4-2), L_k denotes the set of all frequent $k - sequences$, and C_k the set of candidate $k - sequences$. The set of frequent sequences from the previous pass is used to generate the candidate sequences and then measure their support by making a pass over the database. At the end of the pass, the support of the candidates is used to compare with the $\min\_sup$ to determine the frequent sequences. The Apriori generation function takes L_{k-1} as argument, the set of all frequent $(k - 1) - sequences$. The $q.itemset_{k-1}$ represents the (k-1)th item in the sequence q, the function first joins all the sequences with the same prefix and connects the suffix. Next, delete all sequences $c \in C_k$ such that some $(k - 1) - subsequence$ of c is not in L_{k-1}.

```
Algorithm: Apriori all
Input: transaction database, minimal support
Output: frequent itemset

1: $L_1 \leftarrow \{1 - sequences\}$
2: for $(k \leftarrow 2;\ L_{k-1} \neq \emptyset; k++)$ do
3:     $C_k \leftarrow$ GenerateCandiate($L_{k-1}$)
4:     for each $sequence\ c\ in\ the\ database$ do
5:         $increment\ the\ count\ of\ all\ candidates$
6:         $in\ C_k\ that\ are\ contained\ in\ c$
7:     end for
8:     $L_k = Candidiates\ in\ C_k\ with\ minimum\ support.$
9: end for
10: function GenerateCandiate($L_{k-1}$)
11:     $C_k = \emptyset$
12:     for each $p\ in\ L_{k-1}$ do
13:         for each $q\ in\ L_{k-1}$ do
14:             if $(p.item_1 = q.item_2, \ldots, p.item_{k-2} = q.item_{k-2})$
15:                 $w = p \diamond\ q.item_{k-1}$
16:                 if $(\ \forall\ (k-2)\ length\ sub\ sequence\ of\ w \in L_{k-2})$
17:                     $C_k = C_k \cup w$
18:                 end if
19:             end if
20:         end for
21:     end for
22:end function
```

Figure 4-2 The pseudo code for Apriori-all algorithm

Most of early SPM algorithms are based on the Apriori-like approach. Agrawal first introduced Apriori-like approach into SPM field and three algorithms were provided (Agrawal and Srikant 1995): **AprioriAll**, **AprioriSome** and **DynamicSome**. AprioriAll is designed for mining all the frequent sequences, while AprioriSome and DynamicSome only look for maximal sequences that are not contained in any other frequent sequence. DynamicSome generates candidates in an on-the-fly manner. Srikant and Agrawal (Srikant and Agrawal 1996) generalized their definition of sequential patterns to include time constraints, sliding time window, and user-defined taxonomy, and presented an a priori-based, improved algorithm **GSP** (Generalized Sequential Patterns). Zaki proposed another approach for mining frequent sequential patterns, called **SPADE** (Sequential Pattern Discovery Using Equivalence Classes). This algorithm uses a vertical ID-list database format, i.e., for each item an ID list of the identifiers of the sequences in which it appears and their corresponding time stamps are created. The frequent sequential patterns are mined by performing temporal join operations on these ID lists. SPADE decomposes the original problem into smaller sub problems, which can be independently solved in main memory, using lattice search techniques (Tzvetkov, Yan et al. 2005). All sequences are discovered in only three database scans. Experiments show that SPADE outperforms the best previous algorithm by a factor of two(Zaki 1998).

The Apriori-like sequential pattern mining method bears three nontrivial, inherent costs that are independent of detailed implementation techniques (Han, Pei et al. 2000). First, A huge set of candidate sequences could be generated in a large sequence database. For example, if there are 10^4 frequent $1-itemsets$, the Apriori algorithm will need to generate more than 10^7 length-2 candidates and accumulate and test their occurrence frequencies. Moreover, to discover a frequent pattern of size 100, such as $\{a_1, \dots, a_{100}\}$, it must generate more than 2^{100} candidates in total. Second, multiple scans of databases in mining. Third, The Apriori-based method generates a combinatorial explosive number of candidates when mining long sequential patterns.

4.1.2.1.2. Pattern Growth

In order to overcome the bottleneck of the Apriori-like approach, in 2000, Han first introduced their new method which can mine frequent patterns without candidate generation (Han, Pei et al. 2000). This approach is called pattern growth. The key idea is to avoid the candidate generation step altogether, and to focus the search on a restricted portion of the initial database (Mabroukeh and Ezeife 2010). Instead of repeatedly scanning the entire database and generating and testing large sets of candidate sequences, one can recursively project a sequence database into a set of smaller databases associated with the set of patterns mined so far and, then, mine locally frequent patterns in each projected database (Pei, Han et al. 2004).

Definition 7 **(Sequence extension)** Given a sequence $s = \langle e_1 e_2 \dots e_m \rangle$ and an item α, $s \diamond \alpha$ means s concatenates with α, where α is an item. It can be $I-Step$ extension, $s \diamond_i \alpha = \langle e_1 \dots e_m \cup \{\alpha\} \rangle$ if $\forall k \in e_m, k < \alpha$; or $S-step$ extension, $s \diamond_s \alpha = \langle e_1 \dots e_m \{\alpha\} \rangle$.

Definition 8 **(Prefix)** Suppose all the items within an element are listed alphabetically. Given a sequence $\alpha = \langle e_1 e_2 \dots e_n \rangle$ (where each e_i corresponds to a frequent element in S), a sequence $\beta = \langle e'_1 e'_2 \dots e'_m \rangle (m \le n)$ is called a prefix of α if and only if 1) $e'_i = e_i$ for $(i \le m-1)$; 2)$e'_m \subseteq e_m$; and 3) all the frequent items in $(e_m - e'_m)$ are alphabetically after those in e'_m.

Example 5. Sequence $\langle a \rangle$, $\langle aa \rangle$, $\langle a(ab) \rangle$, and $\langle a(abc) \rangle$ are prefixes of sequence $s = \langle a(abc)(ac)d(cf) \rangle$, but neither $\langle ab \rangle$ nor $\langle a(bc) \rangle$ is considered as a prefix if every item in the prefix $\langle a(abc) \rangle$ of sequence s is frequent in s.

Definition 9 **(Suffix)** Given a sequence $\alpha = \langle e_1 e_2 \dots e_n \rangle$(where each e_i corresponds to a frequent element in s), let $\beta = \langle e_1 e_2 \dots e_{m-1} e'_m \rangle$ be the prefix of α. Sequence $\gamma = \langle e''_m e_{m+1} \dots e_n \rangle$ is called the suffix of α with regards to prefix β, denoted as $\gamma = \alpha/\beta$, where $e''_m = (e_m - e'_m)$. We also denote $\alpha = \beta \cdot \gamma$. Note, if β is not a subsequence of α, the suffix of α with regards to β is empty.

Example 6. For the sequence $s = \langle a(abc)(ac)d(cf) \rangle$, $\langle (abc)(ac)d(cf) \rangle$ is the suffix with regards to the prefix $\langle a \rangle$, $\langle (\_bc)(ac)d(cf) \rangle$, is the suffix with regards to the prefix $\langle aa \rangle$, and $\langle (\_c)(ac)d(cf) \rangle$, is the suffix with regards to the prefix $\langle a(ab) \rangle$.

Definition 10 **(Projected database)** Let α be a sequential pattern in a sequence database S. The α-projected database, denoted as $S|_\alpha$, is the collection of suffixes of sequences in S

with regards to prefix α. Let α and β be two sequential patterns in a sequence database S such that α is a prefix of β. 1. $S|_\beta = (S|_\alpha)|_\beta$, 2. For any sequence γ with prefix α, $support_S(\gamma) = support_{S|_\alpha}(\gamma)$, and 3. the size of α-projected database cannot exceed that of S.

Definition 11 (Support count in projected database) Let α be a sequential pattern in sequence database S, and β be a sequence with prefix α. The support count of β in α-projected database $S|_\alpha$, denoted as $support_{S|_\alpha}(\beta)$, is the number of sequences γ in $S|_\alpha$such that $\beta \sqsubseteq \alpha \cdot \gamma$.

Han first proposed **FP-growth** algorithm, it works in a divide-and-conquer way. The algorithm first compresses the database into a frequent-pattern tree, called FP-tree, which retains the itemset association information. The pattern growth is achieved by the concatenation of the suffix pattern with the frequent patterns generated from the FP-tree recursively. The approach is reported at least an order of magnitude faster than Apriori, and such a margin grows even wider when the frequent patterns grow longer (Han, Pei et al. 2000). Later, **FreeSpan** (Frequent pattern-projected sequential pattern mining) was introduced by Han (Han, Pei et al. 2000). It uses frequent items to recursively project sequence databases into a set of smaller projected databases. The $\alpha - projected\ database$ is considered the collection of frequent sequences having α as a subsequence. Infrequent items and items following those infrequent items in α are ignored. The subsequent mining is limited to each of these smaller projected databases. FreeSpan is significantly more efficient than the GSP. However, the problem of FreeSpan is that the same sequence can be repeated in many projected databases. For example, if a sequential pattern appears in each sequence in the database, its projected database will have the same size as the original database, except for the infrequent items that will be removed.(Tzvetkov, Yan et al. 2005)

In a later work (Pei et al. 2001), Pei introduced **PrefixSpan** (prefix-projected sequential pattern mining). Its general idea is to examine only the prefix subsequences and project only their corresponding suffix subsequences into projected databases. In each projected database, sequential patterns are grown by exploring only local frequent patterns. PrefixSpan runs considerably faster than both GSP and FreeSpan, especially when longer sequential patterns are mined.(Tzvetkov, Yan et al. 2005) The major cost of PrefixSpan is the construction of projected databases. Instead of performing physical projection, one can register the index (or identifier) of the corresponding sequence and the starting position of the projected suffix in the sequence. Then, a physical projection of a sequence is replaced by registering a sequence identifier and the projected position index point. Pseudoprojection reduces the cost of projection substantially when the projected database can fit in main memory.

4.1.2.1.3. Minimum Description Length (MDL) based

One of the major problems in pattern mining is the explosion of the number of patterns. Tight constraints (height $\min\_sup$ value, small $window\ size$) reveal only common knowledge, while loose constraints lead to an explosion in the number of returned patterns. This is caused by large groups of patterns essentially describing the same set of transactions. Algorithm based on the Minimum Description Length (MDL) (Rissanen 1978) principle was shown to be very effective in solving the redundancy issue. MDL is an information-theoretic approach to machine learning, which basically says the best model compresses the data best.

Given a set of models $\mathcal{H}$, the best model $H \in \mathcal{H}$ is the one that minimizes $L(H) + L(\mathcal{D}|H)$ in which $\mathcal{D}$ is the data, $L(H)$ is the information size of the description of H, and $L(\mathcal{D}|H)$ is the information size of the description of the $\mathcal{D}$ when encoded with H. In terms of SPM problem, the target is to find the set of frequent itemsets or sequence that yields the best lossless compression of the database.

The MDL principle was first applied for mining compressing patterns in sequence data (Lam, Mörchen et al. 2014). The algorithm called **SeqKrimp** consists of two phases. In the first phase, a set of candidate patterns is generated by using a frequent closed sequential patterns mining algorithm. In the second phase, the SeqKrimp algorithm chooses a good set of patterns from the set of candidates based upon a greedy procedure. The author also proposed an algorithm called **GoKrimp**, an efficient algorithm that directly mines compressing patterns from the data by greedily extending patterns until no additional compression benefit is observed. Shorter codewords are assigned for small gaps, thus penalizing pattern occurrences with longer gaps. Moreover, by using the Elias code for gaps, it allows to encode interleaved patterns. Meanwhile, in the latter work the **SQS** algorithm proposed a clever encoding scheme punishing large gaps. In doing so, the SQS was able to solve the redundancy issue effectively. At the same time it was able to return meaningful patterns based solely on the MDL principle. In ((Lam, Calders et al. 2013)), researches proposed an algorithm called Zips. By extending the Lempel-Ziv's data compression algorithm, the algorithm is able to process the coming stream online. It defines a new online encoding scheme that allows encoding overlapping patterns.

MDL approach is good at dealing with large databases; however, it also suffers from some problems. First, since this kind of compressing problem is NP-hard and belongs to the class of inapproximable problems (Lam, Mörchen et al. 2014), most of the algorithm could only use the heuristic approach which can't ensure getting the best result. Secondly, the output pattern which compress the database most is relate to length, frequency, the punishment on the intervals and the frequency of the other infrequent patterns. That makes it very difficult to study the relations between the parameters and outcomes.

4.1.2.2. By Searching Target

Previous sequential pattern mining algorithms mine the full set of frequent subsequences satisfying a min_sup threshold in a sequence database. However, since a frequent long sequence contains a combinatorial number of frequent subsequences, such mining will generate an explosive number of frequent subsequences for long patterns, which is prohibitively expensive in both time and space (Han, Afshar et al. 2003).

Many variations are proposed for reducing the number of extracted frequent patterns. Many such methods concentrate on eliminating patterns that are deemed to be non-informative given the other frequent patterns. For example, in the context of transaction datasets, concepts such as closed, Top-k closed sequence, and non-derivable (Gan and Dai 2011) were suggested to reduce the number of frequent itemsets extracted. Similarly, closed sequential patterns were proposed for sequence datasets. Even though such methods result in some reduction in the number of patterns returned by the algorithm, the number of patterns still remains substantial. Also, the redundancy in the set of patterns is, often, still large (Ibrahim, Sastry et al. 2014).

4.1.2.2.1. Closed Frequent Sequence

The closed frequent sequence is the frequent sequence that has no other super sequence that as the same support found in the database. It produces a significantly less number of discovered sequences than the full set. The sequential pattern mining algorithms developed so far have good performance in databases consisting of short frequent sequences. Assuming the database contains only one long 100-frequent sequence , finding the whole pattern will generate $2^{100}-1$ frequent subsequences if the minimum support is 1, although all of them except the longest one are redundant because they have the same support as that of(Han, Afshar et al. 2003). In contrast, the closed frequent sequence only finds the longest one.

Definition 12 (closed frequent sequential pattern) Given database D， let CS be the set of closed frequent sequential pattern, and FS is the set of frequent sequences, $CS = \{\alpha | \alpha \in FS\ \ and\ \ \nexists\beta \in FS\ such\ that\ \alpha \sqsubseteq \beta\ and\ support_D(\alpha) = support_D(\beta)\}$. CS is a subset of FS.

Example 7. If a there is a frequent sequence $\alpha = \langle(ab)(c)\rangle$ and $support_D(a) = 10,$ and another frequent sequence $\beta = \langle(ab)(c)(a)\rangle$ and $support_D(\beta) = 10$, α is not a closed sequence because $\alpha \sqsubseteq \beta$ and $support_D(\alpha) = support_D(\beta)$. However, if the $support_D(a) = 11$, α is a frequent sequent sequence, because $support_D(\alpha) > support_D(\beta)$.

CloSpan (closed sequential patterns mining) is a proposed closed sequence mining algorithm (Han, Afshar et al. 2003). It divides the mining process in to two stages. In the first stage, a candidate set is generated. The second stage helps eliminate non-closed sequence set. It uses a Lexicographic Sequence Tree to record the found frequent sequence and do post-pruning on it. It uses some pruning methods, like CommomPrefix and Backward Sub-Pattern pruning to prune the search space. Because CloSpan needs to maintain the set of historical closed sequence candidates, it will consume much memory and lead to huge search space for pattern closure checking when there are many frequent closed sequences. As a result, it does not scale very well with respect to the number of frequent closed sequences (Jianyong and Jiawei 2004).

Latter, A new paradigm is proposed for mining closed sequences without candidate maintenance, called **BIDE** (BIDirectional Extension) (Jianyong and Jiawei 2004). The forward directional extension is used to grow the prefix patterns and also check the closure of prefix patterns, while the backward directional extension, the BackScan pruning method, can be used to both check closure of a prefix pattern and prune the search space. The ScanSkip optimization technique is proposed to speed up the mining and also assure the correctness of the algorithm.

The study shows that it can be an order of magnitude faster than CloSpan but only uses order(s) of magnitude less memory in many cases. It also has very good scalability w.r.t. the database size.

4.1.2.2.2. Top-k Sequence

Most of SPM algorithms require the specification of a minimal support to define the frequency. However, in practice, addition knowledge is needed to decide a threshold. To overcome this difficulty, (Tzvetkov, Yan et al. 2005) proposed an alternative mining task: mining top-k frequent closed sequential patterns. Instead of providing a minimal support, user provides the minimal length of each pattern, starting at min_sup = 1, and the algorithm makes use of the length constraint and the properties of top-k closed sequential patterns to perform dynamic support raising and projected database pruning.

Definition 13 (top-k closed sequential pattern) A closed sequential pattern s is a $top-k\ closed\ sequential\ pattern$ of **minimal** length min_lif there exist no more than $(k\ -1)$ closed sequential patterns for which the length is at least min_l and for which support is higher than that of s.

TFP (Tzvetkov, Yan et al. 2005) is an FP-tree based frequent pattern-mining algorithm for finding the top-k frequent closed patterns without a predefined min_sup threshold. TFP starts the mining process with min_sup threshold equal to 1 and raises the support threshold during both the FP-tree construction and the mining of the FP-tree. TFP explores top-down and bottom-up combined FP-tree mining processes to first mine the most promising tree branches. Also, an efficient closure verification scheme is developed to determine whether the newly discovered patterns are closed..

4.1.2.3. By Different Database

There are basically three kinds of databases: temporal database, transactional database, and sequence database. Transactional database is mainly for frequent itemset mining, while temporal database and sequence database are for sequential pattern mining. Mining frequent sequences on temporal database also called Frequent Episode Mining.

These three databases are interchangeable. Temporal database can be transformed to a transaction database where each transaction is the union of events from sliding window. Temporal database can also be transformed to sequence database, where each sequence is the serial combination of events from sliding window. However, such method is not efficient, since the size of the database will increase.

Moreover, there is a special kind of database combines transactional database and sequence database, called **Multi-dimensional sequence database**. In such kind of database, sequence patterns can be associated with different circumstances, and such circumstances form a multiple dimensional space. For example, customer purchase sequences are associated with region, time, customer group, and others. It is interesting and useful to mine sequential patterns associated with multi-dimensional information. In (Pinto, Han et al. 2001), the author proposed two algorithms **Seq-Dim** and **Dim-seq** for mining multi-dimensional sequence.

4.1.2.4. By Real-Time

Most of the SPM algorithms are designed for mining in static database. However, online and dynamic analysis is needed for many data streams like a stream of HTTP requests received by a Web server, since the streams change over time. Over an online data stream, new data elements are generated and appended continuously and rapidly. Thus, the recently generated stream segment may reflect the latest information and recent trends of the stream, and old data elements before a recent time point may become obsolete. Due to the dynamic evolution of the stream, frequencies of patterns may change over time, and the frequency of a pattern may vary dramatically in different time periods. Since the SPM is computational expensive, to rescan the whole database whenever new data comes will waste a lot of computation. In (Gan and Dai 2012), researchers investigated online mining of recently frequent sequences over data streams. Based on frequency metric T-freq and FE- bases, they abstracted the mining problem as two basic procedures called GFE-B-Update and RFE-B-Deduction. An efficient one-pass method, called **RFE-B-Miner**, was proposed for online mining of RFE-bases adaptively.

4.1.3. Example: Mining with BIDE

The BIDE algorithm is realized in JAVA, and a small demonstration is given to get an intuitive feeling of the process of the searching. Given a set of items $= \{1, 2, 3, 4, 5, 6, 7\}$, the sequence database in (Table 4-5) , and $min\_sup \; = \; 3$. the closed frequent sequences will be found in the end.

Sequence ID	Sequence
1	$\langle(1)(1\ 2\ 3)(1\ 3)(4)(3\ 6)\rangle$
2	$\langle(1\ 4\ )(3)(2\ 3)(1\ 5)\rangle$
3	$\langle(5\ 6\ )(1\ 2)(4\ 6)(3)(2)\rangle$
4	$\langle(5)(7)(1\ 6)(3)(2)(3)\rangle$

Table 4-5 The input sequence database

The algorithm will first scan the database once to get the frequent 1-sequences. $\langle(1)\rangle{:}\,4, \langle(2)\rangle{:}\,4, \langle(3)\rangle{:}\,4, \langle(4)\rangle{:}\,3, \langle(5)\rangle{:}\,3, \langle(6)\rangle{:}\,3$, while $\langle(7)\rangle{:}\,1$ is discarded for the low frequency. Then the pseudo projected database for each sequence in frequent 1-sequence set will be generated. For example, $S|_{\langle(3)\rangle}$ is listed in (Table 4-6). Each frequent 1-sequence is treated as a prefix and uses BackScan pruning method to check if it can be pruned. Take prefix $\langle(3)\rangle$ for example, it will find (1) as its backward-extension-item, which has the same support, equals 4. So, the prefix $\langle(1)\rangle$ in 1-sequences can be pruned, and these two prefixes are merged into $\langle(1)(3)\rangle$. Instead of searching both $S|_{\langle(1)\rangle}$ and $S|_{\langle(3)\rangle}$, the algorithm only needs to search $S|_{\langle(1)(3)\rangle}$, which is same as $S|_{\langle(3)\rangle}$. In $|_{\langle(3)\rangle}$, the support of each item is: $\langle(1)\rangle{:}\,2, \langle(2)\rangle{:}\,3, \langle(3)\rangle{:}\,3, \langle(4)\rangle{:}\,1, \langle(5)\rangle{:}\,1, \langle(6)\rangle{:}\,1$. Only $\langle(2)\rangle$ and $\langle(3)\rangle$ are local frequent items. The prefix will be extended to $\langle(1)(3)(2)\rangle$ and $\langle(1)(3)(3)\rangle$, corresponding pseudo projected database, $S|_{\langle(1)(3)(2)\rangle}$ (Table 4-7)and $S|_{\langle(1)(3)(3)\rangle}$, will be generated again for searching next local frequent item to extend the pattern. However, the local frequency of each item in $S|_{\langle(1)(3)(2)\rangle}$ is: $\langle(\_3)\rangle: 1, \langle(1)\rangle: 1, \langle(3)\rangle: 1, \langle(5)\rangle: 1$, all non-frequency. So the searching is terminated.

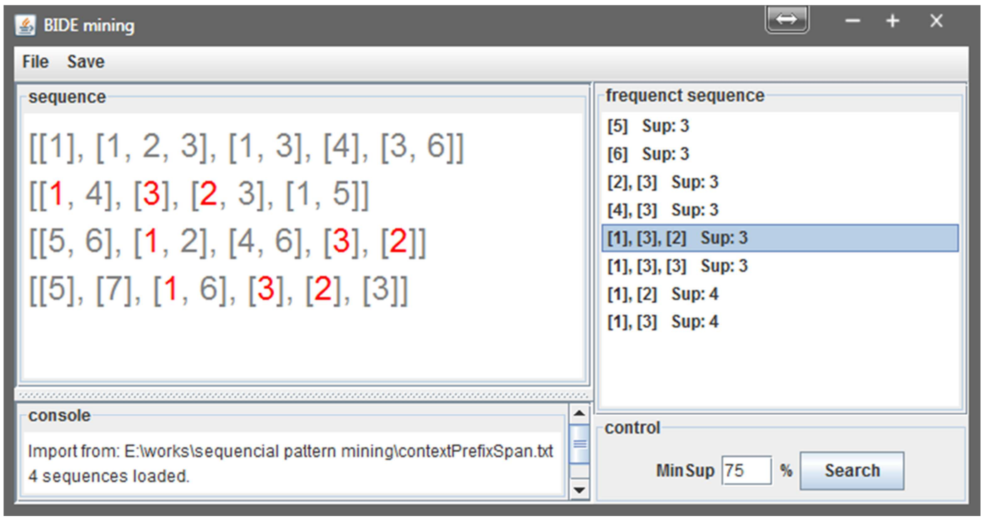
Sequence ID	Sequence
1	$\langle(1\ 3)(4)(3\ 6)\rangle$
2	$\langle(2\ 3)(1\ 5)\rangle$
3	$\langle(2)\rangle$
4	$\langle(2)(3)\rangle$

Table 4-6 The pseudo projected database for <(3)>

Sequence ID	Sequence
1	$\langle\emptyset\rangle$
2	$\langle(\_3)(1\ 5)\rangle$
3	$\langle\emptyset\rangle$
4	$\langle(3)\rangle$

Table 4-7 The pseudo projected database for <(1)(3)(2)>

Eight different closed frequent sequential patterns are founded at the end (Figure 4-3). Occurrences of the selected patterns are highlighted in the sequence panel. Although the pattern $\langle(1)(3)\rangle$ is a subsequence of $\langle(1)(3)(2)\rangle$, its support value is higher than $\langle(1)(3)(2)\rangle$, so it is also a closed frequent sequence.



```
[[1], [1, 2, 3], [1, 3], [4], [3, 6]]
[[1, 4], [3], [2, 3], [1, 5]]
[[5, 6], [1, 2], [4, 6], [3], [2]]
[[5], [7], [1, 6], [3], [2], [3]]
```

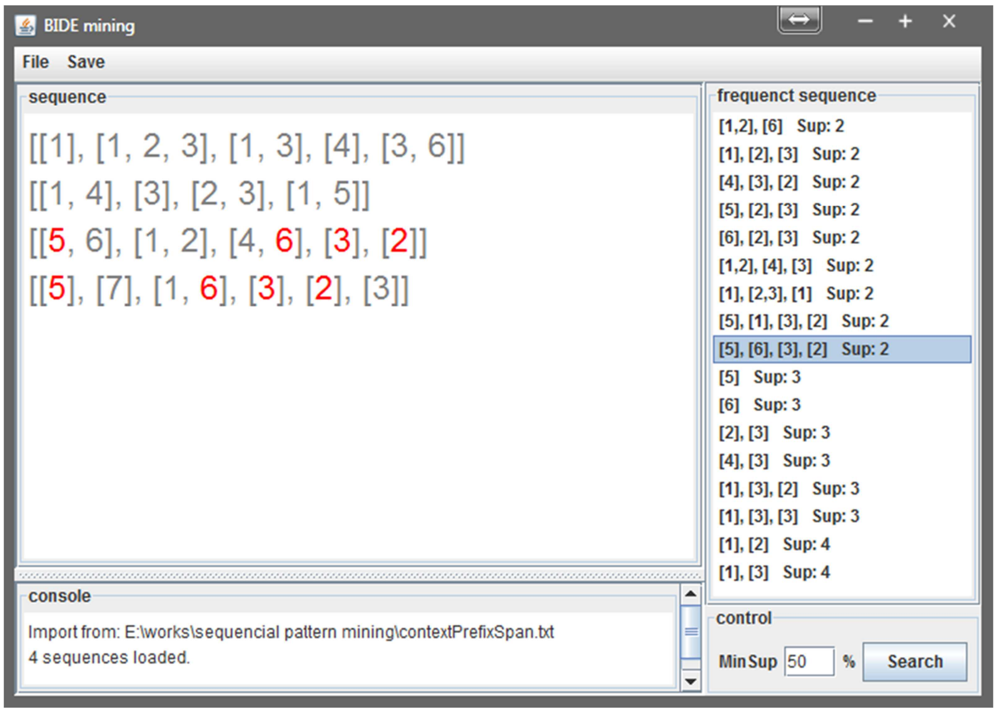
```
[5]   Sup: 3
[6]   Sup: 3
[2], [3]   Sup: 3
[4], [3]   Sup: 3
[1], [3], [2]   Sup: 3
[1], [3], [3]   Sup: 3
[1], [2]   Sup: 4
[1], [3]   Sup: 4
```

```
Import from: E:\works\sequencial pattern mining\contextPrefixSpan.txt
4 sequences loaded.
```

Min Sup 75 %

Figure 4-3 Closed frequent sequential patterns found by BIDE with 75 % support

When the min_sup is down to 50%, i.e. 2 occurrences, more and longer frequent sequential patterns are found (Figure 4-4).



```
[[1], [1, 2, 3], [1, 3], [4], [3, 6]]
[[1, 4], [3], [2, 3], [1, 5]]
[[5, 6], [1, 2], [4, 6], [3], [2]]
[[5], [7], [1, 6], [3], [2], [3]]
```

```
[1,2], [6]   Sup: 2
[1], [2], [3]   Sup: 2
[4], [3], [2]   Sup: 2
[5], [2], [3]   Sup: 2
[6], [2], [3]   Sup: 2
[1,2], [4], [3]   Sup: 2
[1], [2,3], [1]   Sup: 2
[5], [1], [3], [2]   Sup: 2
[5], [6], [3], [2]   Sup: 2
[5]   Sup: 3
[6]   Sup: 3
[2], [3]   Sup: 3
[4], [3]   Sup: 3
[1], [3], [2]   Sup: 3
[1], [3], [3]   Sup: 3
[1], [2]   Sup: 4
[1], [3]   Sup: 4
```

```
Import from: E:\works\sequencial pattern mining\contextPrefixSpan.txt
4 sequences loaded.
```

Figure 4-4 Closed frequent sequential patterns found by BIDE with 50 % support

4.1.4. Problems of SPM Algorithms

Although different optimizations have been proposed to improve the performance of SPM, there are still some limits haven’t been conquered.

- **Huge searching space**

  The SPM is performs an exhaustive search in the database. Although, the down closure property can help in reducing the searching space, the searching space still grows exponentially when the database size and itemset size increase. The searching space is also decided by the character of the patterns in the database, i.e. density, length, gaps. The high performance achieved by down closure property is based on the sparsity of the pattern. If the pattern density is very high, then the optimization is limited. Since, the character of the patterns is unknown information, it is impossible to predict the computational load before the mining.

- **Redundant results**

  The bottleneck of frequent pattern mining is not on whether the complete set of frequent patterns under certain constraints can be found efficiently, but on whether a compact but high quality set of patterns that are most useful in applications can be derived. There are proposals on reduction of such a huge set, including closed patterns, maximal patterns, approximate patterns, condensed pattern bases, representative patterns, clustered patterns, and discriminative frequent patterns, but it is still not clear what kind of patterns will give satisfactory pattern sets in both compactness and representative quality for a particular application.

- **Information lost during mining**

  To reduce the calculation load, most of the information in the original database is neglected. Everything is simplified into items, symbols that represent real world value. Also, all the algorithms only record the amount of the occurrence of the patterns without recording the locations. The abstraction and simplification of the data cause some difficulties in deep understanding and interpretation of patterns, such as semantic annotation for frequent pattern, and contextual analysis. For the data sets that contain structural information, such as associated attributes, linking this information to the patterns will provide more valuable analysis.

- **The missing of rare events**

  SPM only looks for the patterns that occur more than min_sup, the patterns lower than this threshold will be discard. However, in the real world, events happening less doesn't mean they are not interesting. If there are very strong correlations between the some rare events, they are also worth being investigated.

- **Discrete value only**

  All of the SPM can only deal with discrete values, which can be easily converted into items. When facing the continuous value, it is not feasible to assign each difficult value an item, because it will dramatically increase the size of item types, which will then increase the searching space. So, it has to rely on some other cluster algorithms to cluster the continuous value into groups, i.e. discretize the value, which involves a lot of parameter setting and prior knowledge of the original data.

4.1.5. Frequent Episode Mining

Mining frequent sequence in sequence database is called SPM, while mining frequent sequence in temporal database is called Frequent Episode Mining. The frequent sequence here is called **Frequent Episode (FE)**. The discovered FEs have been applied to many areas, such as telecommunication alarm management, intrusion detection, discovery of relation between financial events and stock trends and gene analysis (Gan and Dai 2012). Although FEM derives from SPM and shares a lot of similarities, there are still some differences in terminology and algorithm.

4.1.5.1. Three Classes of Episodes

Mannila introduced three classes of episodes (Mannila, Toivonen et al. 1997): serial episodes, parallel episodes, and composite episodes (Figure 4-5). Serial episodes consider patterns of a total order in the sequence, while parallel episodes have no constraints on the relative order of event sets. The third class contains composite episodes, like serial combination of parallel episodes. Serial and parallel episodes can be captured by sequential patterns and frequent itemsets, respectively. Frequent itemsets for transaction databases are similar to parallel episodes, while sequential patterns for sequence databases are similar to serial episodes.

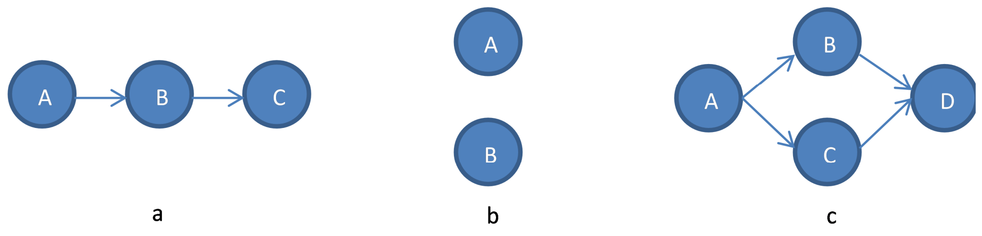


Figure 4-5 Three classes of episode a) serial episode, b) parallel episode, c) composite episode

4.1.5.2. Frequency Metric

One essential difference in mining the temporal database and sequence database is the frequency metric, i.e. the amount of a pattern's occurrence in the temporal database. In transaction database and sequential database, the frequency measure is straightforward. It is the amount of the transactions or sequences that contain the pattern. In contrast, there is only one sequence in temporal database, where s different metric can be adopted for measuring the occurrence. There are several typical frequency metrics proposed by researchers. Consequently, using different measures on the same temporal database, a different support value will be found. However, no measures have been commonly accepted in temporal database mining. In the paper (Min and Honghua 2010), the author suggested three major properties of the frequent metric: **anti-monotonicity**, **maximum-frequency**, and **window-width restriction**. He also name the existing metrics, such as **fixed-win-freq** (Mannila, Toivonen et al. 1995), **mo-freq** (Mannila and Toivonen 1996), **auto-win-growth-freq**, **maxgap-mo-freq**, **T-freq** (Iwanuma, Ishihara et al. 2005), **non-overlapped-freq** and **distinct-bound-st-freq** respectively. Besides that he also proposed a measure technique called **LMaxnR-freq**, which is short for the **Leftmost Maximal non-Redundant set of occurrences**.

Definition 14 (Anti-monotonicity) Frequency measure $freq'$ is anti-monotonic if for any two episodes α and β and any single event sequence S, $freq'(S,\beta) \leq freq'(S,\alpha)$ when $\alpha \sqsubseteq \beta$, where $freq'(S,\alpha)$ represents the frequency of α in S under $freq'$.

Anti-monotonicity is another expression of the **downward-closure property** in frequent episode mining. The anti-monotonicity principle guarantees that a pattern can be pruned safely if any of its sub-patterns is infrequent.

Definition 15 (Maximum-Frequency) Given a restriction of window width w, an anti-monotonic measure, $freq'$, satisfies the maximum-frequency principle, if for $\forall\alpha$ and $\forall$S, $\neg\exists freq*$, s.t. $freq*$ has the same restriction of window width, $freq*$ is anti-monotonic and $freq*(S,\alpha,w) > freq'(S,\alpha,w)$.

The maximum-frequency property ensures that no proper occurrences are missed in the computation of frequency. By comparing the performance of these different frequency metrics, Min examined the effects of the three properties. Min found that the variant window width is more reasonable than fixed window width. Anti-monotonicity is necessary but can't guarantee the accuracy. Maximum-frequency has impact on the accuracy. These existing frequency metrics may include false patterns or incorrect patterns.

To provide a more accurate frequency measure, Min recommended a max_gap based restriction of window width, a new principle called strict anti-monotonicity, and a new frequency measure that satisfies those requirements.

Definition 16 (Max_gap Window Width) Given $S = \langle (e_1)_1 (e_2)_2 \dots (e_n)_n \rangle$ and $\max\_gap$, for any episode $\alpha = \langle a_1 a_2 \dots a_m \rangle$, and its occurrences $o = \langle i_1 i_2 \dots i_m \rangle$, is called an occurrence of α in S w.r.t. $\max\_gap$ iff it satisfies 1) $e_{i_j} = a_j$ for $j = 1, 2, \dots, m$, 2) $1 \leq i_1$ and $i_m \leq n$, and 3) $i_j - i_{j-1} \leq \max\_gap$ for $j = 2, 3, \dots m$.

Example 8. Given episode $\alpha = \langle (ab)(b)(d) \rangle$, $\max\_gap = 3$ and three different sequences, $S_1 = \langle (ab)_1 (e)_2 (f)_3 (b)_4 (f)_5 (ed)_6 \rangle$, $S_2 = \langle (ab)_1 (c)_2 (e)_3 (f)_4 (b)_5 (f)_6 (ed)_7 \rangle$, $S_2 = \langle (ab)_1 (ce)_2 (f)_3 (b)_4 (f)_5 (ed)_6 \rangle$. α occurs in S_1 and S_3, but not S_2, because the timestamp gap between the (ab) and (b) is $5 - 1 = 4 > max\_gap$ in S_2.

Definition 17 (Strict Anti-monotonicity) A frequency measure, $freq'$, is strictly anti-monotonic if (i) it is antimonotonic, and (ii) given any S, any α and $\max\_gap = k$, $freq\,'(S, \alpha, k) \leq \min_{\forall \beta \sqsubseteq \alpha, \forall OS \in MaxnR-O(S,\beta,k)}(|OS|)$

Definition 18 (Strict Maximum-Frequency) Given $\max\_gap = k$, a strict anti-monotonic measure, $freq'$, satisfies the strict maximum-frequency principle, if for $\forall \alpha$ and $\forall S$, $\neg \exists freq *$, s.t. $freq *$ has the same restriction of $\max\_gap$, $freq *$ is strict anti-monotonic and $freq * (S, \alpha, k) > freq' (S, \alpha, k)$.

4.1.6. FPM in Smart Home research

In smart home research, depending on the form of the database and mining target, frequent pattern mining algorithms can be adapted to mining smart home data in a variety of ways. The input database could be a temporal database that contains only one long sequence or pre segmented into a certain time interval. A pattern could be a sequential labeled ADL events or on/off log of sensors that occur periodically in smart homes.

In MavHome project (Das, Cook et al. 2002), frequent itemsets mining technique was used in mining the ordered or unordered home devices activity sequences, which is called Episode Discovery. It can also recognize regular basis (daily, weekly) activities. The algorithm first divides the input sequence into transaction-like collections of events by sliding window. Candidate frequent itemsets are created by frequent itemset mining process. The MDL principle is introduced to evaluate these potential sequences. Candidate itemset that minimizes the description length of the input sequence will be identified as a significant episode. A synthetic 28-day data set was generated to validate this algorithm.

Guralnik developed an automated monitoring and caregiving system called Independent Life Style Assistant ((I.L.S.A.) to learn models of human behavior (Guralnik and Haigh 2002). In that context, a sequential pattern is a list of sensor firings ordered in time. Besides mining the frequent sequential pattern, they also extend the basic approach to incorporate reasoning about the time of the activities. Each sensor activity is assigned with a time interval calculated based on the distribution of sensor firings during the day, such as Bedroom motion [6:00-7:00], Kitchen motion [12:00-13:00], Kitchen motion [13:00-14:00]. After the time intervals are determined, the list of daily event sequences is translated into a data set suitable for learning sequential patterns. A tree projection sequential patterns algorithm was used in finding the frequent patterns. Additionally, the frequent sequential patterns have to be post processed with three filters to remove the redundant information. The data was collected from the I.L.S.A. systems installed in the homes of four of the project engineers within 1 to 2 months.

In the paper(Lühr, West et al. 2007) ,researchers introduced a data mining approach for the detection of new and changing behavior in people living in a smart home. They presented the EFP-Tree, an extension of the FP-Tree for inter transaction mining, and the corresponding EFP-Growth algorithm for the efficient retrieval of frequent inter transaction association rule (IAR) patterns from the tree structure. A method for finding IAR patterns exhibiting significant growth from one data set to another was also introduced. These emergent IARs allow researchers to detect the presence of new, possibly anomalous, behavior and unusually frequent occurrences of behavior that would otherwise be considered normal from amongst the mined rules.

In Center for Advanced Studies in Adaptive Systems (CASAS) smart home project, a mining method, called Discontinuous Varied-Order Sequential Miner (DVSM) is introduced, which is able to find frequent patterns that may be discontinuous and might have variability in the ordering. This algorithm performs frequent sequence mining using DVSM to discover frequent patterns, and then, groups the similar discovered patterns into clusters. Unlike many other sequence mining algorithms, they report a general pattern that comprises all frequent variations of a single pattern that occur in the input data set.

In the paper (Nazerfard, Rashidi et al. 2011), researchers proposed a framework, called “TEREDA”, short for “TEmporal RElation Discovery of Daily Activities”, for discovering and representing temporal aspects of activity patterns, including temporal ordering of activities and their usual start time and duration. TEREDA discovers the order relation between different activities using temporal association rule mining techniques. The same activity will be clustered in to different clusters by using the Expectation Maximization (EM) clustering method. So not only the association rule of activities but the temporal relations will be extracted from the data.

In project (Fatima, Fahim et al. 2013), they extracted the behavioral pattern from the day to day performed activities in a sequential manner. The SPAM sequential pattern mining algorithm was applied by modifying it according to the requirements of behavior modeling from the activity log. In the proposed framework, each sequence is a set of activities performed in a temporal order of three days for consistent sequence prediction.

Table 4-8 FPM in smart home research

Project name	Mining target	Database type	Mining algorithm	environment
MavHome project (Das, Cook et al. 2002)	Frequent activities and regular basis (daily, weekly) activities	Transaction-like collections of events by sliding a window over the event history and viewing the collection of events within the window as an unordered set.	Frequent itemsets mining	A synthetic 28-day data set was generated reflecting MavHome scenario.
I.L.S.A. behavior (Guralnik and Haigh 2002)	Frequent sequence with time interval	Sequence database , each activity is assigned with time interval.	A tree projection sequential patterns algorithm	The data was collected from the I.L.S.A. systems installed in the homes of four of the project engineers within 1~2 month
(Lühr, West et al. 2007)	Emergent IARs, those rules that display significant growth from one data set to another	Sequence database.	EFP-Tree, an extension of the FP-Tree	Two home, 77 and 84 sensors each, 16 days, 658 and 748 transactions each
(Rashidi, Cook et al. 2011)	Continuous, discontinuous and have varied orders	Transaction database of movement sensor data.	Discontinuous Varied-Order Sequential Miner (DVSM)	20 volunteers in the smart apartment testbed includes three bedrooms, one bathroom, a kitchen, and a living room equipped with motion sensors.
(Lymberopoulos, Bamis et al. 2011)	Automatically extracts the person's daily pattern.	a sequence of triplets which contains sensor data, timestamp and duration .	Apriori like sequential pattern mining algorithm	30-day dataset extracted from an ongoing deployment of a sensor network inside a home monitoring an elder.
Relations of Daily Activities(Nazerfard, Rashidi et al. 2011)	Temporal relations such as order of activities, as well as their corresponding start time and duration features.	Sequence of daily activities, associated with start time and duration.		Four months of real data collected from a smart home more than 480, 000 sensor events, motion sensors placed on the ceilings and walls, as well as on doors and cabinets.
(Moutacalli, Bouzouane et al. 2012)	create unsupervised and personalized model activities	Sequence of daily activities clustered with start time.	BIDE based sequential pattern mining	Data recorded by observing an occupant for 28 days, using 14 sensors which detected seven distinct activities.
(Fatima, Fahim et al. 2013)	person's daily pattern	Sequence database, each sequence contains 3 days of activity labels .	SPAM sequential pattern mining	Two real datasets from the CASAS smart home

These experiments show the possibility of mining ADLs with FEM algorithms. There are mainly two different approaches to apply FEM. The first approach is direct **mining on sensor data**, in this approach all of the experiments only use discrete values as the input dataset, such as on/off of the motion sensor, open close of the windows, etc. because they are easier to be converted into symbolic representation. However, in real life, the continuous values, i.e. temperature, humidity, etc., play an important role in occupants' decision making. For example, if the room temperature is higher than outside for a certain degree in summer, then the occupant is more likely to open the window. The other approach is to map the senor data with ADLs first, and then do the **mining on the ADLs**. The problem of this approach is the mapping process is not 100% correct. The correctness in controlled experimental environments with a limited amount of activities can only reach around 80% correctness, not to mention the real life environment. Mining on these unreliable data will cause unreliable results. Another common problem is that these papers only demonstrated the capability in mining the patterns, but haven't provided enough evaluations of the quality. Although some paper claims the prediction based on the frequent patterns can reach very high precision, but no information about the characteristics of the input data is provided. Because in the found patterns, there could be a lot of redundant patterns only reveal some common sense, such as open a door will be followed by close the door. These kinds of patterns are always right, but have no value.

4.2. Features of the Proposed Algorithm

A new algorithm will be introduced to solve deficiencies of the existing algorithms and make the mining process more practical. **The proposed algorithm is able to extract closed frequent episodes on the complex temporal database collected from heterogeneous sensors in a smart home**. As an unsupervised algorithm, it tries to mine on the minimal data, thus bring down the computational complexity and also power consumption of the sensor nodes. By using the prior knowledge of ADLs temperately, the algorithm is able to get more meaningful results without losing generality. Additionally, the user is also possible to specify an event and back track the sequence to lead to it. Here are some unique features of the algorithm.

- **Both numerical and categorical value**

  Typical FPM algorithms only can deal with categorical value. Researchers usually use English letters ("a", "b", "c") or ordinal numbers ("1", "2", "3") as symbolic representation of different items, such as the different web pages or the items in the shopping list, just for demonstration. However, the smart home sensor data consists of both numerical and categorical data. Most of the time, the change of the numerical values will cause a change in status. For example, if the room temperature goes up to a certain degree, then the occupants will probably turn on the air conditioner or open the window. If the numerical values are neglected, a lot of valuable patterns will be missed. We can certainly assign each numerical value with a unique symbolic value. But it will create a large amount of items, which will increase the searching space dramatically. Another problem is that the algorithm lose its generality and less patterns will be found. For example, the room temperature value, 33.5℃ and 33.6 , will be represented with different symbols, thus, sequence

$\langle \text{temperature up to } 33.5℃, \text{air conditioner on} \rangle$ and $\langle \text{temperature up to } 33.6℃, \text{air conditioner on} \rangle$ will be regarded as two different sequences. It is also possible to divide the values into to several regions base on the distance or amount, such as $(25℃, 30℃]$, $(30℃, 35℃]$… However, without prior knowledge, it is very hard to decide the amount and size of each region. So, an unsupervised cluster algorithm can be a solution to cluster the data into several groups. There are several different algorithms that can be choosen from, but K-Means needs the number of cluster and SOM can't keep the consistency of outputs. An algorithm called **DBSCAN (Density-Based Spatial Clustering of Applications with Noise)** is selected, since it clusters the data based on the density of the data points in the space, and only needs to specify the largest distance to the neighbor in the group. In the temperature case, we only need to decide the resolution of the measurement of the temperature, which is easier than deciding the amount of cluster. More details will be discussed in next chapter.

- **Pattern filter**

  A list of several thousand patterns is only marginally more useful than raw data. In order to achieve concise results and better utility, several techniques have been applied in the mining process. First, the mining target is a closed episode, a large amount of closed sub episodes are discarded. Secondly, a maximal gap constrain is applied, which can help remove a lot frequent episodes with large gaps in between each element. Thirdly, approximate closed calculation is implemented to calculated closed episodes. This approach can give a tolerance when comparing the frequency between an episode and its sub episode, which can help to remove those sub episodes with very similar support.

- **Temporal and ambient information**

  In standard episode mining algorithms, the temporal information is only used to sort the element in the right sequential order. The same event detected by a sensor will be assigned with the same symbol. However, in real life, a person usually performs each daily activity during a certain time period. Temporal information is very useful for understanding the context information. For example, the opening of the front door around 8:00 A.M. is a strong indication that the occupant is leaving for work, when it happens around 6:00 P.M., it probably means the occupant is back from work. If we can associate this information with the sensor events, the episode will be more specific and meaningful. Although, this approach will create more items, it won't increase the computational complexity too much, because people usually do their daily routine at a similar time and temporal information can be regarded as numerical data. The temporal information can also be clustered in to several groups with the DBSCAN algorithm. Take the front door case for example. If the occupant usually leaves home at 8:00 A.M. and comes back at 6:00 P.M., it will be clustered into three groups: around 8:00 A.M., around 6:00 P.M., and a noise group that can't be clustered. Thus, only two additional items are added. Since the noise cluster won't be frequent, it will be discarded in the early stage of searching, which will help to reduce the searching space.

- **Reversible data structure**

Studies of algorithm performance usually simplify the data structure for better performance and easier programming and evaluation. The outputs will be only the symbolic representations and their support values of the actual frequent events. No additional information, like the actual occurrences of the episode and gap size of each element, in the sequence is provided. It causes some difficulties for evaluating the episode. For example, pattern A happened n times in one day and pattern B happened n times in n different days will have same support value n. However, pattern B is more meaningful than pattern A, since B happened daily and A was casually happened. In fact, this kind of occurrence information was calculated but discarded during the mining process for a simpler data structure. In the proposed algorithm, a more sophisticated data structure is designed to preserve this information. It's able to backtrack when, where, and how these patterns happen and also visualize these patterns.

- **User specified event**

  The limit of FPM algorithms is that it only finds the patterns that occur more frequently than the user specified threshold. The sequence that happens less than the threshold will be neglected. However, in real life, a rarely happening event sequence doesn't mean it is useless. For example, if there is a sequence A occurs 10 times in the database and every time it will lead to event a, and there is another same length sequence B occurs 100 times, but it leads to event a 90 times, and other 10 times it end with event b. So, it is really hard to tell which pattern is more important. Another drawback is that FPM will find out all frequent patterns. However, in smart home research, not all events are interesting. If researchers only want to know what will lead to the opening of front door, those frequent patterns without this event are redundant in this case. To overcome this kind of problem, the proposed algorithm allows users to specify the event they are interested in, and do a backward search to find the sequences that will end with this event.

- **Complex sequence**

  Since frequent episodes were introduced, several typical approaches have been proposed to discover frequent episodes. Most approaches only focus on frequent episodes mining from simple sequences. However, in a smart home, complex sequences need to be considered when multiple events may appear simultaneously or states of other sensors are needed when an event takes place. For example, the opening of window will be decided by both room temperature and outdoor temperature. If room temperature is high, and outdoor temperature is low, the occupant may open the window for cooling down. If the outdoor temperature is even higher, then the occupant is more likely turn on the air conditioner.

4.3. Introduction of the Proposed Algorithm

4.3.1. The Architecture of the Algorithm

The proposed algorithm is able to read sensor data from **Database Management system (DBMS)**, extract frequent episodes from the sequence and finally visualize their occurrences in the chart. Correspondingly, the whole architecture can be divided into three main parts: **data**

pre-processing module, **pattern mining module** and **visualization module** (Figure 4-6). The data pre-processing module provides functionalities to read and write with the DBM. It can convert different unformatted sensors data sources into the format that accepted by the mining algorithm and push them into the DBM. It can also read the formatted data from the DBM, assign each data value with corresponding symbolic representation and create the complex temporal sequence with these symbols. The pattern mining module consists two different algorithms: one for extracting all closed frequent episodes, the other for searching the episodes with user specified target. The visualization module generates charts for the sensor data table, symbolic data, the episode enumeration tree, and the trace of the frequent episodes in charts.

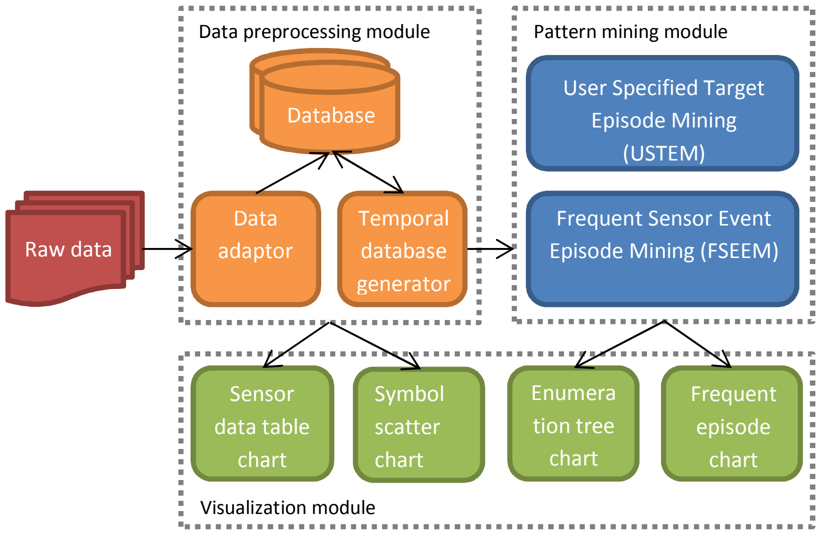


Figure 4-6 The architecture of the proposed algorithm

4.3.2. The Data Preprocessing Module

There are mainly three components in the data pre-processing module: **data adaptor**, **DBM** and **sequence generator**.

There is no standard for the format of smart home sensor data. Data are recorded in different forms according to the ways they were collected and the needs of different projects. It causes lots of difficulties for researchers in exchanging their data source. To make use of the data from other projects, adapters are needed in the first place to connect to different data sources.

DBM is used to manage not only sensor data but also metadata of sensors and background information of the project. Since different data sources contain different information, and the formatted data is not only for generating the temporal database but also for other statistics and visualization, the data structure in the database should have the generality to cover a variety of information. Then, the sequence generator converts the formatted data in the database into the form of temporal database.

4.3.2.1. The Structural of Database

MySQL is selected as the DBM for this algorithm. It is a well-known open-source general purpose relational DBM, which is widely used in web application. The hierarchical data structure in MySQL is organized as **database**, **table**, **column,** and **row**. Database is the top-level structure to distinguish the data for different projects. In database all data are organized in the form of table. A table is a linked two-dimensional data structure, which is comprised of columns and rows. A column, also called attribute, is a set of data values of a particular simple data type, and each row would provide a data value for each column. One of these columns can be selected as **primary key column**. Primary key is the unique identification for each row in the table, thus all the value in the primary key column should be unique.

In the case of a smart home, each data from different smart home projects are separated into different databases. In each database, there are three kinds of tables to record different information: **sensor data table**, **sensor meta data table,** and **project data table**.

- **Sensor data table**

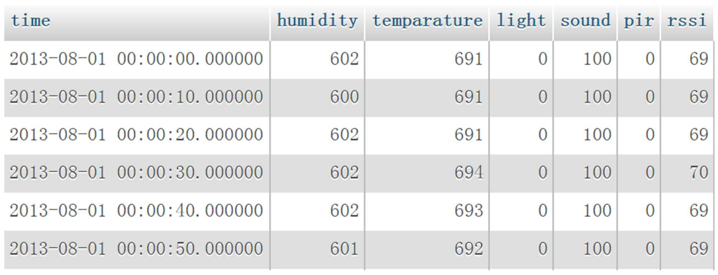
  Sensor data table holds data samples from a certain sensor. Each sensor node has its corresponding table. The table is named after the identification of the sensor node. In the table, the primary key column is the date and time when the sample is recorded. The rest of columns are for the sensor data values. The amount of columns is decided by the amount of sensor integrated on the sensor node. Raw data are directly recorded in the table (Figure 4-7).

- **Sensor metadata**

  The sensor metadata table records all descriptions of the sensor nodes, such as the location of sensor node, the room or appliance the sensor node is attached to, and the amount and version of sensors on the node. With this information, the spatial relationships among the sensors and rooms are preserved. This helps to link and event sensor data with ambient sensor data. For example, the on/off of the room light is depending on the ambient brightness inside the room, the location of the occupant, and also the time of the day. Since only the raw data are record in the database, the model and version of the sensor are needed to find the datasheets for the calibration and mapping.

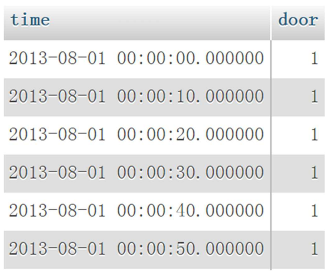
- **Building profile**

  The building profile includes the background information of the test bed, such as the location and orientation of the building, the layout of the rooms, the number of occupants, the sampling rate, and total amount of the samples, etc.



time	humidity	temparature	light	sound	pir	rssi
2013-08-01 00:00:00.000000	602	691	0	100	0	69
2013-08-01 00:00:10.000000	600	691	0	100	0	69
2013-08-01 00:00:20.000000	602	691	0	100	0	69
2013-08-01 00:00:30.000000	602	694	0	100	0	70
2013-08-01 00:00:40.000000	602	693	0	100	0	69
2013-08-01 00:00:50.000000	601	692	0	100	0	69

(a)



time	door
2013-08-01 00:00:00.000000	1
2013-08-01 00:00:10.000000	1
2013-08-01 00:00:20.000000	1
2013-08-01 00:00:30.000000	1
2013-08-01 00:00:40.000000	1
2013-08-01 00:00:50.000000	1

(b)

Figure 4-7 sensor data table a) with multi-values b) with single value

4.3.2.2. The Temporal Database Generator

It is also unnecessary and inefficient to mine on the entire ambient data sample, since we are not interested in the pattern like the variation of the room temperature. Things we are concerned with are the sensor events triggered by occupants and ambient values (thresholds) and time points when events occur. So, the content of the temporal database will be reduced to the **sensor event** and their **associated ambient sensor values**. Sensor events refer to sensor records triggered by the occupants when they are performing their daily activities, such as turn on/off a switch and open/close a door. Associated ambient sensor values refer to the reading of ambient sensors, which are located in the same place with the triggered sensor, when a sensor event is detected. Time is also regarded as the associated ambient sensor value in this case. For example, the room temperature and humidity levels when the occupant opens a window are the associated ambient sensor values for the window-opening event.

4.3.2.2.1. The Conversion of Numerical Value

Both numerical and categorical sensor values have to be converted into categorical values for the temporal database. In this case, natural numbers are used as symbols. It is not difficult to assign categorical values with symbolic values. For example the on/off of one light can be assigned with “1” and “2” respectively, and “3”,”4” can be used for the state of another light. However, for the numerical value, such as the room temperature, ranging from 5 to 30 degrees, it is impossible to assign a symbol for each value recoded. In this algorithm, only the associated ambient sensor values need to be clustered.

People usually perform their daily activities at similar times and in similar circumstances. This assumption can be demonstrated by the data collected for a real life experiment. For example, in Figure 4-8, there is a 15 day record of the door senor of a monitored room collected by author. The value 1 and 0 represents the door is opened and closed, respectively. A very obvious pattern kept occurring in most of the days: the door open/close around 6 A.M. and 7 A.M. when the occupant got up and left home, around 12 P.M. and 1:30 P.M. when the occupant came back for lunch and left again, and around 6 P.M. when the occupant back home.

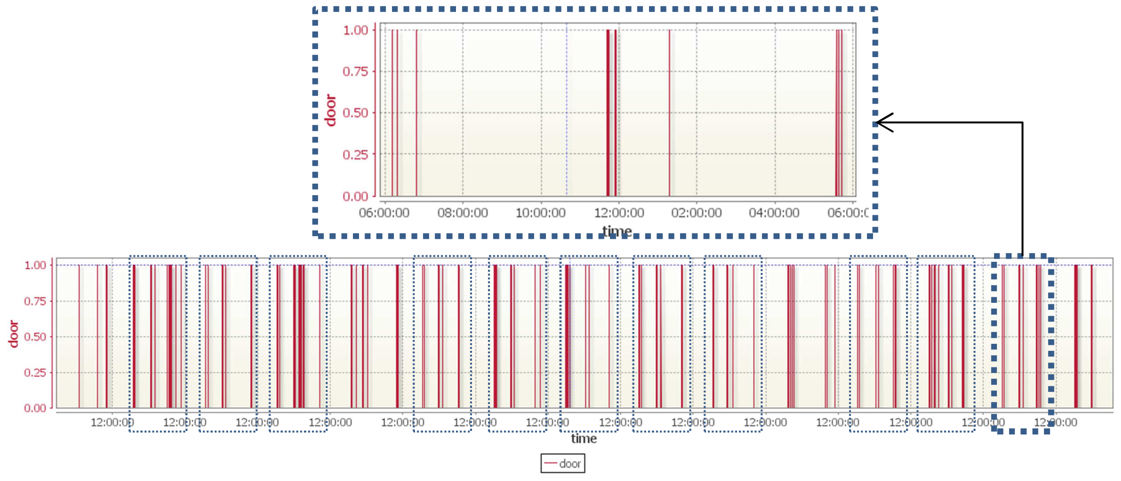


Figure 4-8 pattern exists in door sensor data

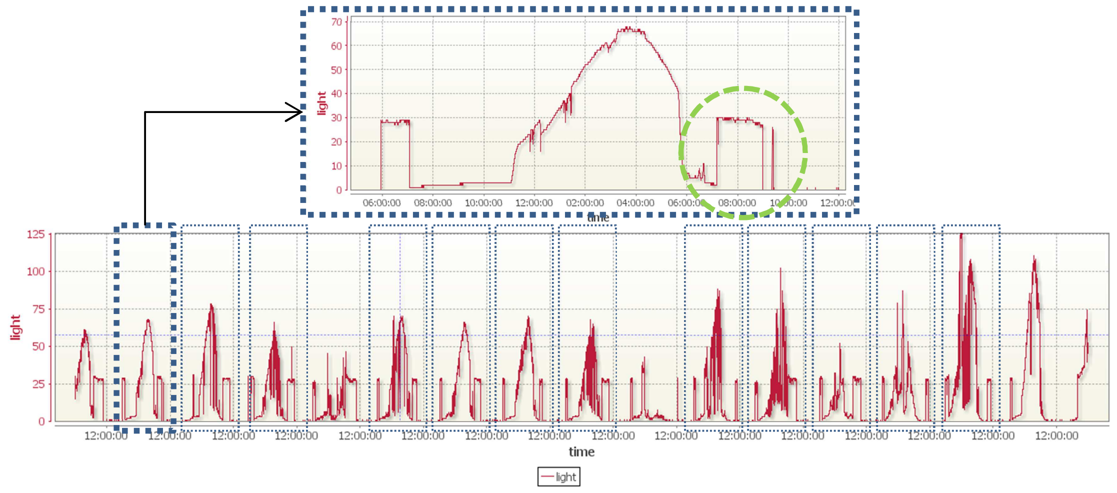


Figure 4-9 pattern exists in the light sensor data

In the same experimental environment, a pattern is also exists in the data recorded by the light sensor (Figure 4-9). It seems like when the light value was lower than 10 (raw data) and the occupant was in the room, then the occupant would turn on the light. Since the thresholds and time points of a certain event always appear within one or several regions, based on this prior knowledge, a cluster algorithm based on the density of the data is needed to cluster the ambient value and time point when event record take place.

After comparing several different cluster algorithms, DBSCAN is chosen for clustering the numerical sensor values and timestamps. It is a density-based clustering algorithm: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low-density regions (whose nearest neighbors are too far away) (Wikipedia 2015). As shown in Figure 4-10, the algorithm first picks a data point as an entry point then it starts to search its neighbor points within the distance R, if the amount of nearby neighbor points is larger than minPoints (minimal amount of point in a cluster), then this point will be defined as a **core point** (red points), and those nearby neighbor points will be defined as **border points** (pink points). The new search will be done by recursively expending the border points. The points not reachable from any other point are regard as noise (purple point).

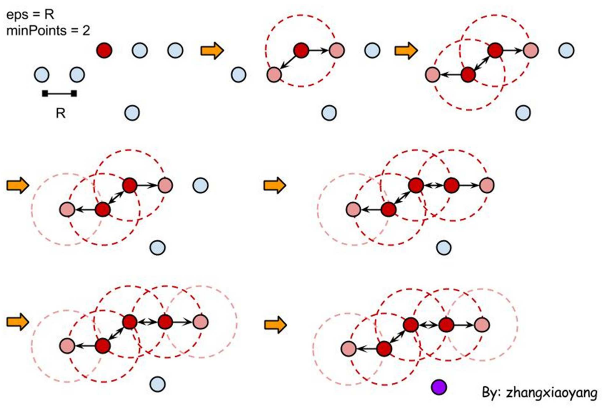


Figure 4-10 Processes of the clustering of DBSCAN

In a smart home case, we can first collect the ambient data when a certain event happens, cluster them with DBSCAN, and then assign each cluster with symbolic values. Take the door-opening event for example. During 15 days, 628 door-opening events were detected. There are very clear clusters that exist in the time line. In Figure 4-11, the upper chart shows the occurrences of door opening events and the time of day. Each diamond shape represents one occurrence. Chart 2 shows the distribution of the amount of events in each 15 minute time slot. Here DBSCAN is used to cluster one-dimensional data. With a 15 minutes distance, these events can be divided into 5 clusters (marked with red oval) and noise (marked with black rectangle). Cluster 1 can represent the time points around 6:30 A.M. when the occupant left home, cluster 2 is around 12:00 P.M. when the occupant back for lunch, and cluster 3 is around 7:00 P.M. when the occupant came back home. Cluster 4 is around 8:45 P.M. when the occupant finished the dinner in the kitchen and came back to the room. Cluster 5 is around 10:30 P.M. when the occupant went to the bathroom to wash up before sleep. There are also some accidental occurrences that can’t be clustered and defined as noise. In this case, there is strong pattern existing between the time and door opening. In other cases, for example, between the room temperature and the door opening, no such pattern exists and the distribution will be random. As a result, DBSCAN won’t able to cluster them and most of the occurrences will be regarded as noise. These noise data won’t be assigned with symbols, so the calculation complexity will be reduced.

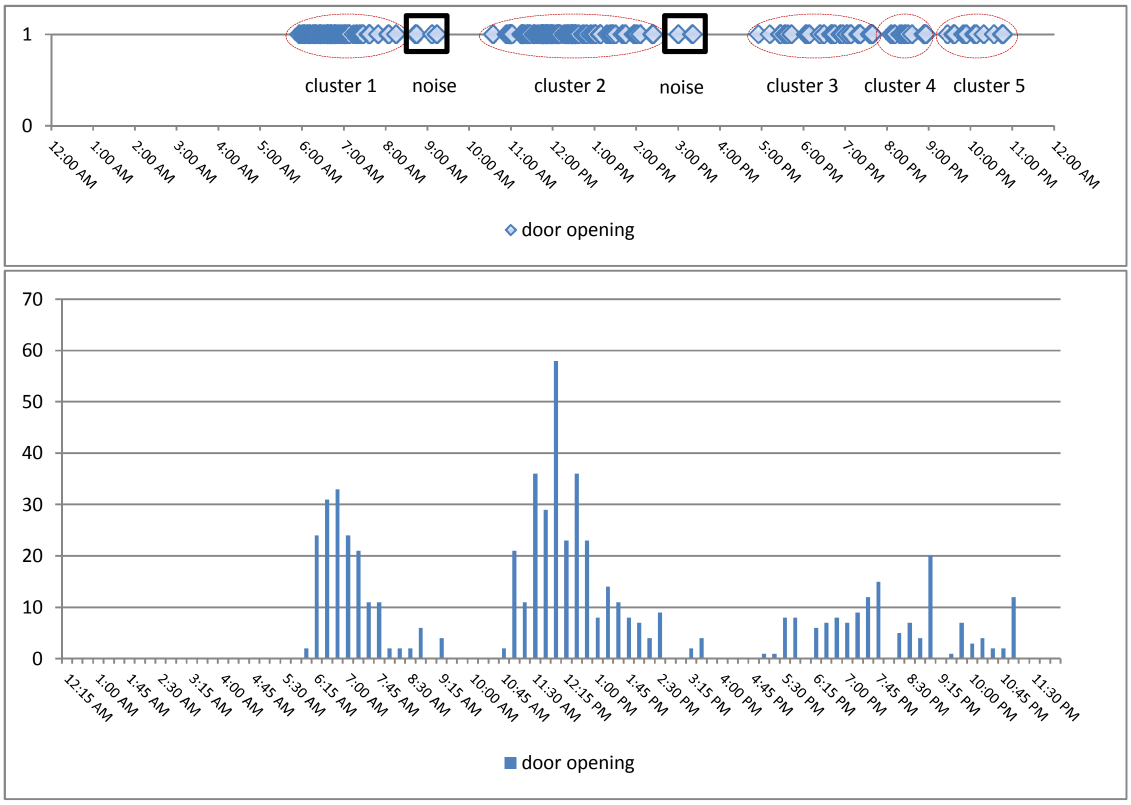


Figure 4-11 The 1)occurrences and 2)distribution of door opening time in one day based on 15 days observation

4.3.2.2.2. The Creation of the Temproal Database

The creation of the temporal database consists of 6 steps: (1) With the information in sensor metadata table, event sensors will be selected; (2) the ambient sensors associated with each event sensors will be selected respectively; (3) extract the sample values from the ambient sensors table when the target event sensors was triggered; (4) cluster these extracted values with DBSCAN; (5) assign each event sensor state and their associate ambient sensors clusters with symbolic values; (6) and sort all symbols in ascending order with the time stamp of the original sensor data. Symbols with same time stamps will be put into the same itemset (Figure 4-12).

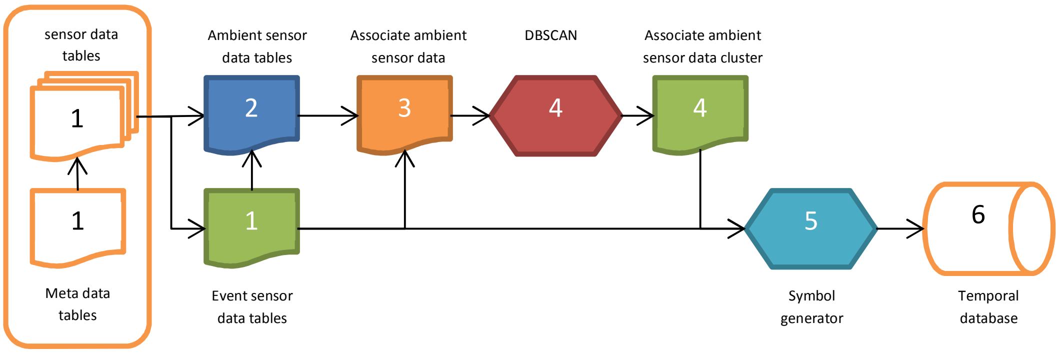


Figure 4-12 Process of creating the temporal database

For example, in the database (Table 4-10), there are three sensor data tables on 4 days of data: data table 2 is door sensor data table, data table 1 is window sensor data table, data table 3 is the room temperature table. In step 1, the metadata table shows that these three sensors are in the same room and data table 1 and 2 are event sensor table, while data table 3 is ambient sensor table. In step 2, table 3 is assigned to table 1 and table 2, respectively, as their associate ambient sensor. In step 3, each sensor event state in table 1 and 2 will extract their associate temperature values from table 3. For example, in table 2, the window-opening event occurs at 6:39 A.M., 7:24 A.M., 8:05 A.M., and 6:33 A.M. on each day. With table 3, we can get the temperature at these time points, which are 27.5℃, 26.8℃, 28.4℃, and 22.5℃. These values are the associated temperature values of the window-opening event. By clustering these data with 1 ℃ distance, the first three temperature values, (27.5℃, 26.8℃, 28.4℃), can be clustered into one group, while the fourth one, (22.5 ℃), will be defined as noise. Correspondingly, the window-closing event can also find its associate temperature values and there are clusters, (18.3℃, 17.5℃) and (22.9℃, 22.2℃). However, there is no cluster in the associated temperature values of the door opening and door closing. Since the time point of the event is also regarded as an ambient value, the cluster process will also be done on the associated time values. No cluster is found for the window-opening and closing event with 15 minutes distance. One cluster for door closing cluster: (4:22 P.M., 4:16 P.M., 4:19 P.M., 4:26 P.M.) is found. Now there are four different event states and 4 clusters, and each of them will be assigned with a symbol (Table 4-9). All the symbols are sorted by the occurring date and time, and then the temporal database is formed. In each element of the temporal database, there will be at least one symbol for the event and several symbols for its associate sensor data (Table 4-11). In this case, the temporal database will be like this:

$$\langle(1\ 5)(3)(4\ 8)(2\ 6)(1\ 5)(3)(4\ 8)(2\ 6)(1\ 5)(3)(4\ 8)(2\ 7)(1)(3)(4\ 8)(2\ 7)\rangle$$

With a $min\_\sup = 3$, one of the frequent episode, $\langle(1\ 5)(3)(4\ 8)(2)\rangle$, can be found with FEM. This episode can be translated into: in the morning the window will be opened when the

Events and clusters	symbol
Window open	1
Window close	2
Door open	3
Door close	4
Cluster (27.5℃, 26.8℃, 28.4℃)	5
Cluster (18.3℃,17.5℃)	6
Cluster (22.9℃, 22,2℃)	7
Cluster (4:22 p.m., 4:16 p.m., 4:19 p.m., 4:26 p.m.)	8

Table 4-9 Symbols assigned for each event and cluster

1 Window sensor	
Time stamp	state
08/18 6:39 a.m.	open
08/18 5:21 p.m.	close
08/19 7:24 a.m.	open
08/19 8:14 p.m.	close
08/20 8:05 a.m.	open
08/20 10:09 p.m.	close
08/21 6:33 a.m.	open
08/21 9:36 p.m.	close

2 Door sensor	
Time stamp	state
08/18 7:30 a.m.	open
08/18 4:22 p.m.	close
08/19 8:15 a.m.	open
08/19 4:16 p.m.	close
08/20 8:45 a.m.	open
08/20 4:19 p.m.	close
08/21 7:50 a.m.	open
08/21 4:26 p.m.	close

3 temperature sensor data	
Time stamp	value
08/18 6:39 a.m.	27.5
08/18 7:30 a.m.	28.8
08/18 4:22 p.m.	19.9
08/18 5:21 p.m.	18.3
08/19 7:24 a.m.	26.8
08/19 8:15 a.m.	23.7
08/19 4:16 p.m.	21.9
08/19 8:14 p.m.	17.5
08/20 8:05 a.m.	28.4
08/20 8:45 a.m.	25.5
08/20 4:19 p.m.	12.4
08/20 10:09 p.m.	22.9
08/21 6:33 a.m.	22.5
08/21 7:50 a.m.	20.4
08/21 4:26 p.m.	15.7
08/21 9:36 p.m.	22.2

Table 4-10 Tables in the database for window sensor, door sensor and temperature sensor data

temperature is around 26.5°C and then the door will be opened, around 4:20 P.M. the door will be closed and afterward also the window.

Date		08/18				08/19				08/20				08/21			
Time		6:39 a.m.	8:30 a.m.	4:22 p.m.	5:21 p.m.	7:24 a.m.	8:15 a.m.	4:16 p.m.	8:14 p.m.	8:05 a.m.	8:25 a.m.	4:19 p.m.	10:09 p.m.	6:33 a.m.	8:33 a.m.	4:26 p.m.	9:36 p.m.
Event		Win open	Door open	Door close	Win close	Win open	Door open	Door close	Win close	Win open	Door open	Door close	Win close	Win open	Door open	Door close	Win close
Temporal database	Symbol (event)	1	3	4	2	1	3	4	2	1	3	4	2	1	3	4	2
	Symbol (ambient)	5		8	6	5		8	6	5		8	7			8	7

Table 4-11 Creation of the temporal database

4.3.3. The Pattern Mining Module

The pattern mining module extracts patterns from the temporal database generated by the data preprocessing module. It provides two different algorithms for different kinds of patterns. The first algorithm derives from standard frequent closed episode mining algorithm, called **Frequent Sensor Event Episode Mining (FSEEM)**, which can extract all closed frequent episode from the temporal database. The second algorithm borrows some concepts form the FEM, called **User Specified Target Episode Mining (USTEM)**, the difference is it can extract the episodes that contain user specified item. With both algorithms we can find frequent and infrequent patterns.

4.3.3.1. The Frequent Sensor Event Episode Mining Algorithm

The reason why FEM is chosen as the mining algorithm instead of SPM is that the sensor event sequence is more like a temporal database rather than sequence database. Although the sensor event sequence can also be divided into segments by day or other interval to form the sequence database, the pattern longer or across the segments will be missed, since the FSEEM algorithm is based on FEM. Some commonly used definitions in FEM will be introduced first and then the algorithms will be introduced.

4.3.3.1.1. Preliminary

The mining target of FEM is the frequent episode, the concept of episode is very similar with the sequence in SPM, and actually, it is alias of sequence in FEM. Also, the concept of sub-episode and super-episode is similar with the sub-sequence and super-sequence. FEM also need a use specified threshold min_sup to set the threshold for frequency, but the calculation of the support is totally different from SPM. The maximal frequent episode is equal to the close frequent sequence.

Definition 19 (Episode) Given itemset $I = \{i_1, i_2, \dots, i_n\}(n \geq 1)$, a serial episode P over I is an ordered list of types of itemset, denoted as $P = \langle p_1 p_2 \dots p_m \rangle$, where $p_j \subseteq I(p_j \neq \emptyset, j =$

$1,2,\dots,m)$ is called an episode element. The length of P, denoted as $P.L$, is defined as m, and the size of P, denoted as $P.size$, is defined as the number of items that P contains. p_j always occurs before . p_{j+1} for all $j = 1,2,\dots m-1$.

Definition 20 (Sub-episode, Super-episode) An episode $\alpha = \langle a_1 a_2 \dots a_m\rangle$ is a sub-episode of another episode $\beta = \langle b_1 b_2 \dots b_n\rangle$, denoted as $\alpha \sqsubseteq \beta$, if there exist $1 \le i_1 < i_2 < \dots < i_m \le n$ such that $a_j \subseteq b_{i_j}$ for all $j = 1,2,\dots,m$, episode β is a super-episode of α.

Definition 21 (Frequent Episode, closed Frequent Episode) Given S and $\min\_sup$, an episode α is frequent with respect to $\min\_sup$ if $suppport(\alpha) \ge \min\_sup$. Episode α is a closed frequent episode with respect to $\min\_sup$ if for any $\beta \sqsupseteq \alpha$, $support(\beta) < \min\_sup$.

In sequence database, a pattern occurs in a sequence if the sequence contains this pattern. In temporal database, since an episode could occur in multiple locations in the single long event sequence, besides knowing the episode occurs, we also have to know where it occurs. The occurrence of the episode is defined like this.

Definition 22 (Occurrence, Minimal Occurrence) Given sequence $S = \langle (e_1)_1 (e_2)_2 \dots (e_n)_n\rangle$ and episode $P = \langle p_1 p_2 \dots p_m\rangle$, if there exist $1 \le j_1 < j_2 < \dots < j_m \le n$ s.t. $p_k \subseteq e_{j_k}$ for all $k = 1,2,\dots,m$, then α occurs in S, and $o = \langle (e_{j_1}, j_1)(e_{j_2}, j_2)\dots(e_{j_m}, j_m)\rangle$ is an occurrence of P in S. For simplicity, timestamps of the occurred elements are used $\langle j_1, j_2, \dots, j_m\rangle$ to denote o and $o[k] = j_k$ $(k = 1,2,\dots,m)$. Furthermore, if $st \le j_i < j_m \le et$, then we say window $win(S, st, et)$ contains P. The width of occurrence $o = \langle j_1, j_2, \dots, j_m\rangle$ is defined as $j_m - j_1$. An occurrence of P in S, $o = \langle j_1, j_2, \dots, j_m\rangle$, is minimal if P has no other occurrence in S, $o' = \langle j'_1, j'_2, \dots, j'_m\rangle$, s.t. $(j_1 < j'_1 \wedge j_m \le j'_m) \vee (j_1 \le j'_1 \wedge j_m < j'_m)$ (i.e., $[j'_1, j'_m] \subset [j_1, j_m]$). The set of all (minimal) occurrences of α in S is denoted as $O(S,\alpha)$ $(MO(S,\alpha))$.

Definition 23 (Window) Given $S = \langle (E_1, t_1)(E_2, t_2) \dots (E_n, t_n)\rangle (n \ge 1)$, a sliding window with width w over S from starting timestamp st, denoted as $win(S, st, w)$, is a sequence segment defined as

$$win(S, st, w) = \begin{cases} (E_{st})_{st}(E_{st+1})_{st+1} \dots (E_{st+w-1})_{st+w-1} & if st + w - 1 \le ct \\ (E_{st})_{st}(E_{st+1})_{st+1} \dots (E_{ct})_{ct} & otherwise \end{cases}$$

A window $win(S, st, w)$ contains an episode α if $\alpha \sqsubseteq win(S, st, w)$.

4.3.3.1.2. Frequency Measure

As discussed before, there are several different frequent metrics. Here, the Leftmost Maximal Non-redundant Set of Occurrences Frequency ($LMaxnR - O - freq$) proposed in (Gan and Dai 2012) is selected as the frequency metric. It is has been demonstrated to be more accurate than the other metrics.

Definition 24 ((Maximal) Non-redundant Sets of Occurrences) Given sequence S and episode P, a set of occurrences of P in S is non-redundant, if for any two occurrences, $o = \langle j_1, j_2, \dots, j_m\rangle$ and $o' = \langle j'_1, j'_2, \dots, j'_m\rangle$ $(o \ne o')$ in the set, $\neg\exists k \in \{1,2,\dots,m\}$, s.t. $j_k = j'_k$, denoted as $nR - O(S,P)$. There could be several different combinations of occurrences to form a set of $nR - O(S,P)$. The subsets with maximal cardinality are included in Maximal Non-redundant Sets of Occurrences, denoted as $MaxnR - O(S,P)$.

Based on the maximal non-redundant set of occurrences, a naive maximal strictly anti-monotonic frequency measure with $\max\_gap = k$ can be defined as

$$MaxnR - O - freq(S,\alpha,k) = \max_{\forall OS \in nR - O(S,\alpha,k)} (|OS|)$$

Here, $|OS|$ represents the size of occurrence set. However, $MaxnR - O - freq$ is time expensive to compute. To achieve a more efficient computation, for $MaxnR - O(S,P)$, a special set called the leftmost maximal non-redundant set, denoted as $LMaxnR - O(S,P)$ was chosen. To define the set, the occurrences in any non-redundant set are ordered by the ending timestamp in ascending order, i.e., in a sorted set $\{o_1, o_2, \dots, o_r\} \in nR - O(S,P), o_l[m] < o_{l+1}[m] (m = P.L)$ holds for all $l = 1,2,\dots,r-1$.

Definition 25 (LMaxnR-O) The leftmost maximal non-redundant set of occurrences of P in S, $LMaxnR - O(S;\ P)$, is defined as the occurrence set, $OS = \langle o_1, o_2, \dots, o_r \rangle$, that satisfies (1) $OS \in MaxnR - O(S,P)$ and (2) $OS' = \langle o_1', o_2', \dots o_r' \rangle \in MaxnR - O(S,P) (OS \neq OS')$, $o_l[k] \leq o_l'[k]$ holds for all $l = 1,2,\dots,r$ and $k = 1,2,\dots,P.L$.

Definition 26 (LMax-nR-O-freq) The measure based on the leftmost maximal non-redundant set of occurrences is defined as $LMaxnR - O - freq(S,P) = |LMaxnR - O(S,P)|$.Given S and $\min\_sup$, P is frequent in S if $sup(S,P) = LMaxnR - O - freq(S,P) > min\_sup$. For simplicity, S is omitted when it is apparent. For example, $sup(S,P) = sup(P)$, $O(S,P) = O(P)$ and $LMaxnR - O - freq(S,P) = LMaxnR - O - freq(P)$.

4.3.3.1.3. The Algorithm

The algorithm consists of two parts: the first module is the **episode growth module** and the other is the **episode enumeration tree**. The episode growth module extends the candidate episode and counts the frequency of the new extended episode with the $LMax - nR - O - freq$ metric, and sends the new found frequent episodes to the episode enumeration tree. The episode enumeration tree takes in these new episodes, compares them with the existing episodes in the tree, adds the new frequent episode in, prunes those redundant nodes to find the **non-derivable approximately closed frequently episodes (nDaCF) set**, and generates new candidates for the pattern growth module.

The main algorithm iteratively generates n-length candidate episodes, searches the new (n+1)-frequent episodes, and prunes the enumeration tree until not new frequent episodes can be found. The inputs of the algorithm are temporal database, minimal support, maximal gap, and maximal error bound. The maximal gap defines the maximal width of the windows size when creating the projected database. The maximal error bound defines the tolerance of the approximate episode comparison. The output of the algorithm is the episode enumeration tree, which contains the nDaCF set (Figure 4-13).

```
Algorithm: Frequent Sensor Event Episode Mining
Input: temporal database, min_sup, max_gap, max_err_bound
Output: nDaCF set

1: function FSEEM (temporal database , min_sup, max_gap, max_err_bound)
2:      epi_enum_tree ← ∅
3:      epi_enum_tree ← Get1LengthEpis (temporal database , min_sup)
4:      cand_epis ← GetCandEpis (epi_enum_tree)
5:      do
6:          freq_extended_epi ← EpisodeExtend (temporal database , cand_epis, min_sup,
        max_gap)
7:          Add&PruneTree (freq_extended_epi, max_err_bound)
8:          cand_epis ← GetCandEpis (epi_enum_tree)
9:      while freq_extended_epi ≠ ∅
11:     return epi_enum_tree
12: end function
```

Figure 4-13 Pseudo code for frequent sensor event episode mining algorithm

The episode growth module does a level wise search in the temporal database. Given n-length episode, it will find (n+1)-length episode. As shown in the pseudo code (Figure 4-14), algorithm first creates the projected database for each occurrences of the episode based on the max_gap(line 4). The frequent items will be selected for extending the episode (line 5). For each frequent item, it will be appended to the episode to form the $LMax - nR - O$ (line 7). If the frequency of the extended episode is larger than min_sup (line 8), then this episode will be returned.

```
Algorithm: extend the candidate episodes
Input: temporal database, cand_epis , min_sup, max_gap
Output: extended frequent episodes.

1: function EpisodeExtend (temporal database , cand_epis, min_sup, max_gap)
2:      freq_extended_epis ← ∅
3:      for $i = 1$ → | cand_epis | do
4:          p_database ← GenProjectDatabase (temporal database, cand_epis[i], max_gap)
5:          fre_items ← FindFrequentItems (p_database, min_sup)
6:          for $j = 1$ → | fre_items | do
7:              lmaxnro = FindLMaxnR_O (temporal database , cand_epi[i], fre_items[j])
8:              if | lmaxnro | ≥ min_sup then
9:                  freq_extended_epis ← Freq_extended_epis ∪ lmaxnro
10:             end if
11:         end for
12:     end for
13:     return freq_extended_epis
14: end function
```

Figure 4-14 Pseudo code for extension of the episode

There are two different ways to extend the episode: vertical extension and horizontal extension. When the new item occurs in the last element (i.e. itemset) of the episode, it is a vertical extension. When the new item is the element after the last element of the episode, it is a horizontal extension. To create the $LMax - nR - O$ set of the new episode, the vertical extension is very straightforward; just add the new frequent item into the last element. For the horizontal extension, for each occurrence o' of the candidate episode's occurrence set O', to find the leftmost occurrence o^* in the frequent item α's occurrence set O^*. As shown in (Figure 4-15), the inputs of the function are candidate episode (cand_epi), candidate frequent item (cand_fre_item), and maximal gap (max_gap). The output is LMaxnR_O of the new extended episode. The function first gets the all occurrences of candidate episode and the frequent item in the projected database (line 2). There is a nest loop to pair the episode and item. The outer loop iterate through each candidate episode (line 4). While The inner loop iterate through the frequent item (line 5). The algorithm compares the last timestamp of each occurrence of the candidate episode ($o'_i[lenght(cand\_epi)]$) with the timestamp of the each occurrence of the frequent item ($o^*_j[1]$) (line 7). If they have the same timestamp, then it is a vertical extension. If they have different timestamps, and the distance is within the max_gap, then it is a vertical extension. Finally, the $LMax - nR - O$ of the new extended episode is created and added to the return set (line 8).

```
Algorithm: calculate the LMaxnR_O
Input: cand_epi , cand_fre_item, max_gap
Output: LMaxnR_O of the extended episode

1: function FindLMaxnR_O (temporal database , cand_epi, cand_fre_item , min_sup, max_gap)
2: ← $\emptyset$; $O'$ ← O(cand_epi); $O^*$ ← O(cand_fre_item);
3: start ← 1
4: for $i = 1 \rightarrow |O'|$ do
5:         for $j$ = start → $|O^*|$ do
6:                 if $0 \leq o^*_j[1] - o'_i[lenght(cand\_epi)] <$ max_gap then
7:                         $o_i \leftarrow \langle o'_i[1] o'_i[2] \dots o'_i[lenght(cand\_epi)] o^*_j[1] \rangle$
8:                         $O \leftarrow O \cup o_i$
9:                         start ← j+1
10:                end if
11:        end for
12:end for
13:return O
14:end function
```

Figure 4-15 Pseudo code for calculating the LMaxnR_O

Items in the itemset don't have order. However, in the calculation they are arranged in an ordered list to reduce the calculation.

Definition 27 (Ordinal front, behind) I is a set of items $\{i_1, i_2, \dots, i_n\}$, i_j is ordinal in front of i_k if $1 \leq j < k \leq n$, denoted as $i_j \leftarrowtail i_k$, i_k is behind i_j, denoted as $i_k \rightarrowtail i_j$.

Episode $P = \langle p_1 p_2 \dots p_m \rangle$ can be extended with an item α in two different ways.

Definition 28 (Vertical Extension) Episode $P' = \langle p_1 p_2 \ldots p_{m-1}, p'_m \rangle$ is a vertical extension of P with item α, if $p'_m = p_m \cup \{\alpha\}$, and $\forall \beta \in p_m, \alpha \rightarrowtail \beta$, denoted as $P \diamond_V \alpha$.

Definition 29 (Horizontal Extension) Episode $P' = \langle p_1 p_2 \ldots p_m \alpha \rangle$ is called a horizontal extension of P with item α, denoted as $P \diamond_H \alpha$.

Episode enumeration tree is a hierarchical tree structure which stores the information of frequent episodes. It starts with a root node $\emptyset$ and the 0 level. Each node in the tree represents an extension of its parent node. Node contains the information of the item and the frequency of the episode. There are two kinds of links between the nodes. One represents vertical extension the other is horizontal extension. By concatenating each node on the path from leaf to root, the frequent episode can be restored. To facilitate the tracking of the patterns, in this algorithm, index of occurrences in the temporal database are also recorded in the node. A series of strategies are introduced to check and prune derivable episodes. There are two kinds of derivable episodes: non-closed frequent episode and approximately derivable episodes.

The concept of nDaCF set was first introduced in 2012 (Gan and Dai 2012). The difference between nDaCF set and the closed episode set is that nDaCF also introduces the approximately closed episode. With this feature, the small different in frequencies between the episode and its super episode can be neglected. So, it is more compact than the set of closed frequent episode. It is defined like this:

Definition 30 ((Non)Exactly Closed Episode) episode P is exactly closed, denoted as eC, if $\neg\exists P' \sqsupset P$, such that $support(P) = support(P')$, else P is non-exactly closed, denoted as neC.

Definition 31 ((Non)Approximately closed Episode) Given P and maximal error bound θ, P is approximately closed, denoted as aC , if $\neg\exists P' \sqsupset P$, such that $|support(P') - support(P)|/\sup(P) \leq \theta$. Otherwise, P is non-approximately closed, denoted as naC.

Definition 32 (Derivation Relationship) Given episode $P' \sqsupset P$, P can be exactly derived from P', if $support(P) = support(P')$, denoted as $eD(P, P')$. Given maximal error bound θ, P can be approximately derived from P', If $|support(P') - support(P)|/\sup(P) \leq \theta$, denoted as $aD(P, P')$.

Definition 33 (The nDaCF Set) Given temporal database, min_sup, and maximal error bound θ. A set of frequent episodes $\mathbb{P}$ is called nDaCF set, if (1) $\forall P \in \mathbb{P}$, P can't be exactly or approximately derived from any other episode in $\mathbb{P}$, and (2) $\forall P \notin \mathbb{P}$, P can be exactly or approximately derived from one of the episode in $\mathbb{P}$.

The pseudo code is provided to show the maintenance of the enumeration tree (Figure 4-16). The inputs of the function are the new frequent episode found by episode growth module (freq_extended_epi) and the maximal error bound (max_err_bound). The output is the episode enumeration tree. When a new episode is added to enumeration tree, the algorithm first checks the amount of layers in the tree. If it is less than the length of the new episode (line 2), the tree will increase a new layer in the tree (line 3), and create a new node for this episode and put it into the corresponding branch (line 5). Then the new episode is compared with each of the episodes recorded in the tree (line 6). If an existing frequent episode is the sub episode of this episode (line 8), their frequency will be compared. If they have the same support, then the existing sub episode is a non-exactly closed episode, the node represents this episode and its branch will be removed from the tree. If the difference of the frequencies is smaller than the maximal error bound, then this episode is non-approximate closed, the corresponding node and branch will also be removed from the tree (line 9, 10).

```
Algorithm: Add new frequent episode and prune derivable episode
Input: freq_extended_epi, max_err_bound
Output: epi_enum_tree

1: function Add&PruneTree (freq_extended_epi, max_err_bound)
2:      if length(freq_extended_epi) > tree layer amount then
3:          AddTreeLayer()
4:      end if
5:      AddNodeToTopLayer()
6:      for each node in enumeration tree do
7:          t_episode = GetEpisode (node)
8:          if IsSubEpisode (t_episode, freq_extended_epi) = true then
9:              if |O(t_episode) – O(freq_extended_epi)|< max_err_bound then
10:                 PruneNode (node)
11:             end if
12:         end if
13:     end for
14:end function
```

Figure 4-16 Algorithm for adding and pruning the episode in the enumeration tree

A simple example will be given here to demonstrate the mining process (Figure 4-17). Given an itemset $I = (1,2,3,4,5,6,7,8)$, minimal support = 3, maximal gap = 2, max error bound = 1 and temporal database S:

$$\langle (2\ 4)(1\ 3\ 4)(6)(5)(3\ 7)(6)(2\ 3)(1\ 3\ 6)(5\ 8)(7)(6)(2)(1\ 2\ 3\ )(4\ 5)\rangle$$

Figure 4-17 Temporal database

The algorithm firstly searches for the 1-length episode in the database. Each appearance of the frequent episode is $\langle(1)\rangle{:}\,3$, $\langle(2)\rangle{:}\,4$, $\langle(3)\rangle{:}\,5$, $\langle(4)\rangle{:}\,3$, $\langle(5)\rangle{:}\,3$, $\langle(6)\rangle{:}\,4$. The frequencies of episodes $\langle(7)\rangle$ and $\langle(8)\rangle$ are less than minimal support. These episodes will be added into the enumeration tree (Figure 4-19, step 1).

$$\langle (2\,\boxed{4)(1\ 3\ 4)(6)}(5)(3\ 7)(6)(2\,\boxed{3)(1\ 3\ 6)(5\ 8)}(7)(6)(2)\boxed{(1\ 2\ 3\ )(4\ 5)}\rangle$$

Figure 4-18 Projected database for episode <(2)> with max_gap = 2

These frequent episodes will be extended in the order of their occurrence in the database, i.e. ⟨(2)⟩ first. This can help to reduce the amount of derivable episodes during the mining process. The projected databases $S|_{\langle(2)\rangle}$ will be generated for each occurrence of the episode based on the maximal gap parameter (Figure 4-18). There are three of them: $\langle(\_\ 4)(1\ 3\ 4)(6)\rangle, \langle(\_\ 3)(1\ 3\ 6)(5\ 8)\rangle, \langle(1\ 2\ 3)(4\ 5)\rangle$. Two new frequent items are found in the projected database: $\langle(1)\rangle: 3, \langle(3)\rangle: 3$. Both of them are horizontal extensions. The episode ⟨(2)⟩ is extended to two new frequent episodes: $\langle(2)(1)\rangle: 3, \langle(2)(3)\rangle: 3$. After adding these two episodes into the enumeration tree, the node 1 in layer 1 will be pruned, since ⟨(1)⟩ is a sub-episode of ⟨(2)(1)⟩ and they have the same frequency. Episode ⟨(1)⟩ is a non-exactly closed episode. The node 2 which represents the ⟨(2)⟩ is also pruned, because the ⟨(2)⟩ is sub-episode of ⟨(2)(1)⟩, and their frequency difference is within the maximal error bound 1. Although ⟨(3)⟩ is the sub set of ⟨(2)(3)⟩, but their frequency difference is 2, node 3 can't be pruned. Similarly, the episode ⟨(3)⟩ can be extended to $\langle(3)(5)\rangle: 3$ and $\langle(3)(6)\rangle$:3. By adding the new episode into the tree, node 5 and 6 can be pruned. In this round no more frequent items can be found (Figure 4-19 step2). The search starts to generate new sequence candidates with the node in layer 2, they are $\langle(2)(1)\rangle: 3$, $\langle(2)(3)\rangle: 3$, $\langle(3)(5)\rangle: 3$, $\langle(3)(6)\rangle$: 3. This time, the episode ⟨(2)(1)⟩ is first extended. One is vertical extension ⟨(2)(1 3)⟩:3, both node 1 and 3 in layer 2, is pruned because they are non-exactly closed. The other one is horizontal extension ⟨(2)(1)(5)⟩:3. Nothing can be pruned with this episode (Figure 4-19 step 3). The tree generates the third round of candidates, they are ⟨(2)(1 3)⟩:3 and ⟨(2)(1)(5)⟩:3. The episode can be extended to ⟨(2)(1 3)(5)⟩:3. By adding it into the tree, node 3 and 5 in layer 3 and node 5 in layer 2 can be pruned. Then the tree generates the fourth round candidates ⟨(2)(1 3)(5)⟩:3, however it can't be extended in its projected database. The whole search comes to the end. Nodes still in blue in the enumeration tree are the episodes haven't been pruned. They are ⟨(2)(1 3)(5)⟩:3, ⟨(3)(6)⟩:3 and ⟨(4)⟩:3 (Figure 4-19). During the whole process, we can see the pruning of derivable episode reduces the searching space efficiently.

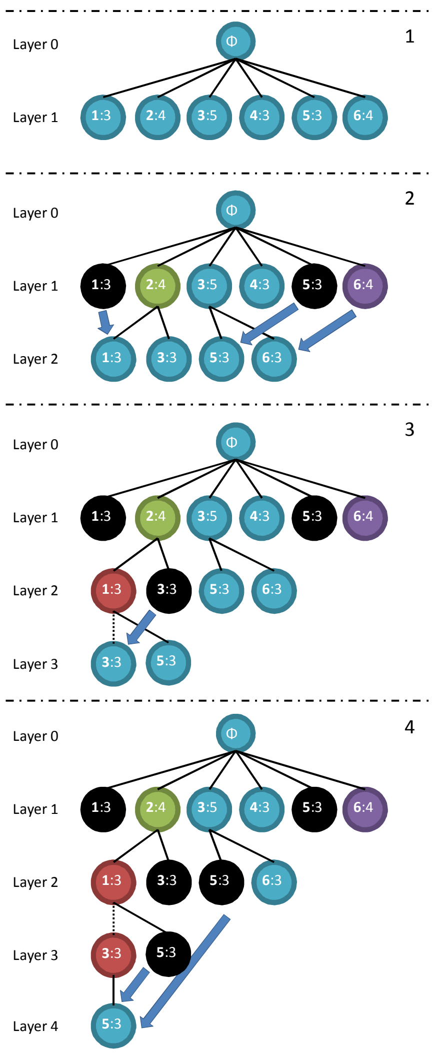


Figure 4-19 Pruning process in the enumeration tree

4.3.3.2. The User Specified Target Episode Mining Algorithm

The user specified target episode mining allows the use to set a target item in the temporal database. The algorithm will find out episodes that end with this item, and each item in the projected database has similar frequency with the target item.

The inputs of the algorithm are temporal database, target item, maximal gap, and maximal error bound. The difference between the FSEEM and USTEM are (1) instead of the minimal support, the USTEM requires a target item (FSEEM looks for the episodes occur more frequently than the minimal support, while USTEM looks for the episodes have the similar frequency with the target item and end with the target item) and (2) the FSEEM does a forward search in the temporal database, while the USTEM dose a backward search start with each occurrence of the target item.

```
Algorithm: User Specified Target Episode Mining
Input: temporal database, target item, max_gap ,max_err_bound
Output: specified target episode

1: function USTEM (temporal database, target item, max_err_bound)
2: epi_enum_tree ← $\emptyset$
3: epi_enum_tree ← target item
4: target_freq ←| O(target item)|
5: cand_epis ← GetCandEpis (epi_enum_tree)
6: do
7:      back_extended_epis ← EpisodeBackwardExtend (temporal database, cand_epis, target_freq,
        max_gap, max_err_bound)
8:      Add&PruneTree (back_extended_epis)
9:      cand_epis ← GetCandEpis (epi_enum_tree)
10: while freq_extended_epi $\neq \emptyset$
11: return epi_enum_tree
12: end function
```

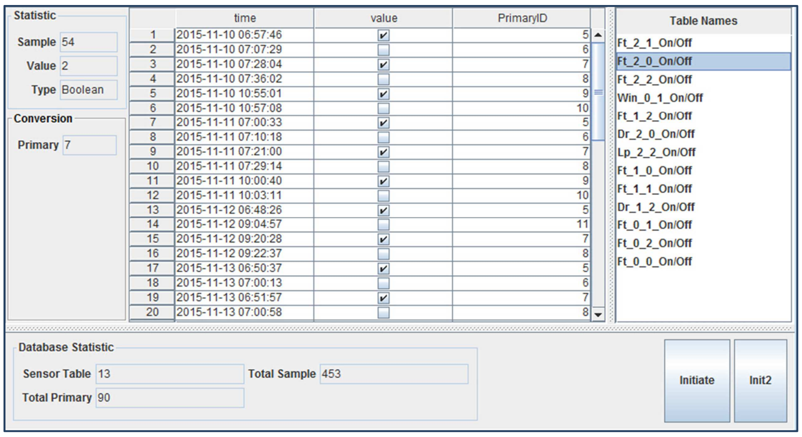
Figure 4-20 Pseudo code for user specified target episode mining

In the algorithm (Figure 4-20), the target item is used as the 1-length episode and added to the episode enumeration tree (line 3). The target frequency is set as same as the frequency of the target item (line 4). Iteratively, new backward extended episode will be generated by backward extension (line 7). The newly found backward extended episode will be added into the enumeration tree (line 8), and derivable episodes will be pruned and new candidates will be generated (line 9). In the backward extension process, not only the low frequent items will be neglected, but the frequency out of the range, $[targe\ frequent - max\_err\_bound,$ $target\ frequent + max\_err\_bound]$, will also be neglected. Beside this, most of the functions used in FSEEM are compatible with USTEM by reversing the temporal database. No detailed description will be given here.

4.3.4. The Visualization Module

The visualization module visualizes the data created during the whole mining process. Unlike the most of the algorithms, it just provides the frequent episodes and their supports. With this visualization module, researchers are able to track the occurrence of the patterns and understand the actual meaning of the patterns. There are four different charts: **sensor data table chart, symbol scatter chart**, **enumeration tree chart**, and **frequent episode chart**.

The sensor data table chart displays the content of the table, the metadata from the sensor metadata table and some statistics of the table, such as the data type of the sensor value, the amount of different values and the size of the table, etc. User is allowed to select the table from the table name list panel. After assigning the symbolic value for each data, the corresponding symbol will appear in the primary ID column (Figure 4-21).



Statistic

Sample 54

Value 2

Type Boolean

Conversion

Primary 7

	time	value	PrimaryID
1	2015-11-10 06:57:46	✓	5
2	2015-11-10 07:07:29	☐	6
3	2015-11-10 07:28:04	✓	7
4	2015-11-10 07:36:02	☐	8
5	2015-11-10 10:55:01	✓	9
6	2015-11-10 10:57:08	☐	10
7	2015-11-11 07:00:33	✓	5
8	2015-11-11 07:10:18	☐	6
9	2015-11-11 07:21:00	✓	7
10	2015-11-11 07:29:14	☐	8
11	2015-11-11 10:00:40	✓	9
12	2015-11-11 10:03:11	☐	10
13	2015-11-12 06:48:26	✓	5
14	2015-11-12 09:04:57	☐	11
15	2015-11-12 09:20:28	✓	7
16	2015-11-12 09:22:37	☐	8
17	2015-11-13 06:50:37	✓	5
18	2015-11-13 07:00:13	☐	6
19	2015-11-13 06:51:57	✓	7
20	2015-11-13 07:00:58	☐	8

Table Names

Ft_2_1_On/Off
Ft_2_0_On/Off
Ft_2_2_On/Off
Win_0_1_On/Off
Ft_1_2_On/Off
Dr_2_0_On/Off
Lp_2_2_On/Off
Ft_1_0_On/Off
Ft_1_1_On/Off
Dr_1_2_On/Off
Ft_0_1_On/Off
Ft_0_2_On/Off
Ft_0_0_On/Off

Database Statistic

Sensor Table 13 | Total Sample 453

Total Primary 90

Figure 4-21 Sensor data table chart

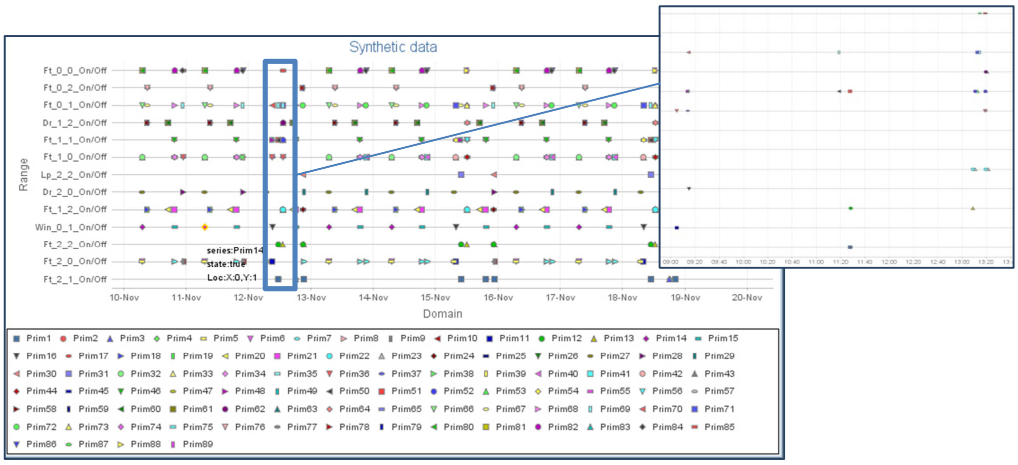


Figure 4-22 Symbol scatter chart

These symbolic values can also be displayed in the symbol scatter chart for getting an intuitive feeling of the distribution of those data on the time line. The X-axis is the time line, and each tick on Y-axis represents an event sensor or one of its associated ambient values. Each colored geometry in the plot represents a symbolic data assigned to a state of the event sensor or a cluster of the ambient sensor data. The original values and metadata are accessible by click on the geometries. The chart is zoomable, so different level of details can be shown (Figure 4-22).

During the mining process, the enumeration tree chart and frequent episode chart work together to display the information of the new frequent sequence. In enumeration tree chart, each circle represents a node in the tree structure. The different color represents the different state of the node: black is pruned and, yellow is not pruned. The number in the circle before the ":" is the primary ID of the node, after ":" is the frequent of the episode. The link between the nodes represents the type of extension: solid line is horizontal extension and dash line is the vertical extension. The frequent episode chart is a 2D-grid, each row for each symbol, and each column represent a timestamp in the temporal database. The cell containing symbolic value will be filled with rectangle. When a certain node is selected in the enumeration tree chart, the corresponding occurrences of this episode will be market in the frequent episode chart. Each event in one occurrence will be connected by red line (Figure 4-23).

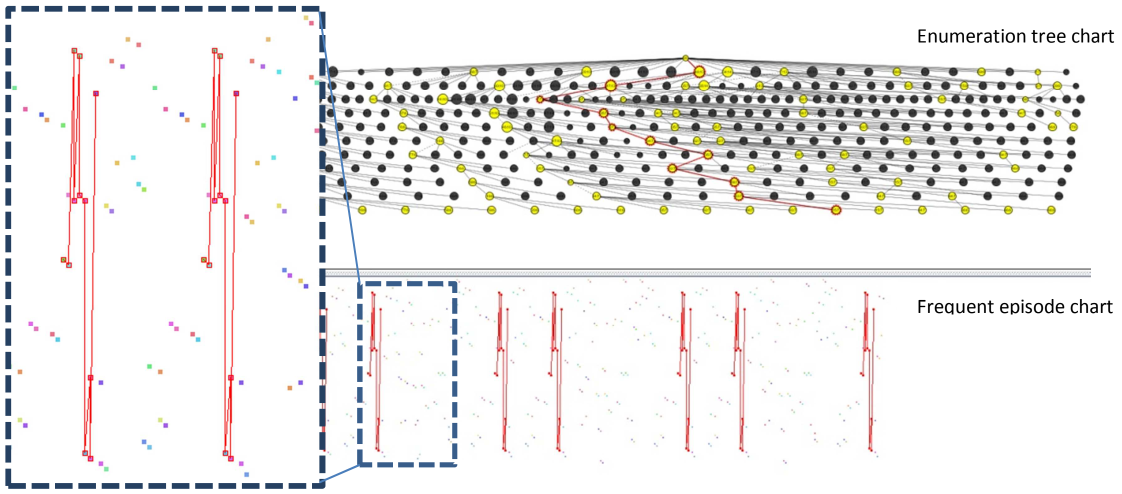


Figure 4-23 Enumeration tree chart and frequent episode chart

Chapter 5. The Framework and Experiments

In this chapter, the framework of the whole pattern discovery process will be introduced. Two tests have been conducted: the first one is based on the synthetic data generated by a simulator, which can generate sensor data records with patterns and noises. This test can examine the performance of the mining algorithm in searching the patterns with a different portion of noises. The second test is based on the real life data collected from a student dormitory, which provides a demonstration and evaluation of the proposed framework in real world application.

5.1. The Framework of the Pattern Discovery Process

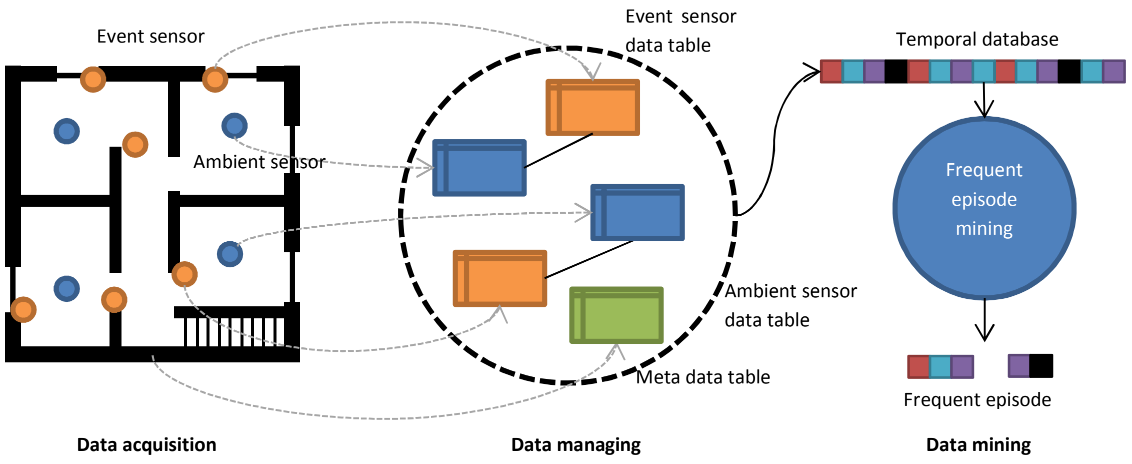


Figure 5-1 Diagram for pattern discovery process

The framework consists of three parts: **data acquisition**, **data managing**, and **data mining** (Figure 5-1). The data acquisition task is done by the WSN, which contains both event sensor and ambient sensor node. The ambient sensor periodically records the environment parameters such as light, temperature and humidity, which can potentially affect the occupants' behaviors. The event sensor records the events triggered by occupants, such as turning on/off the light or open/ close the door. When data acquisition processing is done, it goes to the data managing process. Data are stored and maintained in the form of tables in database. With the database, the data mining process can be **decoupled** with the collection process. This means, the data mining algorithm can use any data in the database regardless the data source, as long as the data follows the format defined in database. On the other side, the data in the database can be used for other mining algorithms or researches. There are event sensor data and ambient sensor data for recording the data from event and ambient sensor respectively. The metadata table provides the spatial relationship for linking the event sensor tables with ambient sensor tables. With the data and their relationships in the database, the temporal database can be created for mining. Until this stage, the WSN data have been converted into a normal FEM problem. Any FEM algorithms can be applied. The frequent episode can be linked back to the data in the database to get meaningful results. With this setup, the amount of data is kept low without losing the event detection ability. This will help to extend the lifetime of the WSN, and bring down the calculation complexity for the mining algorithm.

There are several facts that can affect the output patterns. The first is the features of input sensor data, including the **density** of the sequential pattern and, the average **length** of the patterns. The second is the parameters for clustering the ambient sensor data, including the **maximum distance** between each data points and **minimum number of data points** in one cluster. The third is the parameters for the FEM mining algorithm, including the **minimum support** of the pattern (min_sup), the size of the **window constrain** (max_gap), and the **degree of approximation** for pruning the tree node (max_err_bound). Several experiments have been conducted to examine their effects on the mining results and find the right combination of them.

5.2. Experiments with Synthetic Data

The first experiment was done with the synthetic data. Since, in the real life data, it is hard to know how many patterns exist in the occupants ADLs, to tell if the mining algorithm has detected the pattern or the percentage of the patterns that are detected is very hard. Here, a smart home simulator is designed for generating the synthetic sensor data. With the simulator, researcher can specify the composition, occurrence, and frequency of a pattern. Several patterns can be combined together with noises. With fully controlled data, the performance of the algorithm can be measured precisely.

5.2.1. Description of the Data Source

The simulator is written in JAVA and in object oriented programing fashion. All the components are programmed as JAVA classes. There are mainly five kinds of components: occupant, appliance, sensor, task, and event. The inputs of the simulator are scripts for describing the layouts of the virtual environment and the daily schedule for defining the ADL pattern of the occupant. The occupant operates each task in the schedule, interacts with appliances, and triggers sensors. Sensor records the data into tables. The outputs are these sensor data tables, which can be directly uses by the data mining program.

The architecture of the simulator is organized as shown in Figure 5-2. The Appliance Class is an abstract Class to represent all actual appliance or building elements. Here FloorTile, Window, Door, and Lamp class are derived from Appliance class. The Sensor class is also an abstract class. Each Appliance as a Sensor attached with. The Room class holds instances of Appliance in a two dimensional array. The Task class is an abstract class for all ADLs. The Walk and OpenDoor class are real implementation of the Task class. The Routine class contains an array of instants of Task. The Occupant class contains the information of the occupant, such as location, room, and routine.

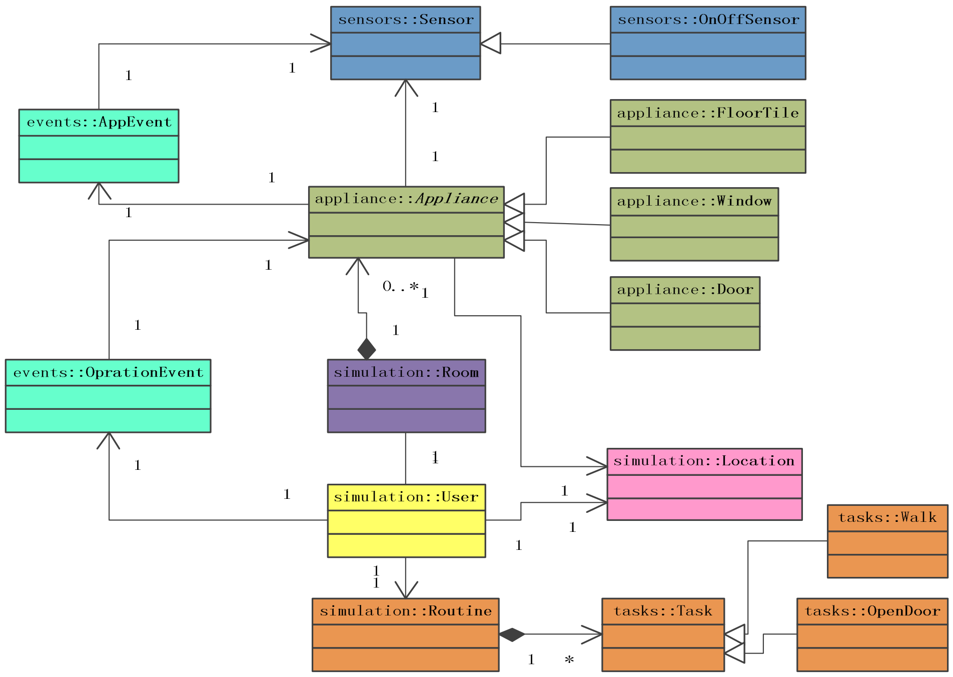


Figure 5-2 Class diagram for the smart home simulator

The whole simulation process is driven by events. The occupant will look into the routine list, and operate the task one by one. The task tells the occupant the time, the target appliance and the operation, such as at 7:00 A.M., lamp1, and turn on. A list of subsequences will be generated to fulfill the task including get to target location and operate on the appliance. Each occupant and appliance has an instance of Location class which specifies its position in the room. It records the x and y coordinates. By comparing the position difference between the occupant and target appliance, a path will be calculated first. Then, a series of Walk task will be generated to guide the occupant goes along the path. The operation of the Walk task might also trigger sensors along the path. When a task is operated, an OperationEvent will be created. The event contains the information of the operation. For example, if it is a turn on event or turn off event. When the appliance receives the event, it will create another appliance event, i.e. an instance of AppEvent. This event will be caught by the attached sensor. The sensor then creates a record in the sensor data table.

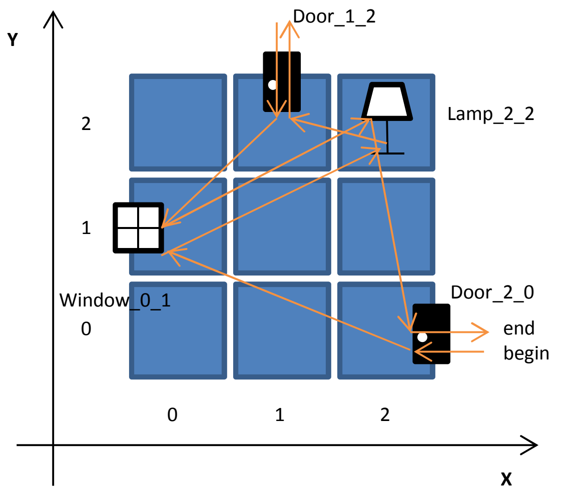


Figure 5-3 Setup of the virtual environment

In this experiment, a small virtual room environment with one occupant is set up for generating the synthetic data (Figure 5-3). The room space is a 3×3 grid. The floor in each cell has a pressure sensor attached. It will be triggered when the occupant step on it. There is one window located in cell (0, 1), and two doors located in cell (1, 2) and cell (2.0) respectively. Each of them is attached with a sensor to detect the open/close state. A lamp is located in cell (2, 2) also attached with an on/off sensor.

A daily routine is given to define the ADL pattern. As shown in the script, each line represents a task. Contents are separated with commas. Take the first task for example, “user1, TurnOnOff: Door_2_0: open, a: 7: h, 10: m” The first part is the occupant ID, i.e. “user1”. The second part is the description of the task, including the task name “TurnOnOff”, the target appliance “Door_2_0”, and the state of the appliance “open”. The third part is the operation time. It can be an absolute time (marked with “a”) in a day or a relative time to the pervious task (marked with “b”). The last part is the range of the randomness in time. In this case, “a: 7: h, 10: m” means 7:00 A.M. $\pm$ 10 minutes.

The whole script can be interpreted like this: in the morning, the occupant first enters the room through Door_2_0. Then opens the window, turn on the lamp, and left the room through Door_1_2. In the afternoon, the occupant returns the room through the Door_1_2, closes the window, turns off the lamp and leaves the room through Door_2_0 again.

In real life data, the data are full of noises and randomness. In the synthetic data, these noises and randomness are generated in four different ways.

In the first way, as shown in the definition of the daily routine, each task is assigned with a random time range. Based on the observation on the real life data, the operation time of a certain task is distributed in a Gaussian distribution fashion.

Secondly, the auto generated path for each task has certain randomness. For example, from Door_2_0 to Window_0_1, the path could be (2,0),(1,0),(0,1) or (2,0),(1,1),(0,1). Every time the occupant will randomly choose one of the paths.

The third randomness is generated by the random task command in the script. For example, when meets the line "user1, Random, r: 2: h, 1: h", the program will create a random task for the occupant in around 2 hours after previous task.

The fourth randomness is generated by mixing different daily routines. Each script file represents a daily routine. The program can load several different routines at one time and combine them together. This can simulate the difference between working day and weekend.

Several experiments have been conducted to test the performances and outputs with different input parameters. With a given virtual environment, there are several variables can be changed: (1) the input sensor data, (2) the minimum support, (3) the size of the look back window and (4) the parameters for the cluster algorithm.

5.2.2. Experiment 1

```
user1,TurnOnOff:Door_2_0:open,a:7:h,10:m
user1,TurnOnOff:Window_0_1:open,r:10:m,20:s
user1,TurnOnOff:Door_1_2:open,r:5:m,20:s
user1,TurnOnOff:Door_1_2:close,r:1:m,20:s
uset1,Walk:-1_-1,r:1:m,20:s
user1,TurnOnOff:Door_1_2:open,a:17:h,10:m
user1,TurnOnOff:Door_1_2:close,r:1:m,20:s
user1,TurnOnOff:Window_0_1:close,r:10:m,60:s
user1,TurnOnOff:Door_2_0:close,r:20:m,1:m
uset1,Walk:-1_-1,r:1:m,20:s
```

Figure 5-4 Script for describing the routine, experiment 1

In this experiment, the daily routine shown in Figure 5-4 is set as the input routine. In this routine, one occupant simply enters the room through one door, opens the window and leaves the room through another door in the morning. In the afternoon, the occupant enters the room, closes the window and leaves the door again. The randomness in the routine exists in the walking path and operation time of each task. The duration of the virtual environment lasted for 10 days. 320 data samples were collected. Since there was no ambient sensor installed, only the operation time needed to be clustered as an ambient value. 15 minutes is chosen as the cluster distance. With these settings, 48 items were generated to represent all events and ambient data groups (Table 5-1). In the mapping table, this first column is the ID for the item, the second and third columns are for the sensor events and its associated ambient data clusters. The minimum support was set to 10, and max gap was set to 3. The approximate threshold for pruning the node was set to 1.

Item ID	Event	Ambient	Item ID	Event	Ambient
1	Floor Tile_0_1 off		**25**		Time{16:52:04, 17:05:49}
2		Time {17:19:05, 17:33:11}	**26**		Time{7:08:07, 7:18:15}
3		Time {7:07:47, 7:17:54}	**27**	Door_1_2 off	
4	Floor Tile_0_1 on		**28**		Time{16:54:13, 17:08:27}
5		Time{17:03:40, 17:16:55}	**29**		Time{7:18:18, 7:18:41}
6		Time{7:02:44, 7:12:45}	**30**	Door_1_2 on	
7	Window_0_1 off		**31**		Time{16:53:11, 17:07:25}
8		Time{17:04:33, 17:17:36}	**32**		Time{7:08:27, 7:18:36}
9	Window_0_1 on		**33**	FloorTile_2_0 off	
10		Time{7:03:36, 7:13:36}	**34**		Time{17:24:29, 17:38:09}
11	Floor Tile_0_2 off		**35**		Time{7:09:57, 7:10:14}
12		Time{7:08:07, 7:18:15}	**36**	FloorTile_2_0 on	
13	Floor Tile_0_2 on		**37**		Time{17:22:40, 17:36:38}
14		Time{7:07:47, 7:17:54}	**38**		Time{6:52:32, 7:02:00}
15	FloorTile_1_1 off		**39**	Door_2_0 off	
16		Time{17:03:40, 17:34:42}	**40**		Time{17:24:29. 17:38:09}
17		Time{7:02:44, 7:12:45}	**41**	Door_2_0 on	
18	FloorTile_1_1 on		**42**		Time{6:53:40, 7:03:45}
19		Time{17:02:49, 17:33:11}	**43**	FloorTile_2_1 off	
20		Time{7:01:54, 7:11:54}	**44**		Time{17:22:40, 17:36:38}
21	FloorTile_1_2 off		**45**		Time{7:01:54, 7:11:54}
22		Time{17:02:49, 17:16:13}	**46**	FloorTile_2_1 on	
23		Time{7:10:47, 7:20:49}	**47**		Time{17:20:54, 17:34:42}
24	FloorTile_1_2 on		**48**		Time{7:01:04, 7:11:00}

Table 5-1 Item ID to event and associated amtient cluster mapping table, experiment 1

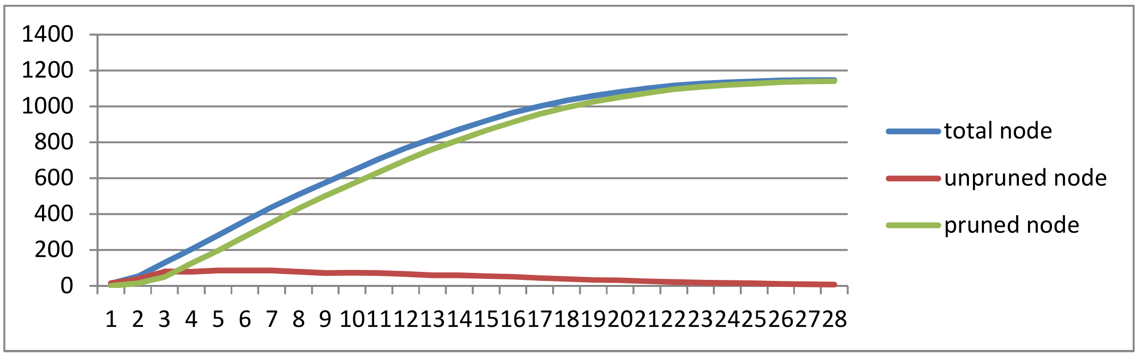


Figure 5-5 Node amount in the enumeration tree in each interation, expeirment 1

Figure 5-5 shows the amount of total, unpruned and pruned nodes in the episode enumeration tree in each iteration. In the first three iterations, the unpruned node, i.e. closed frequent episodes, took a large percent of the found nodes. After that, the amount of unpruned nodes kept decreasing. After 28 iterations, 1148 tree nodes are found, 1141 of them are pruned, and 7 closed frequent episodes were found. They were:

1, <18 15 >:29

2, <36 33 >:19
3, <24 30 27 21 >:19
4, < (18 19) (15 16) >:20
5, <46 43 >:19
6, <18 (4 15) 1 >:19
7, <(24 25) (30 31) (27 28) (18 19 21 22) (4 5 15 16) (1 2 18 19) (15 16 46 47) (36 37 43 44) (33 34) >:10

The entire episode enumeration tree is shown in Figure 5-6. Each circle represents a tree node. Nodes in gray are pruned and in yellow are unpruned nodes.

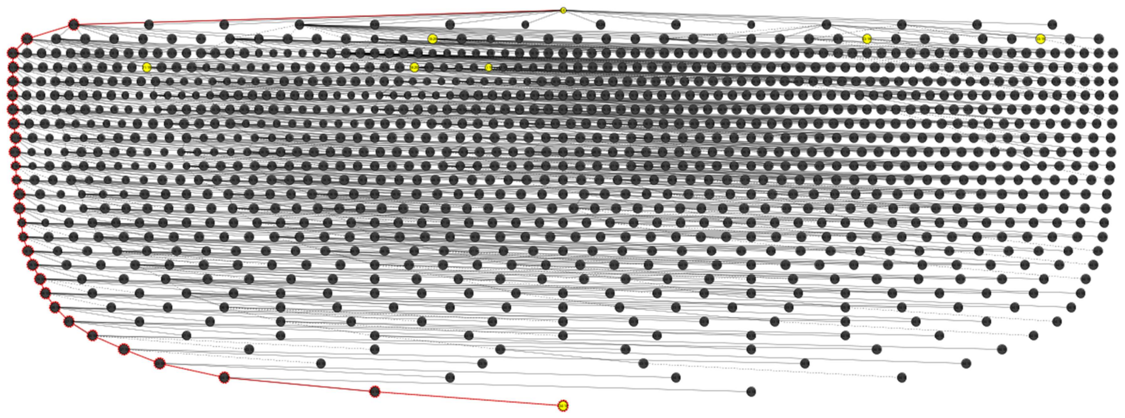


Figure 5-6 Episode enumeration tree, experiment 1

The most frequent episode is <18 15 > which occurred 29 times during 10 days (Figure 5-7). By looking into the Table 5-1, item ID 18 represents the pressure sensor of the floor tile located in (1,1) was turned on, while 15 represents turned off. This is because the occupant has to go across the place to get to the window. There is no ambient sensor data cluster associated, so no more information can be derived from this pattern.

Figure 5-7 Frequent episode chart for patter <18 15>, experiment 1

There is another pattern < (18 19) (15 16) > that occurred 20 times with the same events (Figure 5-8). The frequency is lower than the last one, but associated ambient data are found. The item 19 represents the time cluster ranging from 17:02:49 to 17:33:11 and 16 represents the time cluster from 17:03:40 to 17:34:42. With this pattern, we not only know the sequence of the events, but also the time period when the pattern happened.

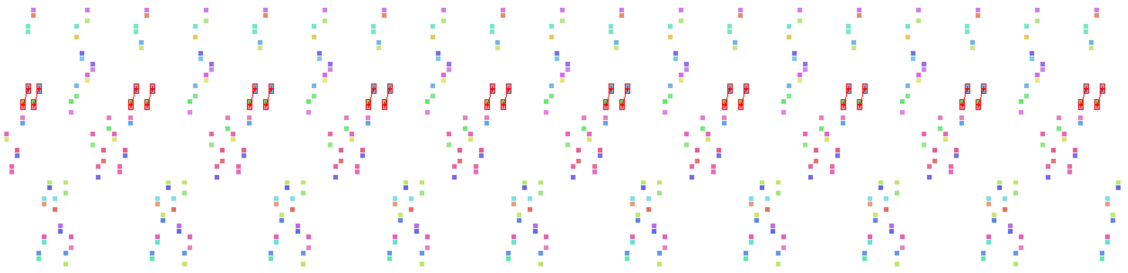


Figure 5-8 Frequent episode chart for patter <(18 19) (15.16)>, experiment 1

The longest frequent episode found is <(24 25) (30 31) (27 28) (18 19 21 22) (4 5 15 16) (1 2 18 19) (15 16 46 47) (36 37 43 44) (33 34) >, it is a length-28 episode that occurred 10 times (Figure 5-9). This episode can be translated into the sensor event sequent shown in Figure 5-10. Only half of the daily routine was detected. The reason the algorithm hasn't detected the whole pattern is because of the initial state of the home appliance. For example, if the window is already opened, the open window task won't be applied on the window again. In fact, some appliances were only triggered for 9 times.

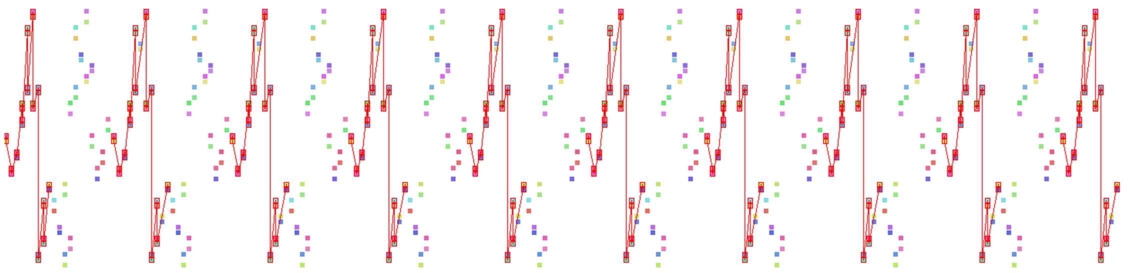


Figure 5-9 Frequent episode chart for patter <(24 25) (30 31) (27 28) (18 19 21 22) (4 5 15 16) (1 2 18 19) (15 16 46 47) (36 37 43 44) (33 34) >

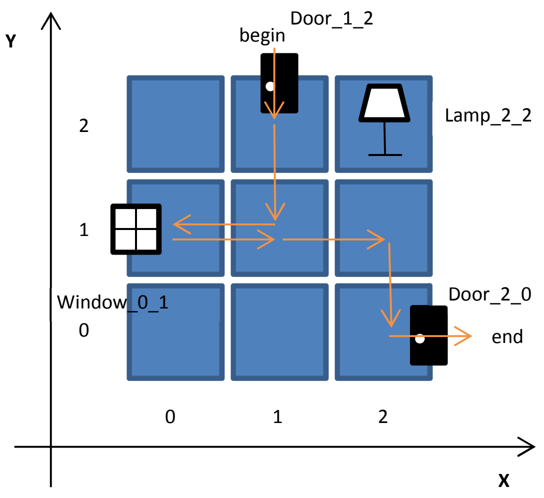


Figure 5-10 Detected pattern in vertural enviroment

5.2.3. Experiment 2

In this experiment, only one parameter was changed, i.e. the min_sup was set to 9 instead of 10. The daily routine and other parameters were kept same. Thus, the impact of the min_sup can be studied.

In the experiment, the mining process lasted for 93 iterations. Figure 5-11 shows the amount of different nodes of the frequent emulation tree in each iteration. The amount of total tree node is 10 times more than the last experiment, while the amount of unpruned nodes was same. In the final iteration, 14026 tree nodes were found, and 14019 of them were pruned. Only 7 nodes were left. They were:

1, <18 15 >:29
2, <36 33 >:19
3, <24 30 27 21 >:19
4, < (18 19) (15 16) >:20
5, <18 (4 15) 1 >:19
6, <46 43 >:19
7, <(24 25) (30 31) (27 28) (18 19 21 22) (4 5 15 16) (1 2 18 19) (15 16 46 47) (36 37 43 44) (33 34) (36 38) (41 42) (33 35 46 48) (18 20 43 45) (4 6 15 17) (9 10) (1 3 13 14) (11 12 24 26) (30 32) (27 29) (21 23) (24 25) (30 31) (27 28) (18 19 21 22) (4 5 15 16) (7 8) (1 2 18 19) (15 16 46 47) (36 37 43 44) (39 40) (33 34) >:9

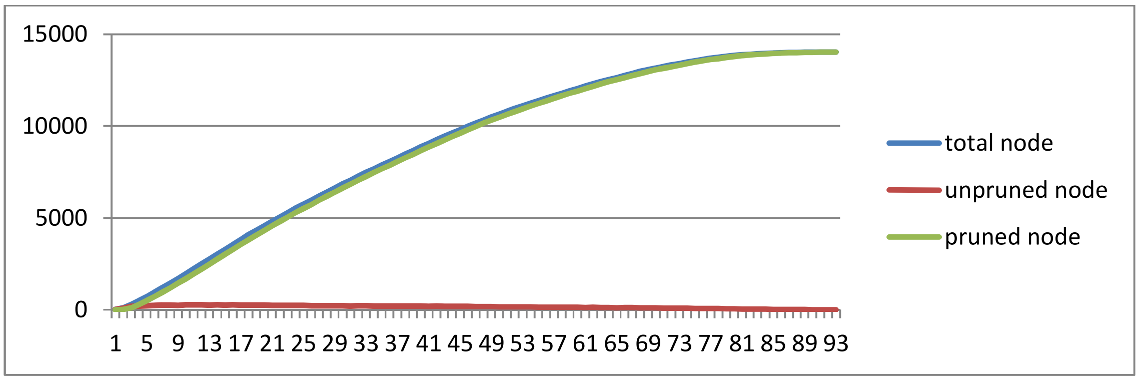


Figure 5-11 Node amount in the enumeration tree in each interation, expeirment 2

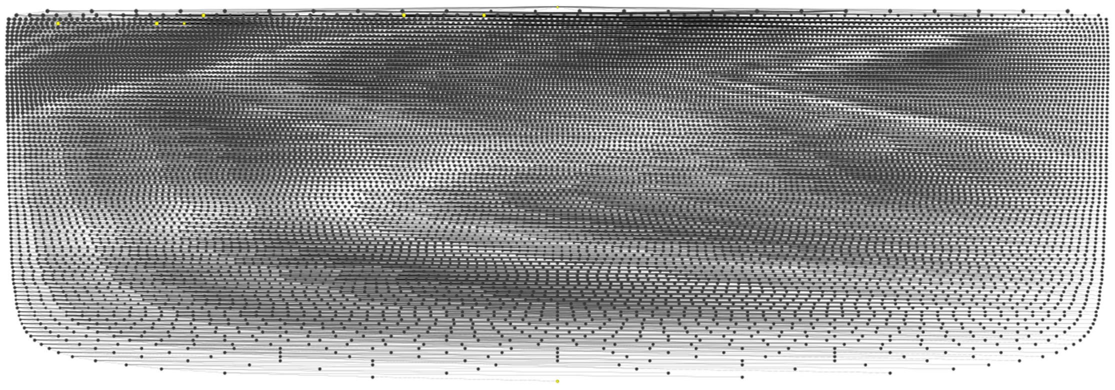


Figure 5-12 Episode enumeration tree, experiment 2

The final state of the enumeration tree is shown in Figure 5-12. Except the longest pattern, all the other frequent sequence remained the same. Figure 5-13 shows the occurrences of the seventh pattern. It is a length-93 episode and covers the whole daily routine as expected.

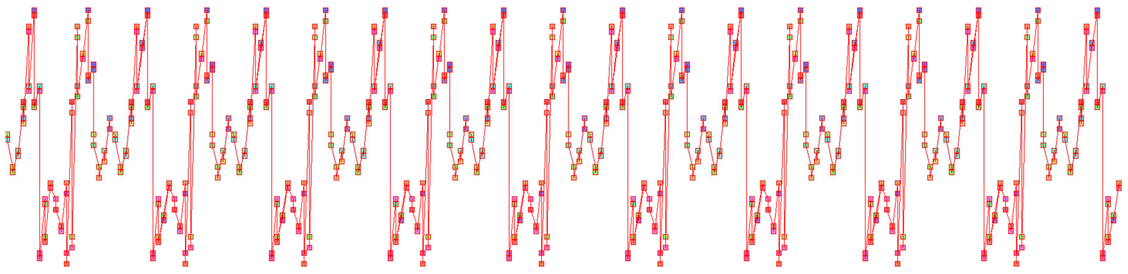


Figure 5-13 Frequent episode chart for patter 7, experiment 2

These two experiments show that the mining algorithm is able to detect the predefined daily routine. Additionally, it will also detect some sub patterns that have higher occurrence. The searching space of the algorithm not only depends on the parameters, but also the density and length of the patterns existing in the temporal database. The denser and the longer the pattern, the larger the searching space will be.

5.2.4. Experiment 3

In this experiment more randomness will be added into the temporal database by adding the random tasks into the routine and mixing different routines. Two routine scripts were mixed to generate 10 days of sensor records. The first script in Figure 5-14 was derived from the script used in experiment 1. Some random tasks were inserted into the routine. The second script contains only random tasks (Figure 5-15). These scripts were randomly selected for each day. The first script is selected for 7 times and the rest are the second script. 269 data samples and 44 items were generated. The min_sup was set as 6 and max gap was set to 3.

```
user1,TurnOnOff:Door_2_0:open,a:7:h,10:m
user1,TurnOnOff:Window_0_1:open,r:10:m,20:s
user1,Random,r:2:h,1:h
user1,TurnOnOff:Door_1_2:open,r:5:m,20:s
user1,TurnOnOff:Door_1_2:close,r:1:m,20:s
uset1,Walk:-1_-1,r:1:m,20:s
user1,TurnOnOff:Door_1_2:open,a:17:h,10:m
user1,TurnOnOff:Door_1_2:close,r:1:m,20:s
user1,Random,r:2:h,1:h
user1,TurnOnOff:Window_0_1:close,r:10:m,60:s
user1,Random,r:2:h,1:h
user1,TurnOnOff:Door_2_0:close,r:20:m,1:m
uset1,Walk:-1_-1,r:1:m,20:s
```

Figure 5-14 Script for describing the routine, experiment 3

```
user1,Random,a:7:h,1:h
user1,Random,r:2:h,1:h
user1,Random,r:2:h,1:h
user1,Random,r:2:h,1:h
user1,Random,r:2:h,1:h
user1,Random,r:2:h,1:h
user1,Random,r:2:h,1:h
user1,Random,r:2:h,1:h
```

Figure 5-15 Script for random routine, experiment 3

In the experiment, the mining process lasted for 38 iterations. Figure 5-16 shows the amount of different nodes of the frequent emulation tree in each iteration. In the final iteration, 1714 tree node was found, and 1692 of them were pruned. 22 nodes were left. They were:

```
1, < (24 26) 21 (1 19) 15 >:9
2, <30 27 30 (24 27) 23 2 (1 19) 15 >:7
3, <24 (6 21) 5 >:13
4, <24 21 24 21 >:9
5, <30 26 21 (1 19) 15 >:9
6, <30 34 32 27 >:13
7, <30 27 >:19
8, <37 36 >:9
9, <30 27 30 26 (21 23) 2 (1 19) 15 >:7
10, <24 21 19 15 >:13
11, <30 27 30 (24 27) 21 2 (1 19) 15 >:9
12, <30 (24 27) 21 24 21 24 21 >:6
13, <30 (24 27) 21 >:13
14, <24 21 >:22
15, <(30 31 ) (34 35 ) (32 33 ) (24 26 27 ) (6 21 ) 5 (1 19 ) 15 36 >:6
16, <19 15 >:16
17, <(30 31 ) (34 35 ) (32 33 ) (24 26 27 ) (6 21 ) 5 (1 19 ) 37 36 >:6
18, <(19 20 ) (15 16 24 25 ) (6 7 21 22 ) (11 12 ) (5 14 ) (13 30 ) 34 32 27 (30 31 ) (34 35 ) (32 33 )
(24 27 ) (6 21 ) 9 (2 5 ) (1 19 ) (15 37 ) 36 >:6
19, < (24 26) (21 23) 2 (1 19) 15 >:7
20, <24 21 30 27 >:9
21, <37 36 24 21 30 27 >:6
22, <30 (24 27) 21 24 21 19 15 >:6
```

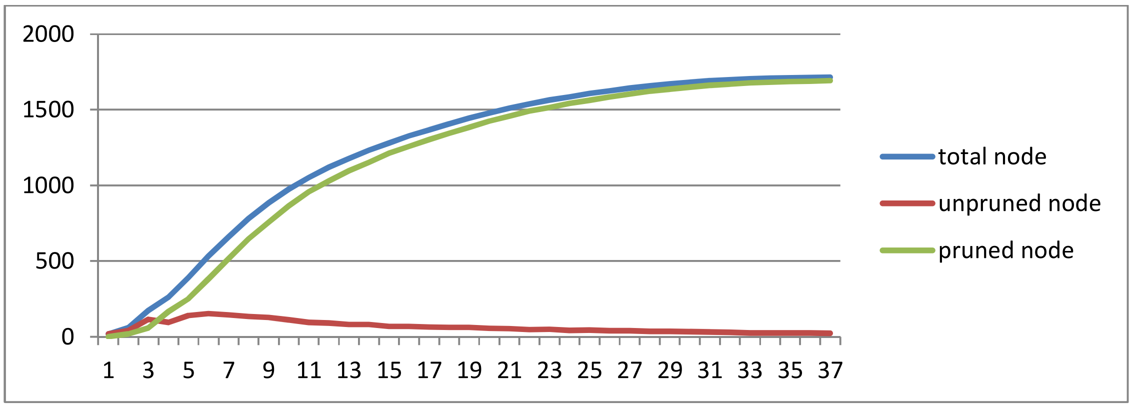


Figure 5-16 Node amount in the enumeration tree in each interation, expeirment 3

More patterns are detected. There are two reasons. Firstly, the min_sup value was smaller, so more patterns with lower occurrence will be found. Secondly, because of the randomness, parallel episodes occurred. Take the pattern 15 and 17 for example. The occurrence of pattern 15 and 17 are shown in the Figure 5-17 and Figure 5-18. The only difference between these two patterns is the last second item in the episode. One is 15 and the other is 37. This is caused by the occurrence sequence of these two items. Sometimes 15 occurred earlier, while sometimes 37. Similar situations also happened between pattern 12 and 22, pattern 9 and 11, pattern 1 and pattern 5. Such patterns will happen when there is no strong sequential relationship between two frequent events.

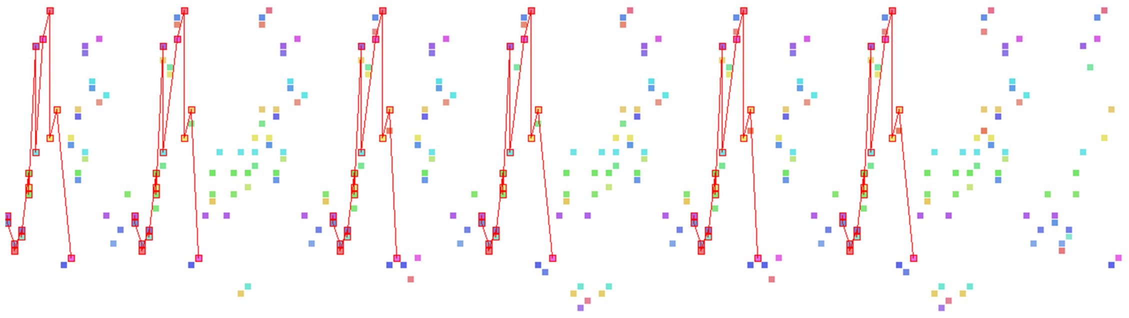


Figure 5-17 Frequent episode chart for patter 15, experiment 2

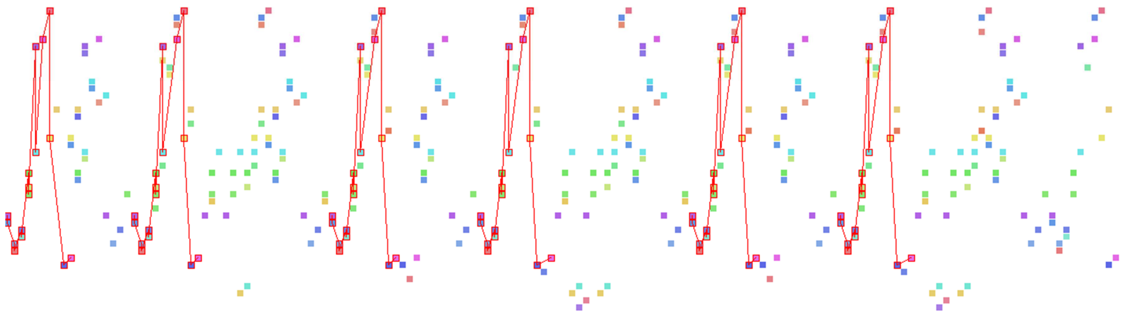


Figure 5-18 Frequent episode chart for patter 17, experiment 2

The longest episode found is pattern 18, which is length-38 episode (Figure 5-19). It covered the input routine.

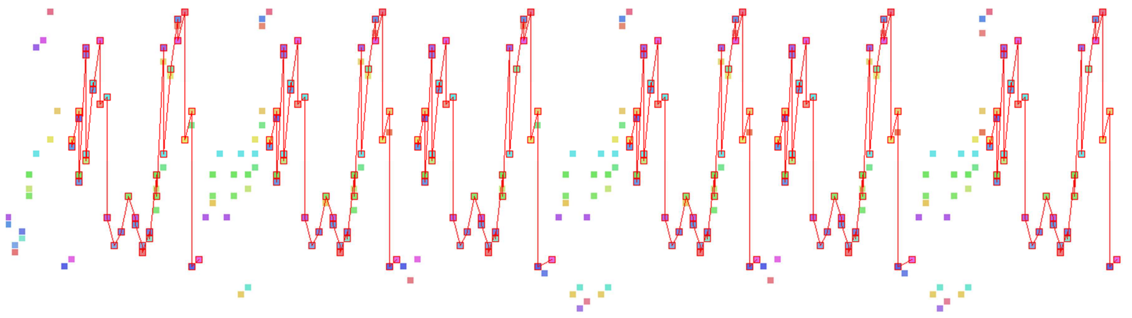


Figure 5-19 Frequent episode chart for patter 18, experiment 2

After testing with synthetic data, it was shown that the mining algorithm is able to detect the pattern correctly with noises, and the redundancy of the frequent episode is kept low. However, in the real world, there are more randomness and uncertainties in the data. The next experiments will try the algorithm on real life data.

5.3. The Experiment on Real Data

In this experiment, data collected with a WSN installed in a room of student dormitory for a one month period is used for testing the mining algorithm. Different parameters will be tested to see their impact on the mining results.

5.3.1. Description of the Data Source

The test environment was a 12 ㎡ single room, with one door, one window, one desk, one bed, and one basin (Figure 5-20). A self-organizing WSN with eight nodes was installed in the room, including one data logging node, five passive sensor nodes, and two active sensor nodes. Two of the passive sensor nodes were placed outdoors to monitor the outdoor, and corridor environment, respectively. The remaining three were placed next to the desk, bed, and basin, respectively (Figure 5-21). The active sensor nodes were respectively installed on the window frame and doo frame to monitor the opening and closing of the windows and doors.

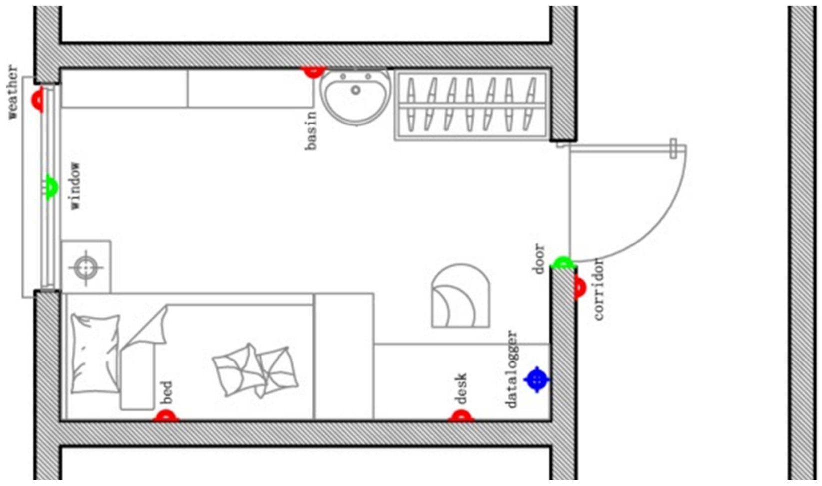


Figure 5-20 Plan of the testbed and installed sensors

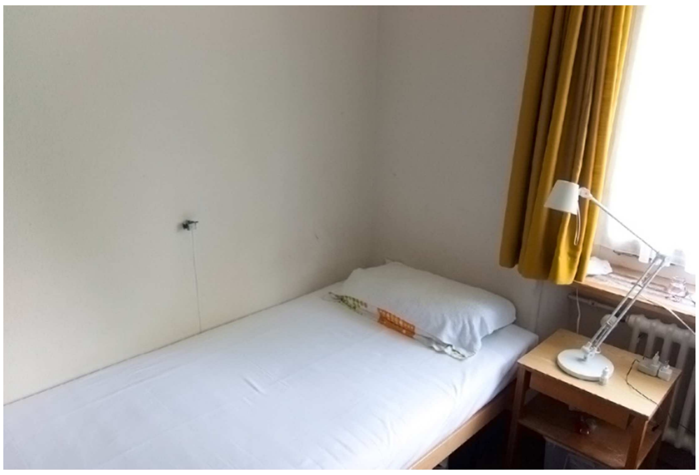


Figure 5-21 Passive sensor node installed next to the bed

On each passive sensor node was equipped with a light, temperature, and PIR sensor. The light and temperature sensors can be used to gather the ambient values. By detecting the jumps in light sensor values, the state of the room light can also be detected. In theory, the sensor node with PIR sensor should be an active sensor node. It should send immediately when PIR is triggered. However, in fact, the PIR sensor readings are very noisy. So, in this application, the node just buffered all the PIR detection and upload passively. The PIR sensor data were post processed to estimate the occupation of the bed and desk. Since the room is small, only one of the passive nodes was selected to provide the ambient data.

One month of sensor data was collected, from 1/8/2013 to 31/8/2013. After cleaning up, all the sensor data were converted into window, door, desk, bed, lamp, temperature, and light sensor tables. The window and door sensor tables recorded the open/ close of the window and door. Since the door will automatically close, only close events were recorded for simplicity. The desk and bed sensor tables recorded when they were occupied by the occupant. The lamp sensor table recorded the on/off of the lamp, and the temperature and humidity senor provided the ambient sensor values for the other sensor tables.

5.3.2. Experiment 4

In this experiment, only window, bed, desk, and lamp sensor table were selected as input dataset. Without the ambient data, this test was trying to find out if the mining algorithm can detect any temporal relationships in the real dataset. The effect of pruning approximately closed episode in reducing the redundancy will be also be examined.

There were 461 data samples. With a 1 hour cluster distance, 29 items were generated. The min_sup was set to 20, because there were 20 working days in that month. With 9 iterations, 250 frequent episodes were found out of 788 tree nodes in the enumeration tree. There were a lot of redundancies in the output patterns. For example, pattern <8 12 8 26 12 23 8 >:24 and <12 8 26 12 23 8 19 >:22 can be combine to one pattern <8 12 8 26 12 23 8 19>, if their difference in occurrences could be neglected. Pattern <12 8 12 8 26 12 23 >:21 and <12 8 12 8

26 12 >:23 can be merged into one pattern, because <12 8 12 8 26 12 > is a sub sequence of <12 8 12 8 26 12 23 > and their difference in support is only 2. This problem can be relieved by increasing the max_err_bound parameter. Figure 5-22 shows the total node and unpruned node under different max_err_bound. When max_err_bound was set to 5, the total node dropped to 403 and the unpruned node dropped to 85.

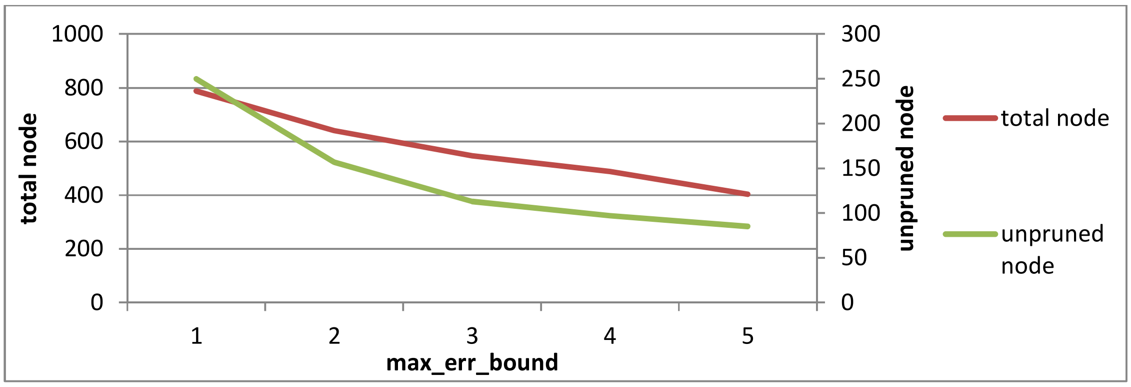


Figure 5-22 Mining result according to the max_err_bound

By changing the max_err_bound to 5, another round test has been done. 85 frequent episodes were found (see in Appendix A). Figure 5-23 shows the enumeration tree.

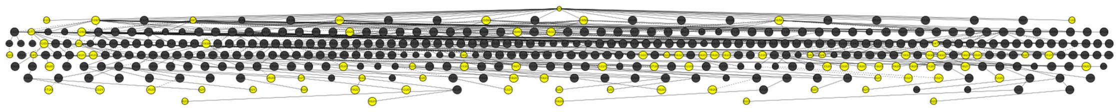


Figure 5-23 Episode enumeration tree, experiment 4

Patterns with associated time clusters can provide more valuable information. In these 85 patterns, there are 35 patterns containing associated time cluster information. 10 out of them contain associated time cluster information in every itemset in the episode. They were <(19 21) (16 18) >:22, <(12 13) (8 9) >:23, <(12 13) (19 21) >:21, <(26 28) (16 17) (23 25) >:20, <(26 27) (12 14) (8 10) >:21, <(12 14) (23 24) >:21, <(12 13) (16 18) >:20, <(26 27) (23 24) >:23, <(12 14) (19 20) >:20, <(19 20) >:26. By looking into the item ID to event and ambient value mapping table (Appendix B), these episodes can be translated back to the sensor event sequences. For example, pattern < (19 20) > means the PIR sensor next to bed was activated by the occupant around 22:00 (Figure 5-24), which can indicate the occupants sleeping time. The pattern < (19 21) (16 18) > can be translated into the bed PIR sensor was activated around 12:30 and deactivated around 13:00. This was probably recorded when the occupant was taking a nap after lunch (Figure 5-25). The longest pattern was <12 8 26 12 23 8 12 8 > (Figure 5-26), 12 represented the activation of the desk PIR sensor. 8 represented deactivation of the desk PIR sensor. 26 represented lamp on. 23 represented lamp off. However, without the associated time cluster information, it is really hard to be interpreted into the occupant’s ADLs.

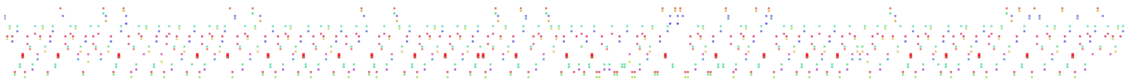


Figure 5-24 Frequent episode chart for patter < (19 20) >, experiment 4

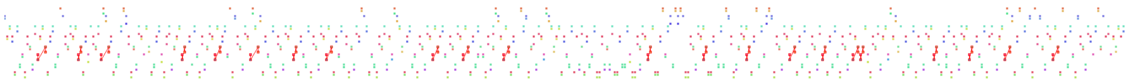


Figure 5-25 Frequent episode chart for patter < (19 21) (16 18) >, experiment 4

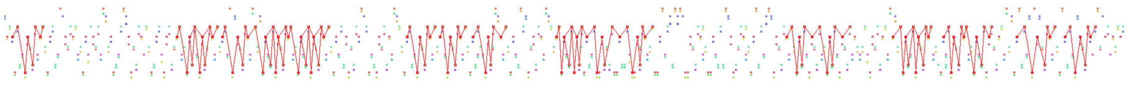


Figure 5-26 Frequent episode chart for patter <12 8 26 12 23 8 12 8 >, experiment 4

This experiment demonstrates that there are patterns that exist in peoples ADLs, which can be recorded by WSN, and that the frequent episode algorithm is able to detect them.

5.3.3. Experiment 5

In this experiment, the temperature and light sensor table were also added into the input dataset. This test was trying to find out if the mining algorithm is able to detect the correlation between the event and ambient values.

The rest of the parameters are kept same as the last experiment: the min_sup was set to 20, the constrain window size was set to 3, and the max_error_bound was set to 5. The maximal distances for temperature was set to 2, and for light was 10.732 data sample were recorded, and 75 items were generated. The searching ended up with 13 iterations. There were 1613 nodes in the enumeration tree (Figure 5-27), 1450 of them were pruned, and 163 were left (see in the Appendix C).

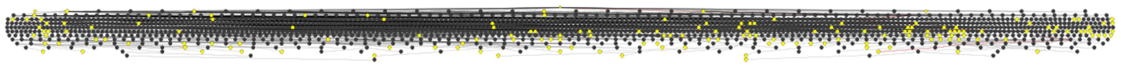


Figure 5-27 Episode enumeration tree, experiment 5

Patterns with more associated ambient value clusters are more meaningful. For example, with the mapping table (see in the Appendix D), the pattern <(25 28 34) (36 37 45) (59 61 67) >:20 (Figure 5-28) can be interpreted as: desk PIR sensor was activated around 6:35 with room light on, bed PIR sensor deactivated around 6:45 with room light, lamp turned off around 7:15, room light off. This was a recode of the period when the occupant got up in the morning and turned off the lamp before leaving the room. The desk PIR sensor activated before the bed PIR sensor, because there was a delay before PIR sensor deactivated.

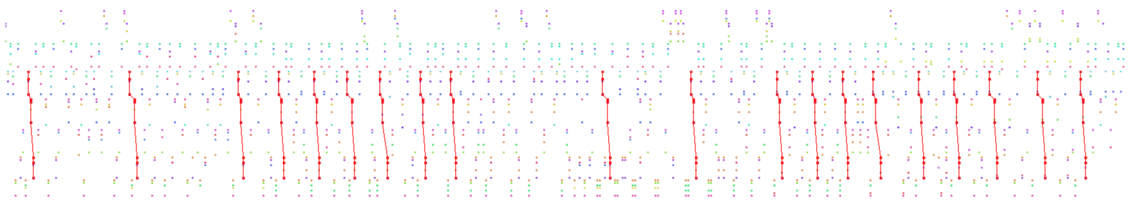


Figure 5-28 Frequent episode chart for patter <(25 28 34) (36 37 45) (59 61 67) >, experiment 5

The pattern <(68 70 74) (25 28 34) (59 61 67) (14 23) > can be interpreted as: Lamp was turned on around 6:24, when there was no light, then desk PIR was activated 6:35 with room light, around 7:15, lamp was turned off, and desk PIR deactivate without room light. The pattern also happened during the morning time before the occupant left room. Some events were over lapped with the last pattern. However, because of the randomness in raw data, their occurrences in the temporal database were different, which makes them two different patterns (Figure 5-29).

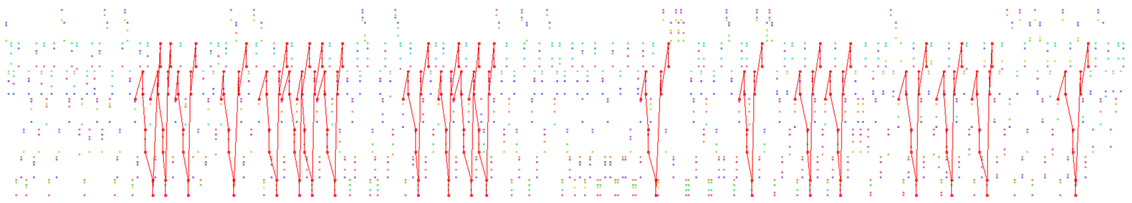


Figure 5-29 Frequent episode chart for patter <(68 70 74) (25 28 34) (59 61 67) (14 23) >, experiment 5

5.3.4. Experiment 6

The door sensor recorded 480 open and close samples. It took one third of the whole amount of samples, which means it occurred more frequently than the other sensor events. In this experiment, the door sensor records were added into the input dataset to check the impacts to the mining results when there was a single high frequency event in the dataset.

There were 1212 data sample in the input dataset; the other parameters were kept same as the last experiment. These data samples were converted into 84 different items. The searching lasted for 45 iterations. 13355 tree nodes were generated, 12793 of them were pruned, and 562 frequent episodes were found (Figure 5-30).

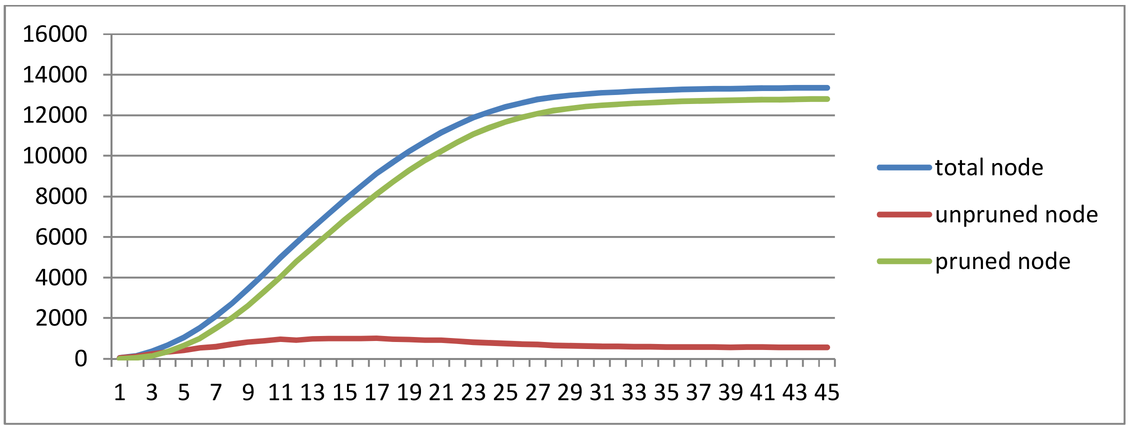


Figure 5-30 Node amount in the enumeration tree in each interation, experiment 6

Although, the pruning process already removed 95% of the tree nodes, 562 patterns were still a large amount of information to process. By looking into the composition of these patterns, we can discover that most of the patterns are just a sequence of the door events, for example <(75 76 81 83) (75 76 81 83) (75 76 81 83) (75 76 81 83) (75 76 81 83) >:26 (Figure 5-31), which means the door will be opened and closed for 5 times around 6:46. Such kind of information is not very valuable for analysis. Another problem caused by this kind of high frequent event is it will break the patterns exits in other events into small segments. Because the constrain window size limits the gap between each event, if the high frequent events will increase the gaps between the other events. To detect patterns, a larger constrain window is needed.

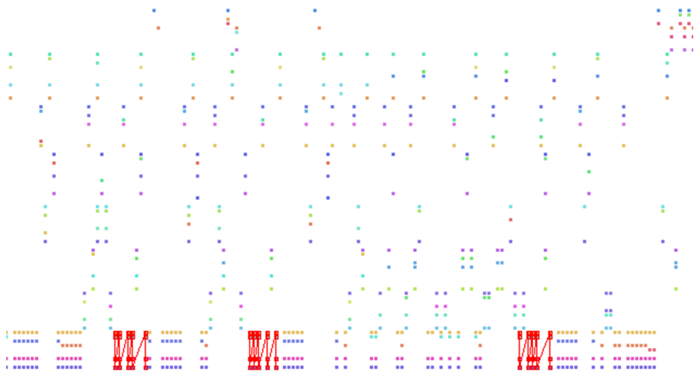


Figure 5-31 Frequent episode chart for patter <(68 70 74) (25 28 34) (59 61 67) (14 23) >, experiment 6

In this chapter, the whole framework was tested with both synthetic data and real life data. The test with synthetic data shows that with right parameters, the mining algorithm is able to detect the predefined patterns with noise and randomness and the redundancy of the pattern is kept low. The proposed encoding method for converting sensor data to a temporal database helps to find out the frequent sequence of events and also the scenarios when the events happened. With the real life data, it was demonstrated that there are patterns that exist in the sensor events, which can be detected with the mining algorithm. However, the redundancy is increased by parallel episodes. The mining efficiency will decrease when there are events that occur more often than the other events.

Chapter 6. Conclusions and future work

6.1. Summary

In chapter 2, a comprehensive review of the history and current state of smart home systems is provided. Two of the contemporary research topics are investigated more deeply, i.e. communication technologies and machine learning technologies. The existing problems in smart home products and research are discussed.

A framework that integrates both wireless sensor network and frequent episode mining is proposed for mining patterns exist in occupant’s daily activities. By dividing sensor data into event and ambient data, the amount of data acquisition and energy consumption of the sensor network can be brought down, and the outcome patterns of the mining algorithm can be more meaningful.

In chapter 3, the composition of the WSN is briefly introduced. Then, two different WSN for different applications are proposed. The self-organizing WSN is design for short-term applications, which can be easily deployed without configurations. The ultra-low power WSN is designed for long-term applications based on battery power. Both of them are designed to collect sensor data in the form of event and ambient data.

Chapter 4 introduces two algorithms: FSEEM and USTEM, derived from frequent episode mining algorithm for mining patterns from the proposed sensor networks. The preliminary knowledge for frequent episode mining is first introduced. By comparing several different mining algorithms, FEM is selected as the core algorithm. DBSCAN cluster is introduced to enable the algorithm mining on both discrete and continual data.

Chapter 5 introduces the whole framework, which integrates both data acquisition and data mining processes. The framework is validated with both synthetic data and real life data. Different combinations of parameters are tested to check their effect on mining results.

6.2. Problems

There are still some problems that exist in this approach.

- **Unpredictable searching space**

  The searching space of the mining algorithm is not only decided by the parameters, but also the density and length of the patterns in the temporal database. High density and long sequence will lead to large searching space. The efficiency of the mining algorithm depends on the sparsity of the patterns. However, this information is unknown before the mining. Small differences in density or length will lead to a huge difference in searching space.

- **The settings of the parameters**

  There are several parameters that need to be set. Some of them can be estimated by analyzing the dataset, such as the distance for clustering the ambient data. Some can be decided by the need of the application, for example, if it is mining daily patterns, than the minimal support can be set equal to the amount of days in the dataset. However, the setting of the approximate for pruning the approximately closed sequence and the constrain window size are decide by the character of the input dataset. Several tests of different values are needed before getting a good result.

- **Redundancy in result**

  In the mining result with the real life dataset, there are a lot of parallel patterns, which are very similar with each other. However, the proposed mining algorithm doesn't support mining parallel episode. A parallel episode is separated into several single episodes.

6.3. Future works

- **More complex scenario**

  Because of the limitation of the project scale and privacy issues, the available real life data is very limited. The data from some third party research groups can't provide enough sensor data or metadata for this research. In future research, a multi room, multi occupant apartment will be selected as a test bed. More sensor nodes will be installed to monitor more appliances. In a more complex scenario, the performance of the proposed algorithm will be investigated.

- **More condense pattern**

  To make the output pattern more condense, a post process step will be added to create the parallel episode, and the techniques for selecting suitable parameters will be studied.

- **The application of the pattern**

  This thesis only discusses the method to find out existing patterns, but not how to make use of them in prediction. There are two possible ways to apply. The first is being used directly in making predictions of the usage of home appliances and inhabitants' daily activities. The second is finding out the right feature vector for other prediction algorithms, such as ANNs. Events and ambient values in the same episode usually have a strong correlation. Including these highly correlated facts in the input feature vector will help to improve the accuracy of the ANN prediction.

List of Figures

List of Tables

References

Agrawal, R., T. Imieli, #324, ski and A. Swami (1993). "Mining association rules between sets of items in large databases." SIGMOD Rec. **22**(2): 207-216.

Agrawal, R. and R. Srikant (1995). Mining sequential patterns. Data Engineering, 1995. Proceedings of the Eleventh International Conference on.

Akyildiz, I. F., W. Su, Y. Sankarasubramaniam and E. Cayirci (2002). "Wireless sensor networks: a survey." Computer Networks **38**(4): 393-422.

Akyildiz, I. F. and M. C. Vuran (2010). Wireless Sensor Networks Wiley.

Al-Karaki, J. N. and A. E. Kamal (2004). "Routing techniques in wireless sensor networks: a survey." Wireless communications, IEEE **11**(6): 6-28.

Aldrich, F. (2003). "Smart homes: past, present and future." Inside the smart home: 17-39.

Anders, A. (2011) "EnOcean Technology – Energy Harvesting Wireless." 6.

Atmel (2014) "ATmega640/1280/1281/2560/2561 Summary."

Bação, F. and V. Lobo (2004) "Introduction to Kohonen’s Self-Organizing Maps." 22.

Badlani, A. and S. Bhanot (2011). Smart Home System Design based on Artificial Neural Networks. Proceedings of the World Congress on Engineering and Computer Science.

Balta-Ozkan, N., B. Boteler and O. Amerighi (2014). "European smart home market development: Public views on technical and economic aspects across the United Kingdom, Germany and Italy." Energy Research & Social Science **3**: 65-77.

Begg, R. and R. Hassan (2006). Artificial Neural Networks in Smart Homes - Designing Smart Homes. J. Augusto and C. Nugent, Springer Berlin / Heidelberg. **4008:** 146-164.

Beutel, J., O. Kasten, F. Mattern, K. Römer, F. Siegemund and L. Thiele (2004). "Prototyping wireless sensor network applications with BTnodes." Wireless Sensor Networks: 323-338.

Blunsom, P. (2004). "Hidden markov models." Lecture notes **15**: 18-19.

Chan, M., D. Estève, C. Escriba and E. Campo (2008). "A review of smart homes—Present state and future challenges." Computer Methods and Programs in Biomedicine **91**(1): 55-81.

Chee-Yee, C. and S. P. Kumar (2003). "Sensor networks: evolution, opportunities, and challenges." Proceedings of the IEEE **91**(8): 1247-1256.

Chen, L. and I. Khalil (2011). Activity Recognition: Approaches, Practices and Trends. Activity Recognition in Pervasive Intelligent Environments. L. Chen, C. D. Nugent, J. Biswas and J. Hoey, Atlantis Press. **4:** 1-31.

Choi, J., D. Shin and D. Shin (2005). Research on Design and Implementation of the Artificial Intelligence Agent for Smart Home Based on Support Vector Machine. Advances in Natural Computation. L. Wang, K. Chen and Y. Ong, Springer Berlin Heidelberg. **3610:** 1185-1188.

Cook, D. J., J. C. Augusto and V. R. Jakkula (2009). "Ambient intelligence: Technologies, applications, and opportunities." Pervasive and Mobile Computing **5**(4): 277-298.

Cortes, C. and V. Vapnik (1995). "Support-Vector Networks." Mach. Learn. **20**(3): 273-297.

Cristianini, N. (2001) "Support Vector and Kernel Machines."

Curtiss, P. S., J. F. Kreider and M. J. Brandemuehl (1994). Local and global control of commercial building HVAC systems using artificial neural networks. American Control Conference, 1994.

Dange, H. V. and V. K. Gondi (2011). Powerline Communication Based Home Automation and Electricity Distribution System. Process Automation, Control and Computing (PACC), 2011 International Conference on.

Das, S. K., D. J. Cook, A. Battacharya, E. O. Heierman, III and L. Tze-Yun (2002). "The role of prediction algorithms in the MavHome smart home architecture." Wireless Communications, IEEE **9**(6): 77-84.

Demiris, G., M. J. Rantz, M. A. Aud, K. D. Marek, H. W. Tyrer, M. Skubic and A. A. Hussam (2004). "Older adults' attitudes towards and perceptions of'smart home'technologies: a pilot study." Informatics for Health and Social Care **29**(2): 87-94.

Dewsbury, G., K. Clarke, M. Rouncefield and I. Sommerville (2002). "Appropriate home technology: Depending on dependable technology systems."

Digi (2008) "Wireless Mesh Networking ZigbBee vs. DigiMesh white paper."

Dulman, S. and P. Havinga (2002). "Operating system fundamentals for the EYES distributed sensor network." Progress Report.

Eckl, R. and A. MacWilliams (2009). "Smart Home Challenges and Approaches to Solve Them: A Practical Industrial Perspective." Intelligent Interactive Assistance and Mobile Multimedia Computing **53**: 119-130.

Edwards, W. and R. Grinter (2001). At home with ubiquitous computing: Seven challenges. Ubicomp 2001: Ubiquitous Computing, Springer.

Estrin, D., L. Girod, G. Pottie and M. Srivastava (2001). Instrumenting the world with wireless sensor networks. Acoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on, IEEE. **4:** 2033-2036.

Eunju, K., S. Helal and D. Cook (2010). "Human Activity Recognition and Pattern Discovery." Pervasive Computing, IEEE **9**(1): 48-53.

Faludi, R. (2010). Building Wireless Sensor Networks: with ZigBee, XBee, Arduino, and Processing, O'Reilly Media, Incorporated.

Fatima, I., M. Fahim, Y.-K. Lee and S. Lee (2013). "A Unified Framework for Activity Recognition-Based Behavior Analysis and Action Prediction in Smart Homes." Sensors **13**(2): 2682-2699.

Fayyad, U., G. Piatetsky-Shapiro and P. Smyth (1996). "From data mining to knowledge discovery in databases." AI magazine **17**(3): 37.

Fleury, A., N. Noury and M. Vacher (2009). Supervised classification of activities of daily living in health smart homes using SVM. Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE.

Gan, M. and H. Dai (2011). Fast mining of non-derivable episode rules in complex sequences. Modeling Decision for Artificial Intelligence, Springer**:** 67-78.

Gan, M. and H. Dai (2012). "An efficient one-pass method for discovering bases of recently frequent episodes over online data streams." International Journal of Innovative, Computing Information and Control **8**(7).

Gan, M. and H. Dai (2012). "Mining condensed sets of frequent episodes with more accurate frequencies from complex sequences." International Journal of Innovative Computing, Information and Control **8**(1).

Gan, M. and H. Dai (2012). Subsequence Frequency Measurement and its Impact on Reliability of Knowledge Discovery in Single Sequences. Reliable Knowledge Discovery. H. Dai, J. N. K. Liu and E. Smirnov, Springer US**:** 239-255.

Gill, K., S. H. Yang, F. Yao and X. Lu (2009). "A ZigBee-based home automation system." Consumer Electronics, IEEE Transactions on **55**(2): 422-430.

Guralnik, V. and K. Z. Haigh (2002). Learning models of human behaviour with sequential patterns. Proceedings of the AAAI-02 workshop “Automation as Caregiver.

Gurney, K. and K. N. Gurney (1997). An introduction to neural networks, CRC Press.
Han, J., R. Afshar and X. Yan (2003). CloSpan: Mining: Closed Sequential Patterns in Large Datasets. Proceedings of the 2003 SIAM International Conference on Data Mining**:** 166-177.
Han, J., H. Cheng, D. Xin and X. Yan (2007). "Frequent pattern mining: current status and future directions." Data Mining and Knowledge Discovery **15**(1): 55-86.
Han, J., J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal and M.-C. Hsu (2000). FreeSpan: frequent pattern-projected sequential pattern mining. Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. Boston, Massachusetts, USA, ACM**:** 355-359.
Han, J., J. Pei and Y. Yin (2000). Mining frequent patterns without candidate generation. Proceedings of the 2000 ACM SIGMOD international conference on Management of data. Dallas, Texas, USA, ACM**:** 1-12.
Hebb, D. O. (1949). The organization of behavior: a neuropsychological theory, Wiley.
hgfdSuryadevara, N. K. and S. C. Mukhopadhyay (2015). Smart Homes: Design, Implementation and Issues, Springer.
Hill, J. L. and D. E. Culler (2002). "Mica: A wireless platform for deeply embedded networks." Micro, IEEE **22**(6): 12-24.
Hinton, G. E., S. Osindero and Y.-W. Teh (2006). "A fast learning algorithm for deep belief nets." Neural computation **18**(7): 1527-1554.
Holroyd, P., P. Watten and P. Newbury (2010). "Why Is My Home Not Smart?" Aging Friendly Technology for Health and Independence **6159**: 53-59.
Hopfield, J. J. (1982). "Neural networks and physical systems with emergent collective computational abilities." Proceedings of the national academy of sciences **79**(8): 2554-2558.
Huiru, Z., W. Haiying and N. Black (2008). Human Activity Detection in Smart Home Environment with Self-Adaptive Neural Networks. Networking, Sensing and Control, 2008. ICNSC 2008. IEEE International Conference on.
Ibrahim, A., S. Sastry and P. Sastry (2014). "Discovering Compressing Serial Episodes from Event Sequences." arXiv preprint arXiv:1401.1043.
iotee. (2015). "Internet of Things Wireless Connectivity Option Analysis: Pros and Cons of Bluetooth Classic, Bluetooth Low Energy, and CSRmesh." from https://iotee.wordpress.com/2015/03/10/internet-of-things-wireless-connectivity-option-bluetooth-classic-bluetooth-low-energy-and-csrmesh/.
Iwanuma, K., R. Ishihara, Y. Takano and H. Nabeshima (2005). Extracting frequent subsequences from a single long data sequence a novel anti-monotonic measure and a simple on-line algorithm. Data Mining, Fifth IEEE International Conference on.
Jain, A. K. and R. C. Dubes (1988). Algorithms for clustering data, Prentice-Hall, Inc.
Jianyong, W. and H. Jiawei (2004). BIDE: efficient mining of frequent closed sequences. Data Engineering, 2004. Proceedings. 20th International Conference on.
Kahn, J. M., R. H. Katz and K. S. J. Pister (1999). Next century challenges: mobile networking for "Smart Dust". Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking, ACM.
Kalogirou, S. A. and M. Bojic (2000). "Artificial neural networks for the prediction of the energy consumption of a passive solar building." Energy **25**(5): 479-491.
Karl, H. and A. Willig (2007). Protocols and architectures for wireless sensor networks, Wiley-Interscience.
Kasteren, T. v., A. Noulas, G. Englebienne, B. Kr, #246 and se (2008). Accurate activity recognition in a home setting. Proceedings of the 10th international conference on Ubiquitous computing. Seoul, Korea, ACM**:** 1-9.

Katz, S., A. B. Ford, R. W. Moskowitz, B. A. Jackson and M. W. Jaffe (1963). "Studies of illness in the aged: the index of ADL: a standardized measure of biological and psychosocial function." Jama **185**(12): 914-919.
Kazem Sohraby, Daniel Minoli and T. Znati (2007). Wireless Sensor Networks: Technology, Protocols, and Applications Wiley-Interscience.
Khalid, M. and S. Omatu (1992). "A neural network controller for a temperature control system." Control Systems, IEEE **12**(3): 58-64.
Kim, E. (2013). "Everything You Wanted to Know about the Kernel Trick." from http://www.eric-kim.net/eric-kim-net/posts/1/kernel_trick.html.
Kohonen, T. (1972). "Correlation Matrix Memories." Computers, IEEE Transactions on **C-21**(4): 353-359.
Kohonen, T. (1982). "Self-organized formation of topologically correct feature maps." Biological Cybernetics **43**(1): 59-69.
Kohonen, T. (1990). "The self-organizing map." Proceedings of the IEEE **78**(9): 1464-1480.
Kohonen, T. (2001). Self-Organizing Maps, Springer-Verlag GmbH.
Kovacs, G. T. A. (1998). Micromachined transducers sourcebook, WCB/McGraw-Hill New York, NY.
Kumar, S. and D. Shepherd (2001). SensIT: Sensor information technology for the warfighter. Proc. 4th Int. Conf. on Information Fusion.
Lühr, S., G. West and S. Venkatesh (2007). "Recognition of emergent human behaviour in a smart home: A data mining approach." Pervasive and Mobile Computing **3**(2): 95-116.
Lam, H. T., T. Calders, J. Yang, F. M, #246, rchen and D. Fradkin (2013). Zips: mining compressing sequential patterns in streams. Proceedings of the ACM SIGKDD Workshop on Interactive Data Exploration and Analytics. Chicago, Illinois, ACM**:** 54-62.
Lam, H. T., F. Mörchen, D. Fradkin and T. Calders (2014). "Mining Compressing Sequential Patterns." Statistical Analysis and Data Mining **7**(1): 34-52.
Lewis, F. L. (2004). "Wireless sensor networks." Smart Environments: Technologies, Protocols, and Applications: 11-46.
LSR. (2015). "ZigBee VS 6LoWPAN for Sensor Networks." from https://www.lsr.com/white-papers/zigbee-vs-6lowpan-for-sensor-networks.
Lu, J., T. Sookoor, V. Srinivasan, G. Gao, B. Holben, J. Stankovic, E. Field and K. Whitehouse (2010). The smart thermostat: using occupancy sensors to save energy in homes. Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, ACM.
Lymberopoulos, D., A. Bamis and A. Savvides (2011). "Extracting spatiotemporal human activity patterns in assisted living using a home sensor network." Universal Access in the Information Society **10**(2): 125-138.
Mabroukeh, N. R. and C. I. Ezeife (2010). "A taxonomy of sequential pattern mining algorithms." ACM Comput. Surv. **43**(1): 1-41.
Machado, C. and J. A. Mendes (2009). "Automatic light control in domotics using artificial neural networks." International Journal of Computer Systems Science and Engineering **4**(2).
Mainardi, E. and M. Bonfè (2008). Powerline Communication in Home-Building Automation Systems, INTECH Open Access Publisher.
Malfa, S. L. (2010). Wireless Sensor Networks.
Mannila, H. and H. Toivonen (1996). Discovering Generalized Episodes Using Minimal Occurrences. KDD.
Mannila, H., H. Toivonen and A. Inkeri Verkamo (1997). "Discovery of Frequent Episodes in Event Sequences." Data Mining and Knowledge Discovery **1**(3): 259-289.

Mannila, H., H. Toivonen and A. I. Verkamo (1995). Discovering frequent episodes in sequences Extended abstract. Proceedings the first Conference on Knowledge Discovery and Data Mining.
Mayrhofer, R., H. Radi and A. Ferscha (2003). Recognizing and predicting context by learning from user behavior. The International Conference On Advances in Mobile Multimedia (MoMM2003).
McCulloch, W. and W. Pitts (1943). "A logical calculus of the ideas immanent in nervous activity." The bulletin of mathematical biophysics **5**(4): 115-133.
Min, G. and D. Honghua (2010). A Study on the Accuracy of Frequency Measures and Its Impact on Knowledge Discovery in Single Sequences. Data Mining Workshops (ICDMW), 2010 IEEE International Conference on.
Moutacalli, M. T., A. Bouzouane and B. Bouchard (2012). Unsupervised Activity Recognition using Temporal Data Mining. SMART 2012, The First International Conference on Smart Systems, Devices and Technologies.
Mozer, M. (1998). "The neural network house: An environment that adapts to its inhabitants." Proceedings of the American Association for Artificial Intelligence.
Mozer, M. C., L. Vidmar and R. H. Dodier (1997). "The neurothermostat: Predictive optimal control of residential heating systems." Advances in Neural Information Processing Systems: 953-959.
Nazerfard, E., P. Rashidi and D. J. Cook (2011). Using association rule mining to discover temporal relations of daily activities. Toward Useful Services for Elderly and People with Disabilities, Springer**:** 49-56.
Nowozin, S., P. V. Gehler, J. Jancsary and C. H. Lampert (2014). Advanced Structured Prediction, MIT Press.
Oja, E. and S. Kaski (1999). Kohonen Maps, Elsevier Science.
Oppermann, I., L. Stoica, A. Rabbachin, Z. Shelby and J. Haapola (2004). "UWB wireless sensor networks: UWEN-a practical example." Communications Magazine, IEEE **42**(12): S27-S32.
Pei, J., J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal and M.-C. Hsu (2004). "Mining sequential patterns by pattern-growth: The prefixspan approach." Knowledge and Data Engineering, IEEE Transactions on **16**(11): 1424-1440.
Pinto, H., J. Han, J. Pei, K. Wang, Q. Chen and U. Dayal (2001). Multi-dimensional sequential pattern mining. Proceedings of the tenth international conference on Information and knowledge management. Atlanta, Georgia, USA, ACM**:** 81-88.
Qandour, A., D. Habibi and I. Ahmad (2012). Wireless sensor networks for fire emergency and gas detection. Networking, Sensing and Control (ICNSC), 2012 9th IEEE International Conference on, IEEE.
Quoc Cuong, N., S. Dongil, S. Dongkyoo and K. Juhan (2009). Real-Time Human Tracker Based on Location and Motion Recognition of User for Smart Home. Multimedia and Ubiquitous Engineering, 2009. MUE '09. Third International Conference on.
Rashidi, P. and D. J. Cook (2013). "COM: A method for mining and monitoring human activity patterns in home-based health monitoring systems." ACM Transactions on Intelligent Systems and Technology (TIST) **4**(4): 64.
Rashidi, P., D. J. Cook, L. B. Holder and M. Schmitter-Edgecombe (2011). "Discovering Activities to Recognize and Track in a Smart Environment." Knowledge and Data Engineering, IEEE Transactions on **23**(4): 527-539.
Reilly, D., L. Cooper and C. Elbaum (1982). "A neural model for category learning." Biological Cybernetics **45**(1): 35-41.
Reilly, D. L. and L. N. Cooper (1990). "An overview of neural networks: Early models to real world systems." An introduction to neural and electronic networks: 229-250.

Rissanen, J. (1978). "Modeling by shortest data description." Automatica **14**(5): 465-471.
Rivera-Illingworth, F., V. Callaghan and H. Hagras (2005). A neural network agent based approach to activity detection in AmI environments.
Romer, K. and F. Mattern (2004). "The design space of wireless sensor networks." Wireless Communications, IEEE **11**(6): 54-61.
Rosenblatt, F. (1957). The perceptron, a perceiving and recognizing automaton Project Para, Cornell Aeronautical Laboratory.
Rovsing, P. E., P. G. Larsen, T. S. Toftegaard, D. Lux and S. ApS (2011). "A Reality Check on Home Automation Technologies." Journal of Green Engineering **303**: 327.
Rovsing, P. G. L. P. E. and T. S. Toftegaard "CHALLENGES IN GAINING LARGE SCALE CARBON REDUCTIONS THROUGH WIRELESS HOME AUTOMATION."
Roy, A., S. K. Das Bhaumik, A. Bhattacharya, K. Basu, D. J. Cook and S. K. Das (2003). Location aware resource management in smart homes. Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on.
Rumelhart, D. E., G. E. Hintont and R. J. Williams (1986). "Learning representations by back-propagating errors." Nature **323**(6088): 533-536.
Sadler, B. M. and A. Swami (2006). Synchronization in Sensor Networks: an Overview. Military Communications Conference, 2006. MILCOM 2006. IEEE.
Schiller, J., A. Liers, H. Ritter, R. Winter and T. Voigt (2005). Scatterweb-low power sensor nodes and energy aware routing. System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on, IEEE.
Seah, W. K. G., Z. A. Eu and H. P. Tan (2009). Wireless sensor networks powered by ambient energy harvesting (WSN-HEAP)-Survey and challenges. Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology, 2009. Wireless VITAE 2009. 1st International Conference on, IEEE.
Sichitiu, M. L. and C. Veerarittiphan (2003). Simple, accurate time synchronization for wireless sensor networks. Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE, IEEE.
Simon, P. (2013). Too Big to Ignore: The Business Case for Big Data, Wiley.
Singh, S. K., M. Singh and D. Singh (2010). "Routing protocols in wireless sensor networks–A survey." International Journal of Computer Science & Engineering Survey (IJCSES) Vol **1**: 63-83.
Srikant, R. and R. Agrawal (1996). Mining sequential patterns: Generalizations and performance improvements, Springer.
Sun, Q., W. Yu, N. Kochurov, Q. Hao and F. Hu (2013). "A multi-agent-based intelligent sensor and actuator network design for smart house and home automation." Journal of Sensor and Actuator Networks **2**(3): 557-588.
Tao, G., W. Liang, W. Zhanqing, T. Xianping and L. Jian (2011). "A Pattern Mining Approach to Sensor-Based Human Activity Recognition." Knowledge and Data Engineering, IEEE Transactions on **23**(9): 1359-1372.
Tapia, E., S. Intille and K. Larson (2004). "Activity recognition in the home using simple and ubiquitous sensors." Pervasive Computing: 158-175.
Trajanoski, Z. (2008) "Self-Organizing Maps." 22.
Tzvetkov, P., X. Yan and J. Han (2005). "TSP: Mining top-k closed sequential patterns." Knowledge and Information Systems **7**(4): 438-457.
Vazquez, F. I. and W. Kastner (2011). Clustering methods for occupancy prediction in smart home control. Industrial Electronics (ISIE), 2011 IEEE International Symposium on.
Vieira, M. A. M., C. N. Coelho Jr, D. C. da Silva Jr and J. M. da Mata (2003). Survey on wireless sensor network devices. Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA'03. IEEE Conference, IEEE. **1:** 537-544.

Vintan, L., A. Gellert, J. Petzold and T. Ungerer (2004). Person Movement Prediction Using Neural Networks.

Weiss, S. and C. Kulikowski (1991). "Computer systems that learn."

Widrow, B. and M. E. Hoff (1960). "Adaptive switching circuits."

Wikipedia. (2015). "DBSCAN." from https://en.wikipedia.org/wiki/DBSCAN.

Yang, I.-H., M.-S. Yeo and K.-W. Kim (2003). "Application of artificial neural network to predict the optimal start time for heating system in building." Energy Conversion and Management **44**(17): 2791-2809.

Zaki, M. J. (1998). Efficient enumeration of frequent sequences. Proceedings of the seventh international conference on Information and knowledge management. Bethesda, Maryland, USA, ACM**:** 68-75.

Zanaty, E. A. (2012). "Support Vector Machines (SVMs) versus Multilayer Perception (MLP) in data classification." Egyptian Informatics Journal **13**(3): 177-183.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor, Prof. Dr. Ludger Hovestadt, for the continuous support of my PhD study and related research. I have been amazingly fortunate to have an advisor who gave me both guidance and freedom to explore on my own.

I owe my deepest gratitude to my co-supervisor, Prof. Dr. Li Biao, and the supervisor of my graduate study, Prof. Dr. Han Dongqing, since with their recommendation and support, I got the chance to study in the highest research institution in the digital architecture field.

I consider it an honor to work with Christoph Wartmann, Daniel Macgregor, Andreas Wassmer, Sandro Gaehler, and Jan Haller, who are all colleagues in the Nexiot AG. They give me the chance to turn my research into real products.

I am indebted to Hua Hao, Benjamin Dillenburger, Michael Hansmeyer, Manuel Kretzer, Vera Bühlmann, Mathias Bernhard, Vahid Moosavi, Nikola Marinčić, Jorge Orozco, Mihye An, and my many colleagues in the chair of CAAD, who supported me and shared a lot of great ideas in their research with me. Special thanks should be given to Mario Guala and Sibylla Spycher for their assistance and advice.

I wish to express my appreciation to the China Scholarship Council, Education Affairs Office of the Embassy of China, and Southeast University. They made my study in Switzerland possible.

Last but not least, I would like to thank my family: my wife, Dong Jia, and my parents for supporting me spiritually throughout writing this thesis and my life in general.

Curriculum vitae

Personal details

Name:	LI LI
Date of Birth:	18.08.1984
Place of Birth:	Changshu, Jiangsu
Nationality:	China

Education

Since 2011	Doctoral study	Chair of Computer-Aided Architectural Design, ETH Zurich Prof. Dr. Ludger Hovestadt
2008-2011	Master	School of Architecture, Southeast University, Nanjing Supervisor: Prof. Dr. Han Dongqing
2003-2008	Bachelor	School of Architecture, Southeast University, Nanjing

Academic Experience

Since 2015.3	Co-founder	Nexiot AG, Switzerland, ETH spin-off
2011.9-2015.3	Researcher	Embedded Lab, CAAD, ETH
2010.8-2010.10	Assistant	Undergraduate architectural design course, School of Architecture, Southeast University
2008.10-2008.11	Assistant	CAAD and Architecture course, School of Architecture, Southeast University

Papers

2014.4	A Self-organizing Wireless Sensor Network for Indoor Environment Surveillance	Proceedings of the DADA2013 International Conference on Digital
2012.12	The optimization of architectural shape based on Genetic Algorithm	Frontiers of Architectural Research
2011.6	The Application of Genetic Algorithm in Architectural Design	Master Thesis
2010.6	The Sight Line Control and Its Application in the Architectural design based on Rhinoscript	New Architecture 06/2010
2010.10	Routing in Mountain Environment with Multi Agent System	Proceeding of the 8th National Annual Meeting of Architecture & Urban Planning Postgraduates

Appendix

Appendix A 85 frequent episodes found in experiment 4

<12 16 12 8 16 12 8 >:21	<(26 27) (12 14) 23 >:20	<8 12 8 26 12 8 12 >:20
<26 12 23 (8 11) >:20	<12 8 12 >:51	<12 16 12 19 26 12 23 8 >:22
<19 (26 28) (12 15) (23 25) >:20	<12 8 12 8 >:45	<12 8 12 19 >:32
<12 16 (12 13) 19 >:20	<8 12 8 18 >:20	<19 26 12 23 8 >:31
<8 28 (12 15) (23 25) 8 >:20	<12 8 19 >:39	<(26 28) (16 17) (8 11) 12 >:21
<26 12 23 8 12 16 12 19 >:20	<12 (16 17) 13 19 >:20	<12 (16 17) (8 11) 19 16 >:20
<8 12 >:60	<19 (26 28) (12 15) (16 17) >:20	<12 23 (12 13) 19 >:20
<(19 21) (16 18) >:22	<12 8 19 16 12 >:20	<12 (16 17) (8 11) 12 8 >:20
<8 12 8 (26 27) >:20	<16 12 8 >:47	<(26 28) (16 17) 8 19 >:20
<19 26 12 23 8 12 8 >:22	<12 8 12 19 16 >:21	<12 8 12 8 26 12 8 >:21
<12 8 26 12 23 8 12 8 >:23	<12 23 12 16 12 8 >:21	<19 16 >:33
<(12 13) (8 9) >:23	<12 8 26 12 23 8 12 16 >:21	<19 16 26 12 >:20
<12 8 >:77	<(12 13) (19 21) >:21	<12 16 12 8 26 12 23 8 >:22
<8 12 8 >:53	<16 27 (12 14) 23 >:20	<(26 27) (23 24) >:23
<12 16 12 19 >:38	<(19 20) >:26	<12 8 12 (19 21) >:20
<19 26 12 (23 25) 8 >:20	<(26 28) (16 17) (23 25) >:20	<(12 14) (19 20) >:20
<12 16 23 8 19 >:21	<26 12 23 8 >:40	<19 >:58
<4 >:23	<23 >:59	<12 8 12 8 16 12 8 >:20
<19 16 12 8 >:24	<19 16 12 19 >:22	<12 8 19 26 12 23 8 >:21
<12 >:87	<8 12 8 26 12 8 >:27	<(12 14) (8 10) 26 >:20
<26 >:59	<(26 27) (12 14) (8 10) >:21	<12 8 12 8 26 12 23 >:21
<12 (16 17) 8 21 (16 18) >:20	<12 16 >:49	<16 >:58
<1 >:22	<16 12 (8 9) >:20	<8 >:87
<19 26 12 23 8 19 >:23	<16 27 (12 14) 19 >:20	<12 8 12 16 >:32
<12 16 12 19 16 12 >:20	<16 27 (12 14) (8 10) >:21	<(12 13) (16 18) >:20
<16 12 8 19 >:23	<12 8 26 12 23 8 >:33	
<8 12 (8 9) >:22	<12 16 23 8 12 >:21	
<12 (16 17) (23 25) >:20	<(26 27) (12 14) 19 >:20	
<12 8 (12 13) >:22	<12 8 26 12 23 8 19 >:22	
<12 16 12 8 >:40	<(12 14) (23 24) >:21	

Appendix B 85 Item ID to event and associated amtient cluster mapping table, experiment 4

Item ID	Event	Ambient
1	Window:fals	
2		Time{22:53:31~23:16:20}
3	Window:true	
4		Time{05:56:10~06:21:57}
5	Desk_PIR:false	
6		Time{11:42:17~12:52:44}
7		Time{21:08:56~22:20:28}
8		Time{06:47:22~07:48:25}
9		Time{22:37:26~23:26:29}
10	Desk_PIR:true	
11		Time{11:28:22~12:43:41}
12		Time{18:39:48~20:49:17}
13		Time{06:08:27~07:05:57}
14	Bed_PIR:false	
15		Time{06:36:08~08:00:32}
16		Time{12:46:52~13:29:07}
17	Bed_PIR:true	
18		Time{21:16:22~23:17:48}
19		Time{12:05:46~13:08:28}
20	Lamp:false	
21		Time{20:58:40~22:10:29}
22		Time{06:47:02~07:48:15}
23		Time{22:29:19~23:16:28}
24	Lamp:true	
25		Time{19:33:49~20:44:49}
26		Time{18:42:58~19:25:25}
27		Time{05:53:38~06:45:10}
28		Time{18:16:05~18:25:51}

Appendix C frequent episodes found in experiment 5

<(48 57) (68 74) (25 34) (36 45) >:21
<48 36 (68 74) (25 34) >:20
<(14 23) (25 34) 14 36 (25 34) 14 >:20
<(25 33) 14 >:21
<(48 49 57) >:26
<(25 34) (59 67) (14 23) (25 34) 36 (25 34) >:20
<36 (25 29 34) 14 >:20
<48 36 >:33
<36 (25 34) (14 20 23) >:20
<(25 34) (14 23) (68 74) (25 29 34) >:20
<48 25 36 14 >:20
<48 (68 74) (25 34) (59 67) (14 23) 48 >:21
<36 >:58
<48 (68 74) (25 34) (59 67) (14 23) (25 34) >:20
<36 (14 17 23) 25 48 >:20
<(14 20) (68 74) >:21
<(25 29 34) 14 (25 29) >:20
<(14 23) (25 34) >:49
<(68 74) (59 67) (14 23) 57 >:20
<(25 34) (14 23) (25 34) 14 36 25 >:20
<14 (25 34) >:55
<(25 29 34) (14 23) >:32
<(25 34) 36 (25 34) 48 36 >:22
<48 (68 74) (25 34) (59 67) (14 23) 25 14 >:21
<(25 34) (59 67) (14 23) >:42
<(25 34) 14 (25 34) 57 >:22
<(36 37 45) (59 61 67) 14 >:20
<(48 57) >:46
<36 (14 17 23) 25 14 >:20
<14 (68 74) (25 34) (59 67) 57 >:20
<36 (25 34) 14 48 >:22
<48 (68 74) (25 34) (59 67) (14 23) >:29
<14 (68 69 74) (25 27 34) (14

<(25 34) (14 17 23) (25 34) 48 >:20
<(25 34) 36 (25 34) (14 23) >:30
<(36 41) 25 >:20
<(25 34) (59 67) (14 23) (48 57) >:21
<(25 34) (59 67) (14 17 23) (25 34) >:20
<(68 74) (59 67) >:45
<(59 61 67) (14 17 23) (25 34) >:20
<14 >:87
<(68 74) (59 67) (14 23) 68 (25 34) 59 14 >:20
<(25 27 34) (14 16 23) 68 >:20
<(25 28 34) (59 61 67) 34 >:20
<36 (25 26) (14 15) >:20
<(14 23) (68 74) (25 34) (59 61 67) 14 >:20
<36 (25 34) (14 23) (68 74) (25 34) (59 67) (14 23) >:21
<14 (25 34) (14 23) (68 74) (25 34) (59 67) >:21
<(25 34) (14 23) 48 (68 74) (25 34) (59 67) >:21
<(14 23) 68 (28 34) (59 61) 14 >:20
<(25 34) (14 23) (48 57) >:27
<36 (25 34) 48 36 25 >:20
<(25 34) (14 23) (25 34) 14 34 >:21
<36 (69 74) (25 27 34) (14 16 23) >:21
<(25 34) (59 67) (25 26) >:20
<(25 28 34) (59 61 67) (14 17 23) 25 >:20
<(25 27 34) (59 60 67) >:21
<(68 74) (59 67) (14 20 23) >:22
<36 (25 34) 14 >:44
<(14 23) (25 34) 14 >:43
<(68 70 74) (25 28 34) (36 37 45) 14 >:20

<(25 34) (14 23) (48 50) >:20
<(25 34) (59 67) (14 23) 48 36 >:20
<48 (68 74) (25 29 34) >:20
<(25 28 34) (36 37) 26 48 >:20
<(25 34) 36 (25 34) 14 36 25 >:20
<36 (25 34) (14 23) >:37
<36 (25 34) (48 57) (68 74) (25 34) (59 67) >:20
<14 (25 34) (14 23) (68 74) (25 34) (14 23) >:21
<(48 57) (68 70) (25 28 34) (36 37) >:20
<(14 23) (25 34) (14 20) >:20
<(68 69 74) (59 60 67) >:23
<14 (25 34) (14 23) >:42
<(25 28 34) (36 37 45) (59 61 67) >:20
<(68 74) (59 67) (14 23) (25 34) 36 (25 34) 14 >:20
<(36 37 45) (14 17 23) (25 34) >:20
<(25 34) (14 23) 48 (68 74) (25 34) (14 23) >:22
<(14 23) (25 26 34) 48 >:20
<(25 34) (59 67) 25 48 >:21
<(25 34) 36 (25 34) 48 >:34
<(48 57) (68 70) (25 28 34) (59 61) >:20
<(48 57) (68 74) (25 34) (59 67) (14 23) >:23
<36 14 57 >:20
<(25 29 34) (59 67) (14 23) (25 34) >:20
<(25 34) (14 20 23) >:31
<(25 34) 36 (25 34) (14 23) 68 (25 34) >:20
<(25 27 34) (49 57) >:20
<(68 74) (59 64 67) >:20
<25 >:87
<36 (25 34) 48 >:40
<(25 34) (14 23) (25 34) 14 >:37
<36 (69 74) (25 27 34) (48

23) >:20
<(68 74) (59 67) (14 23) (25 34) 36 (25 34) 48 >:20
<(14 23) (68 74) (25 29 34) (59 67) >:20
<(14 23) (68 74) (25 34) (36 45) >:20
<(25 34) 14 (68 74) (59 67) (14 23) (25 34) 14 >:20
<48 >:58
<(25 28 34) (36 37 45) (14 17 23) 25 >:21
<14 (25 34) 14 >:49
<(25 34) >:81
<(14 23) (25 26 34) (14 15) >:20
<(25 34) 36 >:46
<(14 20 23) (25 34) (14 23) >:20
<36 (25 34) (48 57) (68 74) (25 34) (14 23) >:20
<(25 34) 36 (25 26) 48 >:20
<(25 34) (36 45) 59 (14 23) >:20
<(25 34) (59 67) (14 23) (25 34) (14 23) >:24
<(25 34) 36 (25 34) 48 (68 74) (25 34) (14 23) >:20
<(25 34) 14 >:72
<(25 34) (14 23) 25 50 >:20
<48 36 (25 34) (14 23) >:22
<(25 34) 14 38 >:20
<(14 23) (25 34) 14 (68 74) (25 34) >:21
<(68 74) (59 67) (14 23) (25 34) (14 23) >:23
<(25 29 34) >:41
<(25 34) (59 67) (14 23) 48 >:28
<(25 28 34) (36 37 45) (14 17 23) 48 >:20
<36 (14 17 23) (48 50) >:20
<(25 34) (14 23) (25 34) 48 36 >:20
<(68 74) >:58
<36 (25 34) (48 57) >:32
<14 (68 74) (59 67) (48 57) >:20
<(25 34) (14 23) >:64
<(25 34) (59 67) (14 20 23) >:22
<48 36 (25 34) (48 57) >:20
<(59 64 67) (14 20 23) >:20
<(68 72 74) >:23
<(25 29 34) (59 64 67) >:20
<(25 34) (59 67) (14 23) 48 68 (25 34) >:20
<36 (14 17 23) 48 36 >:20
<(68 74) (59 67) (14 23) >:39
<(25 34) (14 20 23) (25 34) >:22
<(25 34) (36 45) 14 34 14 >:20
<(25 34) (14 23) 48 >:36
<(25 26) (36 38) >:20
<(25 29 34) 48 >:20
<(25 28 34) (36 37 45) 14 50 >:20
<36 (25 34) 14 36 (25 34) 14 >:20
<(25 29 34) (36 45) >:20
<(59 67) >:57
<6 >:23
<1 >:22
<(68 70 74) (25 28 34) (17 23) >:20
<(14 23) (25 26) 50 >:21
<14 25 38 >:20
<(14 20 23) >:37
<(25 34) 36 (25 34) 14 >:36
<(25 29 34) (14 20 23) >:22
<(14 20) (29 34) >:21
<(25 34) (14 23) (25 34) 48 >:27
<(25 28 34) (59 61 67) 14 48 >:21
<(48 57) (68 74) (25 34) (59 61 67) >:20
57) >:20
<(25 28 34) (36 37) 34 14 >:20
<(14 23) >:74
<(25 28 34) (59 61 67) 26 >:20
<(48 50) (36 38) >:22
<(68 70 74) (25 28 34) (59 61 67) (14 23) >:20
<36 (14 17 23) (25 26) >:20
<(25 34) 36 (25 34) (14 23) (68 74) >:21
<36 (69 74) (25 27 34) (59 67) >:20
<36 (14 23) 48 38 >:20
<(25 34) (14 17 23) (25 26) >:20
<(25 34) 36 (25 34) (48 57) >:26

Appendix D Item ID to event and associated amtient cluster mapping table, experiment 5

Item ID	Event	Ambient
1	Window:false	
2		Time{22:53:31~23:16:20}
3		Temperature{711.8~715.62}
4		Light_sensor{0~11.4}
5	Window:true	
6		Time{05:56:10~06:21:57}
7		Temperature{725.07~725.9}
8		Temperature{711.55~714.1}
9		Temperature{689.9~692.93}
10		Light_sensor{0~19.52}
11	Desk_PIR:false	
12		Time{11:42:17~12:52:44}
13		Time{21:08:56~22:20:28}
14		Time{06:47:22~07:48:25}
15		Time{22:37:26~23:26:29}
16		Temperature{718.04~728.49}
17		Temperature{729.85~734.14}
18		Temperature{695.05~715.52}
19		Temperature{683.47~692.54}
20		Light_sensor{73.32~75.03}
21		Light_sensor{0~62.19}
22		Light_sensor{86.84~100.37}
23	Desk_PIR:true	
24		Time{11:28:22~12:43:41}
25		Time{18:39:48~20:49:17}
26		Time{06:08:27~07:05:57}
27		Temperature{703.7~726.23}
28		Temperature{739.18~742.27}
29		Temperature{745~748.12}
30		Temperature{728.52~731.1}
31		Temperature{688.45~700.98}
32		Light_sensor{0~89.22}
33		Light_sensor{96.8~107.7}
34	Bed_PIR:false	
35		Time{06:36:08~08:00:32}
36		Time{12:46:52~13:29:07}
37		Temperature{739.87~742.98}
38		Temperature{723.17~727.03}
39		Temperature{697.14~709.56}
40		Temperature{714~715.36}
41		Temperature{690.2~692.56}
42		Temperature{684.31~688.19}
43		Light_sensor{0.1~31.56}

44		Light_sensor{38.28~95.9}
45		Light_sensor{108.18~115.88}
46	Bed_PIR:true	
47		Time{21:16:22~23:17:48}
48		Time{12:05:46~13:08:28}
49		Temperature{730.74~733.08}
50		Temperature{703.68~707.9}
51		Temperature{709.87~715.52}
52		Temperature{695.04~701.3}
53		Temperature{683.2~685.71}
54		Light_sensor{0~56.36}
55		Light_sensor{74.61~111.45}
56	Lamp:false	
57		Time{20:58:40~22:10:29}
58		Time{06:47:02~07:48:15}
59		Time{22:29:19~23:16:28}
60		Temperature{718.88~727.27}
61		Temperature{690.58~692.68}
62		Temperature{703.85~715.53}
63		Temperature{695.21~697.12}
64		Temperature{685.67~688.18}
65		Light_sensor{0~19.42}
66	Lamp:true	
67		Time{19:33:49~20:44:49}
68		Time{18:42:58~19:25:25}
69		Time{05:53:38~06:45:10}
70		Time{18:16:05~18:25:51}
71		Temperature{719.48~725.77}
72		Temperature{704.37~715.39}
73		Temperature{696.6~698.7}
74		Light_sensor{0~15.06}