# Activity monitoring through home automation devices

Julie Soulas

UNIVERSITE
BRETAGNE
LOIRE

**THÈSE / Télécom Bretagne**
sous le sceau de l'Université Bretagne Loire

pour obtenir le grade de Docteur de Télécom Bretagne
En accréditation conjointe avec l'Ecole Doctorale Sicma
Mention : Sciences et Technologies de l'Information et de la Communication

présentée par
**Julie Soulas**
préparée dans le département Logique des usages, sciences sociales & sciences de l'information
Laboratoire Labsticc

# Activity monitoring through home automation devices

Thèse soutenue le 29 mars 2016
Devant le jury composé de :

**Pascal Poncelet**
Professeur, Université de Montpellier / président

**João Gama**
Associate Professor, Université de Porto – Portugal / rapporteur

**Xiaolan Xie**
Professeur, École nationale supérieure des mines de Saint-Étienne / rapporteur

**Jérôme Boudy**
Professeur, Télécom SudParis / examinateur

**André Thépaut**
Directeur d'études, Télécom Brestagne / examinateur

**Philippe Lenca**
Professeur, Télécom Bretagne / directeur de thèse

**N° d'ordre** : 2016telb0377

**Sous le sceau de l'Université Bretagne Loire**

# Télécom Bretagne

**En accréditation conjointe avec l'École Doctorale Sicma**

# ACTIVITY MONITORING THROUGH HOME AUTOMATION DEVICES

## Thèse de Doctorat

Mention : STIC

Présentée par **Julie Soulas**

Département : Logique des usages, sciences sociales et sciences de l'information

Laboratoire : Lab-STICC, Pôle : CID

Directeur de thèse : Philippe Lenca

Soutenue le 29 mars 2016

**Jury :**

M. João Gama, Associate Professor, Université de Porto (Rapporteur)
M. Xiaolan Xie, Professeur, Mines de Saint-Étienne (Rapporteur)
M. Philippe Lenca, Professeur, Télécom Bretagne (Directeur de thèse)
M. André Thépaut, Directeur d'études, Télécom Bretagne (Encadrant de thèse)
M. Jérôme Boudy, Professeur, Télécom SudParis (Examinateur)
M. Pascal Poncelet, Professeur, Université de Montpellier (Examinateur)

**Abstract**

The aging of the population in the coming decades raises new challenges in order to help elderly people live longer at home, independently and safely. The emergence of assistive technologies, and in particular home automation devices, sensor networks and communication devices open up new opportunities to ease the interactions between the elderly and their environment, and to monitor their health status remotely day after day. In particular, the home automation sensors record information on the activity in the home and make it possible to assess autonomy and well-being.

In this thesis, we focus on the mining of the data recorded by such sensor networks. We focus more particularly on the discovery of habits. Indeed, the daily routines help maintain autonomy. We thus propose unsupervised data mining algorithms for the discovery and description of periodic behaviors. A first contribution, the extended Episode Discovery algorithm (xED), allows the discovery of such habits in transactional data, and the characterization of their variability. xED was further developed for the handling of event streams and the update of the habits when time passes. A top-$k$ approach is also described, for the discovery and update of regular patterns in event streams. The interactions with the human supervisor (physician, caregiver, family member) have also been studied, which lead to the design of a supervision model, combining sequence mining with expert knowledge allows a custom monitoring, tailored to the exact expectations of the supervisor. All of these contributions are evaluated on real-life datasets, that have been extensively used in the literature.

These contributions opens the way towards innovative methods for the monitoring of isolated individuals, in particular thanks to the personalized adaptation of the home, anomaly detection, or the analysis of the evolution of habits and heath status.

**Résumé**

Le vieillissement de la population mondiale au cours des décennies à venir pose de nouveaux défis pour la prise en charge, le logement, et les soins à apporter à cette population. L'un de ces défis est de permettre aux personnes âgées, souvent isolées et fragilisée de vivre dans leur domicile personnel le plus longtemps possible, et dans les meilleures conditions de confort et de sécurité. L'émergence de nouvelles technologies domotiques, des réseaux de capteurs et de communication offrent de nouvelles opportunités pour faciliter les interactions des personnes âgées avec leur environnement, et pour assurer un suivi médical peu intrusif au jour le jour. En particulier, les capteurs domotiques collectent des informations sur l'activité dans le logement, et permettent d'évaluer l'autonomie ainsi que l'état de santé de la personne suivie.

Dans cette thèse, nous nous concentrons sur l'exploitation des données enregistrées par de tels réseaux de capteurs. Nous nous intéressons en particulier à la découverte des habitudes, puisqu'elles jouent un rôle important dans le maintien de l'autonomie chez les personnes âgées. Nous proposons donc ainsi plusieurs méthodes pour la caractérisation et la découverte non supervisée des habitudes. Une première contribution, extended Episode Discovery (xED), est un algorithme permettant la découverte de telles habitudes dans des données statiques, et la caractérisation de leur variabilité. xED a ensuite été étendu pour la gestion de flots de données, et la mise à jour des habitudes en fonction de leur évolution au cours du temps. Une troisième approche, TKRES, est également proposée pour la découverte et la mise à jour des k épisodes les plus réguliers dans les flots d'événements. Ces différentes contributions sont évaluées qualitativement et quantitativement sur des jeux de données réelles, issues de la littérature du domaine.

Ces contributions ouvrent la voie pour la mise en place de nouvelles méthodes pour le monitoring d'individus isolés, notamment pour l'adaptation personnalisée du logement, la détection d'anomalies, ou l'analyse de l'évolution des habitudes et de l'état de santé.

# Remerciements

Un travail de thèse n'est pas le fruit du travail d'une personne isolée du reste du monde. C'est plutôt le résultat de nombreuses discussions, réunions et collaborations. Cette page me donne l'opportunité d'exprimer toute mon appréciation à ceux qui m'ont aidé et soutenu d'une façon ou d'une autre au cours de cette thèse à Télécom Bretagne.

Je voudrais exprimer mes remerciements à mes encadrants, Philippe Lenca et André Thépaut. Leurs conseils et leur encadrement m'ont donné la confiance nécessaire à la réalisation des travaux de thèse. Je les remercie de m'avoir guidée à travers la découverte du monde de la recherche.

Un remerciement particulier va à l'intention de Jacques Simonin et de Komate Amphawan. Ce fut un formidable plaisir de travailler avec eux, et une expérience très enrichissante de découvrir de nouveaux domaines de recherche et de participer à des thématiques de recherche aux confins entre les leurs et les miennes.

J'aimerais également remercier João Gama, Associate professor à l'université de Porto, et Xiaolan Xie, Professeur à l'école de Mines de Saint-Étienne d'avoir accepté de rapporter mon travail. Merci également à Jérôme Boudy, Professeur à Télécom SudParis et Pascal Poncelet, Professeur à l'université de Montpellier d'avoir accepté de faire partie du jury de cette thèse, en tant qu'examinateurs.

Merci également à tous les membres de l'équipe DECIDE du Lab-STICC, ainsi qu'aux membres du projet transverse HAAL, pour le partage de connaissances et d'idées. Merci au département LUSSI de Télécom Bretagne, et en particuliers les doctorants, pour l'amitié et le soutien au jour le jour.

Enfin, j'aimerai adresser ma gratitude à mes parents et ma famille, et à tous les amis pour leurs encouragements, leur support constants durant la thèse.

# Contents

# List of Tables

# List of Figures

# List of Algorithms

# List of Acronyms

| | |
|---|---|
| **AIC** | Akaike Information Criterion |
| **BIC** | Bayesian Information Criterion |
| **DBSCAN** | Density-Based Spatial Clustering of Applications with Noise |
| **EM** | Expectation-Maximization |
| **FEL** | Frequent Episode Lattice |
| **FP-growth** | Frequent Pattern-growth |
| **GMM** | Gaussian Mixture Model |
| **MDE** | Model Driven Engineering |
| **MDL** | Minimum Description Length |
| **MO** | Minimal Occurrence |
| **NOMO** | Non-Overlapping Minimal Occurrence |
| **PG** | Pattern Growth |
| **RMN** | Recently Modified Nodes |
| **sxED** | extended Episode Discovery for data Streams |
| **TKRES** | Top-$k$ Regular Episodes in Streams |
| **TQ** | Time Queue |
| **WF** | Window frequency |
| **xED** | extended Episode Discovery |

# General introduction

## Research scope

With the aging of the population, especially in the industrialized countries, people now live longer, and wish to continue living in their homes for as long as possible. The elderly population is however frailer than the rest of the population. Living alone at home is thus more dangerous for them.

Security and well-being at home is usually improved thanks to the home visits of caregivers, medical staff and housekeeping services. The house is also adapted to the reduced strength and mobility of elderly people, for instance thanks to stair-climbing devices, support bars, or electric shutters. Adapted communication devices, such as emergency pendants, allow them to call for help whenever they need to.

Moreover, the emergence and wide spread of sensing and communication technologies now allows new services to develop, such as home automation and home hospitalization. These technologies offer promising opportunities for aging at home: home automation eases the interactions between older people and their environment, the communication devices help keep the link with distant family members and facilitate the social insertion in the local community. The sensors disseminated in the home continuously record activity traces. These traces form patterns which carry information on the health and well-being of the inhabitant, and are thus very valuable for the physicians, caregivers and family members. There is thus an increasing need for monitoring at home, via the analysis of sensor data.

The pursued objectives for such continuous monitoring include for instance fall and danger detection, activity recognition, autonomy assessment, trend analysis, etc. The mining of the sensor logs remains however challenging.

## Objectives

The objective of this thesis is to contribute to the data analysis field for activity monitoring. In particular, routines and habits are prominent behaviors in the daily life, which have not been deeply investigated yet. This thesis focuses thus mostly on the discovery and description of personal habits. We propose a description formalism to characterize habits, as well as several algorithms and tools for their unsupervised discovery.

## Outline of this thesis

Chapter 1 introduces the socio-demographic context and details the current approaches for the monitoring of the behavior at home, based on home-sensing technologies. A particular focus is set on activity monitoring and the challenges it entails.

Chapter 2 characterizes a class of human behaviors: the habits. Existing habit monitoring frameworks are described and compared. Most existing approaches present a major drawback: they do not handle well the variability inherent in human life. We thus propose a new periodicity model for the description of human habits. This model not only tolerates variability, but also characterizes it. It is thus a powerful descriptive tool, which we then use in our contributions described in chapters 3 and 4.

Chapter 3 details the e**x**tended **E**pisode **D**iscovery (xED) algorithm (Soulas et al., 2013, 2015). This algorithm provides a solution for the unsupervised discovery of the periodic patterns (the habits) in a transactional database. xED uses the formalisms proposed in chapter 2 for the characterization of the periodic episodes. xED is evaluated on real-life datasets and compared to another algorithm from the literature (Heierman et al., 2004).

Chapter 4 further extends xED and adapts it for the handling of data streams. The proposed algorithm (Soulas and Lenca, 2015), sxED (**s**treaming **xED**) allows the discovery of periodic patterns in a window sliding over the dataset, and the update of the discovered habits when the home automation sensors record new information. In particular, new habits are detected, and the descriptions of existing habits are updated. sxED is evaluated on real-life datasets and qualitatively compared to xED.

Chapter 5 focuses on another type of periodicity description: regularity. This periodicity model, though less robust to variability, remains however very interesting, notably because of its simplicity and scalability. We propose in this chapter TKRES (Amphawan et al., 2015), for the discovery and update of the regular episodes in a sensor event stream. The top-$k$ approach allows us to constrain the size of the results in output. TKRES is also evaluated on real-life datasets.

Chapter 6 concludes this thesis. It summarizes the main contributions of this work, and highlights opportunities for future work.

The major contributions of this work are of an algorithmic nature. Nevertheless, we have also considered ways to include the medical staff and caregivers in the monitoring process, which we describe in two appendices: appendix A presents a user-centered process mining approach, allowing the supervisor to input his or her knowledge and mining expectations and requirements. Appendix B contains some of the visual tools used to communicate about the data to people who are not data experts.

# Chapter 1

# Ambient assisted living and activity monitoring

**Chapter outline** This introductory chapter points out the motivations for this work, highlighting the socio-demographic context in which it is held (section 1.1), and the rationale behind activity monitoring (section 1.2). Section 1.3 introduced the smart homes and ambient assisted systems that are used for activity monitoring. Some prominent projects are referred to, and the used infrastructure and sensors are further detailed in section 1.4. Sections 1.5 and 1.6 review the two main trends for activity monitoring: activity classification (supervised learning) and activity pattern discovery (unsupervised learning). Finally, section 1.7 narrows down the specific context of this thesis, and summarizes its main contributions.

## 1.1 Socio-demographic context

The population in many countries is getting older. For example, it is expected that in 2025, a third of the European population will be over 60 years old (Sölvesdotter et al., 2007, Chap. 2), and in China the proportion of people older than 65 is predicted to be as high as 22.7% in 2050 (Zhang and Chen, 2006).

Meeting the needs of this aging population, which is more prone to health problems and loneliness, is thus one of the big challenges of this century.

Elderly people are indeed frailer and more prone to chronic diseases. Falls and malnutrition problems are frequently observed.

Because of the increasing mobility amongst the younger generations, many older people's families live far away, and cannot take care of their older relatives on an daily basis. Because of these threats to the elderly people's health and well-being, it becomes more dangerous for them to live alone at home, which is also a source of anxiety for them and their families. That is notably why families tend to push them to move to a nursing-home. Health is in fact the first reason why people over 75 move (Kotchera et al., 2005).

However, most elderly people would rather continue to live in their own home (Kotchera et al., 2005). In addition, solutions enabling aging at home are also usually cheaper for the society than the funding of nursing homes. In particular, they allow a better control and management of the hospital resources (Rodríguez-Verjan et al., 2013). Helping people stay in their home longer and in better conditions is thus an active effort. Traditional means to achieve this goal include the home visits of medical staff and the use of technical aids, like support bars and emergency pendants. But thanks to the development of sensor technologies, Smart Home and Ambient Assisted Living (SHAAL) systems have also gained much attention during the last decade.

## 1.2 Activities of daily living

In order to assess one's ability to live alone, it is necessary to monitor the activities occurring in the home, with a particular focus on the Activities of Daily Living (ADLs). The ADLs are activities we carry out as part of our daily life to ensure our own self-care, and are described by Katz (1983). The classical ADLs (bathing, dressing, toileting, transferring, feeding) are often supplemented by instrumental activities of daily living, such as using the phone, shopping, preparing food, housekeeping, making laundry, taking responsibility for own medication, handling finances, etc.). Being able to perform such activities is a necessary condition for living independently, and so for aging at home.

These activities are thus the activities that are assessed in order to determine the degree of autonomy of a person. The autonomy level is measured thanks to different scales. Law and Letts (1989) propose a critical review of the

existing scales. The assessment is traditionally performed through a series of tests and interviews in controlled environments by the physician. Autonomy assessments thus occurs rarely, and could arguably benefit from day-to-day activity monitoring.

The sensors in SHAAL systems allow such monitoring. Moreover, the gathered information helps with the detection of anomalies and potential threats for the health, comfort or happiness of the elderly. This has lead to the rapid development of SHAAL systems for the evaluation and monitoring of the activities of daily living. Next section presents some prominent work using smart homes for ambient assisted living.

## 1.3 Smart homes and ambient assisted living

The sensors disseminated in the home collect information on the interactions between the inhabitants and their environment. The activities occurring in the house impact the measures: for example, a person preparing dinner might trigger a motion detector in the kitchen, a door switch on the fridge door or on a cupboard. An RFID tag on a pill-box could also register events related to medication. The sensors thus capture data describing health-related events as well as the ADLs.

Over the last couple decades, many smart home projects have thus been proposed, see for example the surveys by Silva et al. (2012); Rashidi and Mihailidis (2013); Acampora et al. (2013). One may for instance pay a particular attention to the historical projects AwareHome (Kidd et al., 1999) or MavHome (Cook et al., 2003). Ongoing projects like CASAS (Cook, 2012), DOMUS (Pigot, 2010), or the Gator Tech Smart House (Helal and Chen, 2009) keep raising new challenges and propose solutions for the improvement of aging at home. More and more instrumented labs and smart homes are being set up across the world.

The monitoring also allows the detection of anomalies or dangerous situations, like falls, and reassures distant families. Regardless of the end goal, the monitoring systems focus primarily on one of two tasks: the recognition of the ADLs (classification task), or the discovery of activity patterns (via clustering, time-series analysis, frequent pattern mining). The results may then used for prediction. Kim et al. (2010) introduce and explain the main challenges for activity monitoring, and insist in particular on the differences

between activity recognition (supervised learning) and activity pattern discovery (unsupervised learning).

## 1.4 Sensors

The sensors can be installed in the house, or worn by the user. In either cases, the activity is not directly observed, but merely the effects it has on the environment and the sensing devices. The underlying activity can however be estimated from the sensor readings. For example, if a person generates movement in the kitchen, and uses the fridge, it seems sound to assume that the person is going to eat (but could also be putting away the groceries, or cleaning the fridge).

The sensors in the house do not require the users to alter their behaviors: they just carry out their everyday activities, as if the sensors were not in their environment. However, such sensors do not discriminate the inhabitants: the effects of the activities on the environment are recorded regardless of who triggered the sensors (including visitors and pets). Initial attempts at handling this situation have been proposed, for example by Wilson and Atkeson (2005), who recognize and track the different inhabitants of a house.

The sensors in the home may be simple and fairly cheap sensors, like motion detectors, opening sensors on the doors, windows and cabinets, pressure mats on the chairs or in the bed, RFID tags, sensors measuring ambiance (temperature, luminosity, humidity), etc. These sensors can be installed without major renovation work, since they can be wireless and battery powered. They can also be already present in the house, if the person already had a home automation system (for example to control the heating system, reduce the energy consumption, improve comfort or ensure security with an alarm system). Each of these sensors provides a simple information (where the person is, what object is used), which can often be used directly, without preprocessing or feature extraction step. Thanks to the fairly low-level information they record, these sensors are not considered intrusive. More complex knowledge is built thanks to the multiplication of sensors and knowledge discovery algorithms. Projects like CASAS (Cook et al., 2013a), MITes (Tapia et al., 2004) or van Kasteren et al. (2010a) focus exclusively on this type of sensors.

More complicated sensors are also used, such as cameras (Pusiol et al., 2011), microphones (Fleury et al., 2010), electricity consumption (Patel et al., 2007)

or water flow monitoring devices. The information provided by each sensor is richer, but the sensors are also more intrusive. This raises privacy and user-acceptance issues. Their processing is more costly, and requires dedicated data preprocessing. Such sensors are often more expensive as well.

Previously existent home automation actuators can also be used as sensors: indeed, reasoning on the interactions between the user and the home automation functionalities may allow the localization of the elderly, and the understanding of their intentions (Truong et al., 2009; Allègre et al., 2012). For example, if a user asks to turn on the light in the kitchen, then it means that: (i) the user is in or intends to go to the kitchen, (ii) it is too dark in the kitchen.

The monitored users may also wear sensors or their body (accelerometers, RFID tag reader, physiological sensors) or on their clothes (smart fabric). Accelerometers have for example proven to be very efficient for the recognition of body postures and falls (Luštrek et al., 2012). Usually, the sensor recordings, directly linked to the activities of the monitored user, are sent to a smartphone or PDA, which in turn sends them to a computing unit in the monitoring system. Body-worn sensors allow a user-centered monitoring, including outside of the home. However, body-worn sensors are also very intrusive, and the monitored elderly must remember to carry the sensors and communication devices. Moreover, sensors cannot be worn all day long, and dangerous situations, like the falls occurring while bathing, or while going to the bathroom at night remain problematic.

Combinations of the different strategies have also been investigated. For example, Stikic et al. (2008) combine RFID tags with accelerometers. The combined use of the two types of sensors greatly improve the quality of the activity recognition. But due to intrusiveness and privacy issues, there is a trend toward fewer, more discrete sensors. Chernbumroong et al. (2013) consider the use of a sport / smart watch, combining an accelerometer, a temperature sensor, an altimeter and a gyroscope in a simple object. The watch allows nevertheless good activity classification.

The wide range of available sensors makes it possible to monitor every activity of daily living. However, the sensors may also be intrusive and expensive. Their data may be heterogeneous and noisy. The multiplication of sensors however allows us to consolidate the information and extract reliable knowledge from the raw data.

The next sections introduce the main learning tasks for SHAAL systems: activity recognition (section 1.5), and unsupervised activity analysis (section 1.6).

## 1.5 Activity recognition

Activity recognition is a classification task: based on the sensor reading, the task is to find the corresponding activity label. Research has been very prolific on the topic, and tenths of relevant articles have been published over the last decade. For clarity only the main trends are cited here, illustrated with a couple of relevant articles.

The classical machine learning tools have been investigated. Static and dynamic probabilistic methods have been extensively used, thanks to their good handling of uncertainty and noisy data. Static methods include Naive Bayes (Tapia et al., 2004; Stikic et al., 2008), decision trees and random forests, SVM (Fogarty et al., 2006; Fleury et al., 2010), or multiple binary classifiers (Stikic et al., 2008). Luštrek and Kaluza (2009) compare eight algorithms (C4.5 decision trees, RIPPER decision rules, Naive Bayes, kNN, SVM, random forests, bagging and boosting) for the classification of body postures (falling, lying down, standing, etc.). SVM classifiers appeared to be the most accurate classifiers, including in the presence of noisy data.

Dynamic methods, such as Hidden Markov models (Stikic et al., 2008; Chen et al., 2005), dynamic Bayesian Networks (Philipose et al., 2004), conditional random fields (Nazerfard et al., 2010a) also consider the transitions between the activities. van Kasteren et al. (2010a) compare dynamic models. Conditional Random Fields and Hidden Markov Models both allow a good activity classification. The authors moreover notice that the actual value of the sensors is in general less relevant than the fact that their value changes.

Activities have also been learned and recognized thanks to neural networks (Kim et al., 2015) and logical reasoning based on ontologies (Allègre et al., 2012; Aloulou et al., 2014). (Roy et al., 2011) propose an hybrid methods, involving both logical and probabilistic recognition rules for the plan recognition of Alzheimer patients, and the detection of errors in the plan realization.

Table 1.1 summarizes the main contributions to activity recognition.

Table 1.1 – Summary of activity recognition

| | Goals | Sensors | Algorithms |
|---|---|---|---|
| Tapia et al. (2004) | ADL classification | State-change sensors, taped on objects | Naive bayesian classifiers |
| Philipose et al. (2004) | ADL classification | RFID tags and glove | Dynamic Bayesian networks |
| Chen et al. (2005) | Bathroom activity monitoring | Microphone in the bathroom | Classification with HMM |
| Fogarty et al. (2006) | Water-related activity classification | Microphones listening to the water-flow | SVM |
| Stikic et al. (2008) | ADL classification | RFID tags, accelerometers | Naive Bayes, HMM, Joint Boosting |
| Gu et al. (2009) | Interleaved and concurrent activity classification | RFID, Accelerometers | Emerging patterns |
| Luštrek and Kaluza (2009) | Body posture recognition, fall detection | Accelerometers | C4.5, decision rules, Naive Bayes, kNN, SVM, random forests, bagging, boosting |
| Fleury et al. (2010) | ADL classification | Switch, presence and ambiance sensors, accelerometers, microphones | SVM |
| van Kasteren et al. (2010a) | ADL classification | Contact switches, motion detectors, pressure mats | HMM, CRF |
| Nazerfard et al. (2010a) | ADL classification | Motion sensors, temperature | Conditional random fields |
| Roy et al. (2011) | Plan recognition, anomaly detection | Switch, motion and pressure sensors, RFID, accelerometers | Hybrid logical / probabilistic algorithm |
| Dawadi et al. (2015) | Health and well-being assessment | Motion detectors, ambiance sensors, contact switches | time-series |
| Kim et al. (2015) | Uncertainty assessment in ADL classification | Accelerometers on objects, touch sensors | Multi-layered neural networks |

**Getting annotated data** A very large proportion of the aforementioned algorithms rely on supervised learning. They require annotated data, which is hard to get in the context of SHAAL systems (Cleland et al., 2013): two main strategies are currently used. In the first one, the inhabitant is asked to write down the name of every activity and the time when it was undertaken. In the second one, cameras are installed in the home, and someone later watches the video footprints and annotates the activities. Either method is very demanding and time-consuming. They are also intrusive into the user's daily life, and prone to annotation errors.

Moreover, the set activities that may occur in the home is usually much wider than the available annotations. This leads to an increasing interest for unsupervised algorithms for the discovery of frequent and periodic activities (Cook et al., 2013b).

## 1.6 Unsupervised activity analysis

Unsupervised activity discovery has used methods from other domain. Since the sensors generate sequences of events, sensor logs have sometimes been seen as documents containing words (the sensor values). Methods from the natural language processing field have thus been adapted to this context.

Hamid et al. (2009) uses $n$-grams to represent the activities: they search activities as a sequence of events, where an event is a key-object interaction. The activities are encoded based on their contiguous event subsequences: a window of fixed size $n$ is slided over the instances of the activities. The $n$-grams contained in the successive window are listed and counted, generating an histogram of $n$-grams for each instance. These encodings are then compared thanks to a similarity measure to extract activity classes. Activities that deviate from the characteristics of the activity classes can then be detected.

Huynh et al. (2008) exploit topic models. Activity patterns (low-level descriptions, linked with posture and used objects) are inferred from the sensors, and can be recognized thanks to traditional (supervised) activity recognition algorithms. However, Huynh et al. argue that higher-level descriptions (working in the office, commuting, having dinner) are not fully characterized by the objects and postures they involve. They thus describe each instant as a probabilistic combination of topics, these topics being strongly linked to

the users routine and learned from the data.

Zheng et al. (2008) successfully apply growing self-organizing maps to cluster the ADLs in a house equipped with RFID-like sensors monitoring object usage. The control of the growing rate of the network allows them to build hierarchical clusters. Indeed, a coarse clustering (little network growth) allows them to detect areas of interest, which are then further analyzed using a finer-grained clustering.

Jakkula et al. (2009) studies the temporal relationships between the activities: they first mine the frequent sequences using an APriori-like algorithm. Then, the temporal relationships between the events in the sequence are searched (if A and B are two activities, the temporal relations could be: A starts before B, A overlaps B, A starts at the same time than B, etc.). When a new event is recorded, it is assessed in order to determine whether it is an anomaly, based on its occurrence probability, derived from the frequent temporal relationships and the context.

Li and Dustdar (2011) propose a semi-supervised classification framework for high-dimensional heterogeneous data: an unsupervised subspace clustering on the training data enables the discovery of activity classes (the clusters). Each cluster is furthermore associated with a relevant subset of the dimensions characterizing the cluster. The clusters are then labeled by an expert, which provides the ground truth for a traditional classification algorithm. Thanks to the data projection on a reduced set of dimensions, the expert labeling is made easier, which reduces the labeling errors. Moreover, the unsupervised subspace clustering allows the discovery of relevant, but unnoticed activities, and a feature selection adapted to each target activity.

Rodner and Litz (2013) use association rules: the recorded events are preprocessed to generate transactions of items. This preprocessing uses data extraction, aggregation and transformation in order to enrich the raw events with geographic data; the numeric data and dates are discretized, etc. The transactions are then mined within the association rules framework to extract frequent and confident rules. This approach relies a lot on the expressiveness of the transactions, and hence depends on the richness of the preprocessing.

Truong et al. (2009) aim to increase the autonomy of disabled people by simplifying their interactions with a smart environment. They thus use the home automation control logs to identify relevant scenarios: the home automation commands (the services) are clustered into scenarios, thanks to

self-organizing maps. Services that are often used in succession, or within a short time-span are more likely to be grouped together. The discovered services allow the control of the devices with fewer interactions, and in a manner that is fully adapted to the habits of the user.

Nazerfard et al. (2010b) study the occurrence patterns of the activities of daily living: for each activity, the start time of the occurrences are gathered, and clustered using K-Means. For each cluster, the mean start time and standard deviation are computed, which produces for each activity a mixture model representing the start time. A similar representation is produced for the activity durations. This allows a description of the normal occurrences of the activities with temporal features. Temporal association rules are then mined from these features, in order to discover for each cluster of each activity, which activities are most likely to follow. Though this analysis is unsupervised, it requires activity data (instead of raw sensor data): indeed, it needs to know when the activities occur, and how long they last. The activities are analyzed, but not discovered.

Rashidi et al. (2011) propose ADM (Activity Discovery Method) for the unsupervised discovery of activity models. They first discover frequent discontinuous sequences as well as their frequent variations. These interesting sequences are then clustered using K-Means, in order to group the observations into activity definitions. The results of ADM are then learned to recognize activity models, thanks to Hidden Markov models. This approach was also extended (Rashidi and Cook, 2010) to address the problems of stream mining and varying event densities in different areas in the home.

Avci and Passerini (2013) split the sequence of sensor records thanks to partitioning algorithms based on the geographic distance between the sensors in the home, or based on the context (the recently observed events). The segments are then clustered, and both the segments and clusters are used to train a Hidden Semi-Markov Model. The frameworks allows the unsupervised classification of the activities, even without knowing the kind or number of activities in the dataset. However, interleaved activities and idle times still raise issues for the segmentation and clustering of the activities.

Another approach, Episode Discovery (ED, extensively described in section 3.3) (Heierman et al., 2004) searches for interesting habits. The interest of the patterns is assessed on a combination of measures, including their length, frequency and periodicity.

Table 1.2 summarizes the characteristics of the aforementioned unsupervised approaches. In particular, one can notice that the announced objectives are quite different, and thus the shape of the searched patterns vary as well.

## 1.7 Positioning

The proposals of this thesis contribute to the unsupervised activity analysis scope, and focus primarily on habit discovery and monitoring. The formalisms used for habit mining are presented in the next chapter, followed by our three proposals for the discovery of habits:

- xED (Soulas et al., 2013, 2015) is described in chapter 3. It proposes a strategy for the discovery of periodic episodes in a sensor event database, and is built upon the proposal of Heierman et al. (2004);
- sxED (Soulas and Lenca, 2015), detailed in chapter 4, extends xED to the context of event streams;
- TKRES (Amphawan et al., 2015) is presented in chapter 5. It proposes a framework for the discovery of regular patterns in event streams.

The algorithms are designed to handle events recorded by simple sensors disseminated in the home, such as motion detectors, contact switch sensors, or RFID tags. Indeed, these sensors are not intrusive, do not require any maintenance on behalf of the inhabitant, and are usually cheaper than the more complex sensors.

Table 1.2 – Summary of the characteristics of unsupervised approaches for activity monitoring in SHAAL systems

| | **Goals** | **Sensors** | **Algorithms** |
|---|---|---|---|
| Heierman et al. (2004) | Periodic behavior discovery | Synthetic data | Event set generation, periodicity analysis |
| Huynh et al. (2008) | Unsupervised classification | Accelerometers | topic models |
| Zheng et al. (2008) | Activity hierarchical clustering | RFID | Growing Self-Organizing maps |
| Hamid et al. (2009) | Automated activity discovery | Camera | Suffix trees |
| Jakkula et al. (2009) | Anomaly detection | Motion detectors, temperature, light, humidity sensors | APriori Sequence mining, temporal relationship analysis |
| Truong et al. (2009) | Scenario identification | Home automation commands | Reinforced graphs, SOM |
| Nazerfard et al. (2010b) | Anomaly detection | Motion detectors, door/cabinet sensors | $k$-Means clustering, temporal association rule mining |
| Li and Dustdar (2011) | Activity class discovery | $\emptyset$, the paper is a proof of concept | Subspace clustering |
| Rashidi et al. (2011) | Unsupervised activity recognition | Motion detectors, contact switches | Sequence mining, K-Means clustering, HMM |
| Avci and Passerini (2013) | Unsupervised activity recognition | Motion detectors, contact switches | Sequence partitioning, clustering, HSMM |
| Rodner and Litz (2013) | Behavior modeling, Anomaly detection | Motion detectors, contact switches, water meter, switches for light and roller blinds | FP-Growth, Association rule mining |

# Chapter 2

# Habit monitoring

## 2.1 Introduction

Every individual tends to follow routines. These habits are exacerbated for elderly people, as pointed out by Bergua et al. (2013). The authors argue that such habits help the elderly people maintain control over the course of their day and over their environment. Habits reduce the risks of experiencing unexpected and unpleasant situations, and thus improve their well-being. Habits and routines thus play an important role for the maintenance of autonomy.

Moreover, changes in the routines may sometimes indicate that carrying out everyday activities is becoming harder, or even account for the onset of a disorder, like Alzheimer's disease. For example, the symptoms of dementia include increasing apathy or disorientation (Benoit et al., 2005). Such symptoms reflect on activity completion and habits. The study of habits and routines thus is as an important aspect of activity monitoring.

The objective of this chapter is to characterize what habits look like, and how they can be discovered in sensor data. A new formalism for the description of periodic behaviors is proposed.

**Chapter outline** The remainder of this chapter is organized as follows: section 2.2 focuses on the characterization of the habits. The properties of a useful habit discovery algorithm are also listed. Section 2.3 presents the

Table 2.1 – Examples of habits, expressed in a natural language

| Id | Condition | Action |
|---|---|---|
| 1 | When time is around 8:00 | the user wakes up |
| 2 | When lunch is ready | the user eats it |
| 3 | When it rains | the user takes an umbrella |
| 4 | On Tuesdays or Saturdays | the user goes to the market |

main trends that have been explored in the literature to monitor habits and periodic behaviors. We propose a periodicity framework in section 2.4 (Soulas et al., 2013), that is then used throughout the remainder of the thesis.

## 2.2 Habit characterization and discovery

Habits are behaviors each individual tends to follow given a particular context. The context may be temporal (time, day, season), a personal feeling (hunger, thirst, tiredness), a situation or an event that just occurred (it started raining, mail was delivered, etc.). The behavior the user puts into action to react to the condition is personal: two individuals are unlikely to share the same habits, react in the same way, or share the same daily organization. Habits can usually be described using a natural language thanks to sentences such as:

"When `<condition>`, the person usually `<does something>`"

The `<condition>` can also be a combination of conditions. Table 2.1 provides some examples of such habits.

In the context of ambient assisted living, we monitor the ADLs, and habits related to the ADLs are of particular interest, since they characterize normal behaviors and landmarks to assess safety and well-being. Deviations from usual habits and missing regular behaviors can hint a problem.

Habits triggered by an event and habits triggered by a temporal context call for different tools to formalize and detect habits. For example, usual reactions to an event that just occurred or a feeling (examples 2 and 3 in table 2.1) are usually detected thanks to association rules (Rodner and Litz,

Table 2.2 – Example of periodic habits

| Id | Period | Relative timestamp | Action |
|---|---|---|---|
| 1 | Every day | when time is around 8:00 | the user wakes up |
| 4 | Every week | on Tuesday or Saturday | the user goes to the market |

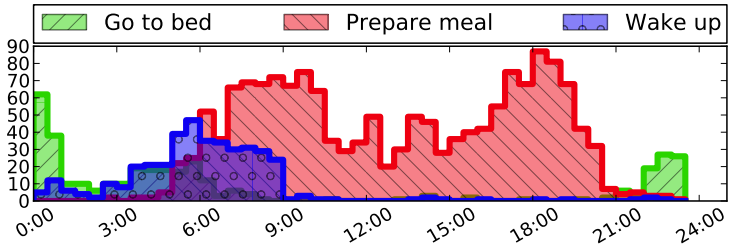



Figure 2.1 – Example of an histogram representing the observed occurrence times (start time) of three typical ALDs: eating, waking up, going to bed. These activities are extracted from the CASAS Aruba dataset[1]

2013) and temporal association rules to exhibit causality (Nazerfard et al., 2010b), or thanks to frequent sequences (Rashidi and Cook, 2010).

In this thesis, we extend these approaches by focusing on habits triggered by temporal contexts (time, date, such as examples 1 and 4 in table 2.1). Such habits can also be expressed as:

> "Every `<period>`, around `<time position within the period>`, the user usually `<does something>`"

Some of the habits described in table 2.1 can be reformulated with this (`<period>`, `<time position>`) formalism (table 2.2). We focus here on the detection of this kind of behaviors.

The rhythm at which activities tend to occur (i.e. the period) varies from one activity to the other. For example, waking up and meals occur daily (even several times a day for the meals), shopping is weekly, visits to the physician may occur once or twice a month, etc.

[1] http://ailab.wsu.edu/casas/datasets/aruba.zip, this dataset is more precisely described in chapter 3

Figure 2.1 presents an example histogram of the occurrence times for three typical ADLs: waking up, eating, and going to bed. It highlights some characteristics of the habits:

- There may be more than one cluster of occurrences within the period of interest (here, two meals at home a day: a breakfast around 8:00 and a dinner around 18:00). These clusters of occurrences form the components of a habit;
- Each component has a preferential occurrence time (mean) and its own variability (standard deviation). For example in the example figure 2.1, the occurrence times vary less for dinner than for breakfast. The variability for going to bed is also lower than the variability of the waking up activity.
- Some occurrences happen outside of the preferential occurrence intervals: some occurrences do not follow the periodic trend. For example, there are some "wake up" activities in the afternoon (probably occurring after a nap) or during the night (for example to go to the bathroom).

**Routine characterization** Habits are thus personal and contextualized. Even if they tend to follow strict patterns, habits are carried out by human beings, not machines, and are thus prone to variability, which is activity-dependent. Some activities, such as taking medication, should for instance have a very low variance. But there may not be an underlying meaning to a relatively high variance when studying bedtime, and this variability can for instance be linked to the broadcasting of an interesting film on television, or leisure-related activities.

We thus argue that a good habit discovery algorithm should thus learn habits for each user, and be able to adapt to the variability inherent to human lifestyle. We propose to search the habits in the form of periodic patterns.

## 2.3 Periodicity analysis

The discovery of patterns based on the temporal regularities in their occurrences is of great interest in a wide range of applications, such as genetic data analysis (Glynn et al., 2006), social interactions analysis (Lahiri and

Berger-Wolf, 2008), transactional database mining (Kiran and Reddy, 2010), biological sustainability studies (Li et al., 2012), mobility (Baratchi et al., 2013), etc., and of course ambient assisted living.

In order to find which patterns are periodic, the rhythm of the patterns appearances is studied. There is however no standard, universally-accepted definition for periodicity. Possible definitions can be classified into basic categories:

**Regularity** It considers the maximal time gap between two consecutive occurrences (Tanbeer et al., 2009; Amphawan et al., 2011). This simple measure was originally defined to assess the regularity of itemsets in transactional datasets. This measure is further exploited in TKRES (Amphawan et al., 2015), the contribution presented in chapter 5. Regularity allows the description of behaviors in the form:

> "The user `<does something>` at least once every `<time duration>`"

**Cycles of intervals** The objective is to discover whether the time gaps between consecutive occurrences form a repeating cycle. This is the definition used by Episode Discovery (Heierman et al., 2004), described in details in the next chapter (section 3.4). It is also used by Lahiri and Berger-Wolf (2008), with cycles of length 1. The time gaps describe habits in the form:

> "The user `<does something>`, then does it again after `<gap 1>` has passed, then after `<gap 2>`, etc., then `<gap` $n$`>`, then again after `<gap 1>`, etc."

**High probability behaviors** The patterns are searched in the form:

> "If time is around `<time>` then the monitored subject is likely to `<do something>`"

In particular, this kind of approach is used by Nazerfard et al. (2010b); Li et al. (2012); Baratchi et al. (2013). It is also the one we use for xED (Soulas et al., 2013, 2015) (detailed in chapter 3) and sxED (Soulas and Lenca, 2015) (see chapter 4).

Table 2.3 – Overview of different visions of periodicity

| Algorithm | Raw data | Searched patterns | Periodicity definition | Impact if an occurrence is missing | Impact if an occurrence is extra |
|---|---|---|---|---|---|
| ED (Heierman et al., 2004) | Event sequence | Episode | Repeating time intervals | high | high |
| Lahiri and Berger-Wolf (2008) | Dynamic graph | Subgraphs | Repeating time interval | high | high |
| Tanbeer et al. (2009) | Itemset sequence | Itemsets | Regularity | high | **low** |
| Kiran and Reddy (2010) | Itemset sequence | Itemsets | Regularity | high | **low** |
| Nazerfard et al. (2010b) | Activity log | Temporal association rules | Rule confidence | **low** | **low** |
| Li et al. (2012) | GPS track | Location | Time-wise location probability | **low** | **low** |
| Baratchi et al. (2013) | GPS track | Location | Time-wise location probability | **low** | **low** |
| xED (Soulas et al., 2013, 2015) | Event sequence | Episode | GMM | **low** | **low** |
| sxED (Soulas and Lenca, 2015) | Event sequence | Episode | GMM | **low** | **low** |
| TKRES (Amphawan et al., 2015) | Event sequence | Episode | Regularity | high | **low** |

**Discussion** Table 2.3 summarizes the above-mentioned algorithms and their characteristics. Their ability to handle variability is also assessed: in order to perform well, the produced models should not be modified much if an unexpected occurrence of the habit is recorded, or if an expected occurrence is missing. The three main contributions of this thesis are positioned with regards to the other periodic behavior discovery algorithms.

Most of these algorithms do not handle well unexpected behaviors. In particular, missing occurrences may greatly impact regularity measures and repeating cycles of time intervals. The probabilistic habit discovery algorithms are much more suitable: the periodicity models are not impacted much by the

presence of a few outliers. However, the methods presented in (Li et al., 2012) and (Baratchi et al., 2013) focus on the prediction of the position (GPS coordinates), not the discovery of activities. The periodicity model (described in section 2.4) used in xED and sxED allows the discovery of habits and the characterization of their usual variability. TKRES uses however regularity, and is thus a bit more sensible to missing occurrences of expected patterns.

## 2.4 Periodicity as a Gaussian mixture model

We argue that a good way to apprehend periodicity is to describe the clusters of occurrences, such as they are represented in figure 2.1, which we propose to do thanks to Gaussian Mixture Models (GMM). Gaussian Mixture Models are frequently used to model the behaviors of random variables. We apply them here to periodicity description. This description covers the desirable characteristics of a periodicity description for the characterization of behaviors emanating from living species: in particular, it is not perturbed by missing occurrences, and shows robustness with regard to the presence of shifted or extra occurrences (see the coming paragraphs for more detail). It also characterizes how and how often these unexpected behaviors occur. The next paragraphs describe how the mixtures models are used in xED and sxED.

**Relative timestamps.** For a period $T$ (e.g. one day, one week, one month), and a timestamp $t$, the relative timestamp $t_r$ refers to the position of $t$ within the period: $t_r = t \mod T$. For example, with a period of one day, the relative timestamp corresponding to `2015-11-24 10:21:00` is the time: `10:21:00`. With a period of one week, it is the combination of the day within the week and time: `Tuesday, 10:21:00`. The periodicity description with GMMs aims at the description of the distribution of the relative timestamps for a given period $T$.

**Gaussian Mixture Models** In this formalism, the periodicity (for a period $T$) of each habit $E$ is described with a Gaussian Mixture Model $M_E^T$, that is to say a list of components, described by their means and standard deviations:

$$M_E^T = \{(\mu_1, \sigma_1), \ldots, (\mu_m, \sigma_m)\}$$

The GMM models the usual relative occurrence timestamps of the habit. The means $\mu_1, \ldots, \mu_n$ of the components describe the relative timestamps around which occurrences are expected to occur,and take values with the range $[0, T]$. The standard deviations $\sigma_1, \ldots, \sigma_n$ describe how much the actual relative timestamps usually differ from the expected timestamps.

According to our characterization of the occurrence behavior of an habit $M_E^T$, we expect $m$ occurrences of $E$ to occur during each period, once around each mean $\mu_i$ in $M_E^T$. More formally, we define the expected occurrences of an episode $E$ as follows:

**Definition 2.1** (Expected occurrence)**.** For every $i \in [1, m]$ and every integer $n \geq 0$, let $t_{n,i} = n \cdot T + \mu_i$ ($t_{n,i}$ is a timestamp, $\mu_i$ a relative timestamp). An occurrence of the episode is *expected* to start around each time $t_{n,i}$ ($\pm a \cdot \sigma_i$), that is to say that an occurrence is expected to start in the time interval $[t_{n,i} - a \cdot \sigma_i, t_{n,i} + a \cdot \sigma_i]$, where $a$ is a user-defined parameter. Using the natural habit description used in the explanation for high probability behaviors, we could also say:

> "If time is `<around` $t_{n,i}$`>`, then the monitored subject is expected to `<do` $E$`>`"

If several occurrences happen around the same time $t_{n,i}$, the one closest to $t_{n,i}$ is considered as the expected occurrence. The others are extra occurrences, as well as all the occurrences that are recorded outside of the expected intervals.

The parameter $a$ controls the proportion of the occurrences that are considered as normal. If the time distribution of an habit indeed follows a Gaussian distribution, and if the mean and standard deviation of this distribution are correctly estimated, a value $a = 2$ will classify an average of 95% of the occurrences as *occurring as expected*. In the experiments in subsequent chapters, $a = 2$.

With this representation, each episode is attached to its own list of (mean time, standard deviation) pairs: the proposed periodicity characterization allows the construction of models that are highly adapted to each episode and to the user.

Figure 2.2 represents the occurrences of an event `kitchen` over the course of four days. The event occurs twice a day, once in the morning around 8:03 with a standard deviation of 13 minutes, and then again around 12:40, with

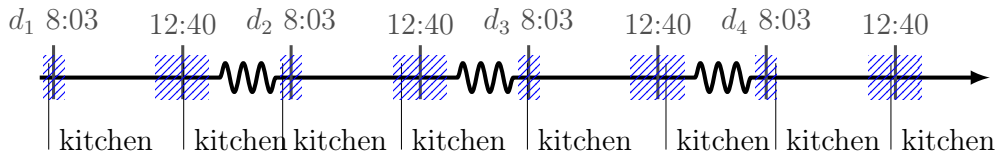



Figure 2.2 – Periodicity for episode {kitchen}

a standard deviation of 1 hour and 10 minutes. A periodicity description for the `kitchen` event, with a period $T$ =1 day is thus: {(8:03, 0:13), (12:40, 1:10)}.

The quality of the descriptions is evaluated on their accuracy (definition 2.2).

**Definition 2.2** (Periodicity accuracy)**.** The *accuracy* of a periodicity description is the proportion of the occurrences expected to occur that were indeed observed.

**Definition 2.3** (Periodic episode)**.** An episode $E$ is *periodic* on $\Delta t$ if its accuracy is higher than an accuracy threshold $A_{min}$.

Depending on the considered period $T$, each episode can have several periodicities. For instance, a "wake up" habit occurring on week days around seven (with standard deviation $\sigma$), can be seen as a daily or weekly habit:

- Daily habit ($T$ = 1day): its periodicity is $\{(7{:}00, \sigma)\}$, with expected but missing occurrences on Saturdays and Sundays (accuracy: $5/7 = 71\%$),
- Weekly habit ($T$ = 1week): its periodicity is {(Mon 7:00, $\sigma$), (Tue 7:00, $\sigma$), (Wed 7:00, $\sigma$), (Thu 7:00, $\sigma$), (Fri 7:00, $\sigma$)}. Since it occurs as expected on the five week day, it has a 100% accuracy. This second description is longer, but also more accurate.

## 2.5 Conclusion

Habits are instinctively seen as periodic patterns. The human life means variability. However, there is currently no consensus on the definition of periodicity, or on methods to characterize it. Most existing strategies do not manage well the mining task when the behaviors do not follow exactly the expected periodicity.

We thus propose a first contribution for this thesis: a periodicity description based on mixture models, which we evaluate based on its accuracy. In the subsequent chapters, we propose several strategies to discover patterns that can be pertinently characterized with such periodicity descriptions, and update them when habits evolve.

# Chapter 3

# Periodic episode discovery in static databases

## 3.1 Introduction

In this chapter, the information registered by the sensors is used as a data source for the discovery of interesting patterns describing the habits, i.e. the discovery of periodic patterns, such as they were described in the previous chapter. We propose e**x**tended **E**pisode **D**iscovery, or xED for short (Soulas et al., 2013, 2015), an unsupervised algorithm for the discovery of frequent and periodic episodes over an event database. Being unsupervized, xED automatically adapts to different users, and different periodicity patterns. xED also describes the times when an expected behavior does not occur, which is useful for anomaly detection.

**Chapter outline** Section 3.2 presents the activity patterns xED looks for. Section 3.3 presents a related algorithm: the Episode Discovery algorithm (Heierman et al., 2004), which looks for periodic patterns as well. As shown in table 2.3, Episode Discovery is the closest algorithm to our habit monitoring target, and has served as inspiration for xED. The differences with our approach are highlighted. Our contribution, xED is described in section 3.4. Its different components are detailed and analyzed. Section 3.5 presents experiments on six real-life datasets, as well as a comparison of the performances of xED with Episode Discovery. Conclusions and perspectives

are proposed in section 3.6.

## 3.2 Habits and episodes

This section presents the definitions used for the mining of periodic episodes. The processed data is an *event* sequence, generated by the user activity and recorded by the sensors disseminated in the smart home. Each event is characterized with a *timestamp* and a *label*. The event label is usually composed of an identifier characterizing the triggered sensor and its value. They can also be obtained from a preprocessing algorithm, for instance an activity classifier, such as those of van Kasteren et al. (2010b).

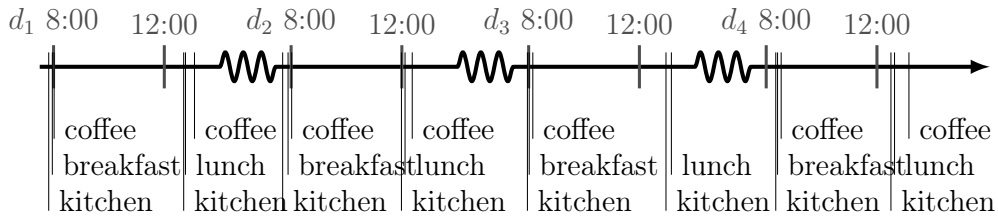
The periodic behavior, i.e. the action the user carries out periodically is described as an *episode*, that is to say a collection of events labels (definition 3.1). The periodicity is then evaluated on the timestamps of their *occurrences* (definition 4.2). The assumption behind the use of episodes is that activities and behaviors can be at least partially characterized by the sensors they trigger (and thus the events they generate). Each sensor records just a simple piece of information (a location, the use of an appliance or a resource), and is thus not accurate enough to discriminate activities, but the combinations of sensors (the episodes) better tackle this task.

**Definition 3.1** (Episode)**.** An episode $E$ is a *set of event labels* $\{e_1, ..., e_n\}$.

The label order in the episode is thus not taken into account, and duplicate labels are considered only once. Indeed, sensors like motion detectors or RFID tag readers tend to generate bursts of events while the person moves, or is located close the tagged object. However, the number of events in the bursts is not relevant in this context. The relative order of the sensor activations is not always relevant either: it is greatly impacted by sensor sensibility or data transmission delays.

**Definition 3.2** (Episode Occurrence)**.** Let $E = \{e_1, ..., e_n\}$ be an episode. There is an occurrence $o$ of $E$ at time $t_1$ if there exists a permutation $p$ of $(1, ..., n)$ and $n$ timestamps $t_1 \leq ... \leq t_n$ such that $o = \langle (t_1, e_{p(1)}), ..., (t_n, e_{p(n)}) \rangle$ is a subsequence of the events in the dataset. $t_n - t_1$ is the *duration* of $o$ (we can also say that $o$ *lasts for* $t_n - t_1$).

This definition for episode occurrence allows the recognition of an episode even if it is interleaved with other events (e.g., an activity is temporarily




$d_1$ `7:51 kitchen;` $d_1$ `7:59 breakfast;` $d_1$ `8:02 coffee;` $d_1$ `12:42 kitchen;`
$d_1$ `12:46 lunch;` $d_1$ `13:06 coffee;` $d_2$ `7:43 kitchen; ...`

Figure 3.1 – Toy example dataset

interrupted to do another one, and is resumed later): the recognition is thus more robust. If two activities are frequently interleaved, then it is likely that the episode of interest is the combination of both activities: for example, if the user always listens to the news while preparing breakfast, then listening to the news becomes a component of the breakfast routine.

The duration of the episode occurrences can be constrained with an upper bound, the *maximal episode duration* parameter $T_{ep}$. Occurrences that last for a longer duration are not taken into account. The assumption behind $T_{ep}$ is that two events that are close in time are likely triggered by the same human activity. They should thus be considered as a pair. On the contrary, events that are distant in time are likely unrelated. A reasonable value is $T_{ep} = 30$ minutes: it corresponds to the time needed for most daily life activities (prepare a meal, take a shower, ...), according to physicians. It can also be estimated thanks to data exploration and segmentation techniques.

**Example** A toy dataset is presented figure 3.1. It is used as a running example in the current chapter. It contains 22 events spanning over four days $(d_1, d_2, d_3, d_4)$. Four event labels are stored: `kitchen`, `breakfast`, `lunch`, `coffee`. Many episodes can be extracted, such as {`kitchen`, `lunch`}, or {`breakfast`}. The episode {`kitchen`, `breakfast`, `coffee`} occurs four times under the constraint $T_{ep} = 30$ min: at 7:51 on day 1, at 7:43 on day 2, 8:01 on day 3, and 8:19 on day 4.

**Problem statement** The objective of xED is to discover the periodic episodes in the event dataset. The periodicity is described with the Gaussian Mixture Model formalism described in section 2.4, and evaluated by its accuracy.

## 3.3 Related work: Episode Discovery

To the best of our knowledge, algorithms used in SHAAL systems are not designed for the same purposes as xED, that is to say: periodic episode discovery and usual variability characterization. The closest algorithm is Episode Discovery, the main focus of which is also the periodic episode discovery (but with a different periodicity definition). This section details how Episode Discovery works, and discusses its limitations.

**Episode Discovery (ED) (Heierman et al., 2004)** is based on the *Minimum Description Length* (MDL) principle (Rissanen, 1989), which states that the best model to represent data is the one with the shortest description. Following this principle, ED replaces the occurrences of periodically reoccurring episodes by a single header, describing the episodes and when they occur. This allows the rewriting of the dataset into more compact representations. ED iteratively follows three steps:

1. *Generation of a list of candidate episodes*
2. *Periodicity analysis of each candidate.* Two periodicity descriptions are computed, with different granularities (the timestamps are truncated):
   - a fine-grained periodicity (truncation to the full hour),
   - a coarse-grained periodicity (truncation to the full day).

   In ED, the *periodicity* of an episode is based on repeating cycles of time intervals. For an episode $E$ occurring at times $t_0, t_1, ..., t_n$, the periodicity analysis goes as follows:
   - Truncation of the timestamps to the currently investigated granularity (hour or day): the resulting timestamps are $t'_0, t'_1, ..., t'_n$;
   - Computation of the intervals between the occurrences $\delta t_1, ..., \delta t_n$, with $\delta t_i = t'_i - t'_{i-1}$ for $1 \leq i \leq n$;
   - Estimation of the most likely length $l$ of the repeating cycle of intervals, thanks to an autocorrelation measure;
   - Processing of the occurrences: The first $l$ intervals $\delta t_1, ..., \delta t_l$ build up the first periodicity description. $\delta t_{l+1}$ is matched against $\delta t_1$; $\delta t_{l+2}$ against $\delta t_2$; etc., as long as the subsequent intervals match

Table 3.1 – Periodicity analysis for episode {`kitchen`} with ED

| **Occurrences** | | $d_1$ 7:51, 12:42, $d_2$ 7:43, 12:02, $d_3$ 8:01, 12:57, $d_4$ 8:19, 12:30 | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Fine** | Truncated | $d_1$ 7:00, | 12:00, | $d_2$ 7:00, | 12:00, | $d_3$ 8:00, | 12:00, | $d_4$ 8:19, | 12:00 |
| | Intervals | | 5:00, | 19:00, | 5:00, | 20:00, | 4:00, | 20:00, | 4:00 |
| | Periodicity | | (5:00, 19:00) | | | | (4:00, 20:00) | | |
| | Mistakes | | | | | $d_3$ 8:01 | | | |
| **Coarse** | Truncated | $d_1$, | $d_1$, | $d_2$, | $d_2$, | $d_3$, | $d_3$, | $d_4$, | $d_4$ |
| | Intervals | | 0:00, | 24:00, | 0:00, | 24:00, | 0:00, | 24:00, | 0:00 |
| | Periodicity | | (0:00, 24:00) | | | | | | |

the interval cycle. When the observed interval does not match the expected one (too short or too long), the displaced occurrence is marked as a mistake, and the next $l$ intervals make up a new periodicity.

3. *Dataset factorization, using the periodicity information.* The patterns allowing the smallest dataset rewriting are deemed most interesting.

ED thus includes all the occurrences of an episode in the periodicity model: every extra, shifted or missing occurrence is recorded as a mistake, and the periodicity cycles need to be recomputed. In the worst-case scenario, a periodicity description remains valid for $l$ intervals only (the $l$ intervals on which it was built). Every $l+1$ occurrence can potentially be a mistake: there can thus be up to $N_{MAX} = \lceil \frac{n+1-(l+1)}{l+1} \rceil$ such mistakes ($n+1$ is the number of occurrences for the episode, $\lceil \cdot \rceil$ denotes the ceiling function). Since there is no such measure in the original paper describing ED, we here define the *mistake rate* of an episode as the ratio between the mistake count and $N_{MAX}$. It measures how long the periodicity cycles remain valid: it should be very low for a periodic episode.

**Application on episode {`kitchen`}** Table 3.1 sums up the periodicity analysis for the {`kitchen`} episode for the toy dataset from figure 3.1. A fine-grained and a coarse-grained periodicity is computed. After the truncation of the timestamps and the computation of the interval lists, an autocorrelation measure sets the length of the interval cycles to $l = 2$. The periodicities are then described thanks to the intervals. For the fine-grain periodicity, the first two intervals 5:00 and 19:00 form the first periodicity cycle (5:00, 19:00). The third interval 5:00 matches the expected interval (the first one

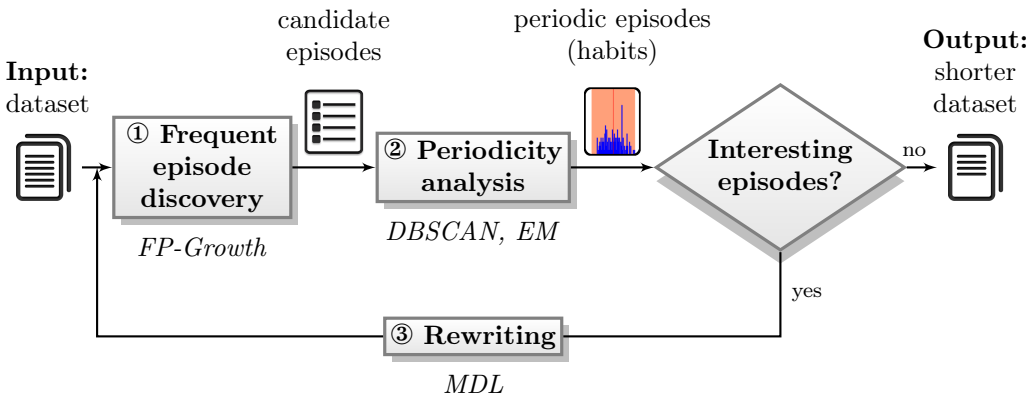



Figure 3.2 – xED: general flowchart

in the cycle), but the fourth interval (20:00) does not match the expected interval (19:00). The occurrence $d_3$ 8:01 is thus marked as a mistake, and the two following intervals form a new periodicity cycle (4:00, 20:00). No other mistake is detected until the end of the dataset. The error rate for this periodicity description is thus $\frac{1}{\lceil \frac{5}{3} \rceil} = 50\%$.

**Discussion** ED has been successfully applied in the MavHome project (Cook et al., 2003). However, it does not allow flexibility in the occurrence times of the episodes. It considers extra (unexpected) occurrences as mistakes, when they can be perfectly normal, and may thus miss some interesting episodes. The built periodicity descriptions are moreover complicated and can be overwhelming for the caregiver, when many mistakes are found.

In the next section, we present and describe the second contribution of this thesis, xED. In particular, we present the core enhancements of xED to cope with the issues of ED.

## 3.4 Extended Episode Discovery

Each iteration of xED has three steps (see also the general flowchart figure 3.2 and the overview algorithm 1):

1. The discovery of frequent episodes (section 3.4.1),

**Algorithm 1** Overview of xED algorithm: find the periodic episodes

**Input:** $data, T_{ep}, C_{ep}, S_{min}, A_{min}, SD_{max}, \Delta T_{max}$ (see table 4.2 for a detailed description of the parameters)

1: $compressed \leftarrow$ TRUE
2: **while** $compressed$ **do**
3: $compressed \leftarrow$ FALSE
4: $frequent\_episodes \leftarrow$ list of the frequent (min support: $S_{min}$) episodes (duration: $T_{ep}$)
5: $periodicities \leftarrow [\,]$
6: **for all** episode $E$ in $frequent\_episodes$ **do**
7: $(\Delta t, Descr) \leftarrow$ output from the candidate study step (algorithm 2) for $E$ $(\Delta T_{max}, S_{min}, SD_{max}, A_{min})$
8: Add $(\Delta t, Descr)$ to the interesting $periodicities$
9: Sort the descriptions by decreasing compression power (see section 3.4.3)
10: $factorized\_events \leftarrow [\,]$
11: **while** $periodicities[0]$ does not involve events in $factorized\_events$ **and** compression power $> 1$ **do**
12: Factorize $data$ by $periodicities[0]$
13: Remove $periodicities[0]$ from the list
14: Add all the factorized events to $factorized\_events$
15: $compressed \leftarrow$ TRUE /* *Another iteration is needed* */
16: **return** New, shorter description of $data$, highlighting the periodic episodes

2. The periodicity analysis of the frequent episodes (section 3.4.2), and

3. The factorization of the most interesting periodic episodes (section 3.4.3).

In order to tune xED to the particular requirements and needs of the end-user, several parameters help control the course of the algorithm, summarized in table 4.2. A description of each parameter is given, as well as its area of influence, and recommendations on the choice of its value. The parameters are also referred to in the descriptions of the steps in which they take part.

Table 3.2 – Overview of the parameters in xED

| Name | Description | Influence area | Value range * |
|---|---|---|---|
| Episode length $T_{ep}$ | Maximal time interval between events in an episode occurrence. Should correspond to the maximal duration of the ADLs. | Candidate episode construction | 0 – 24h (30 min) |
| Minimal support $S_{min}$ | Minimal number of occurrences of an episode, for that episode to be considered as frequent. If set too high, weekly patterns not found. | Candidate episode construction, DBSCAN | $> 1$ (once a week) |
| Maximal standard deviation $SD_{max}$ | Maximal standard deviation considered as normal. Should be set by a field expert (physician). | DBSCAN: controls neighborhood size | $> 0.0$ (10 % of the period) |
| Tolerance ratio $a$ | An event expected to happen at time $t$ (with standard deviation $\sigma$) occurs as expected if it occurs in the interval $[t - a \cdot \sigma, t + a \cdot \sigma]$. | Pattern quality assessment | $> 0.0$ (2) |
| Minimal accuracy $A_{min}$ | Minimal accuracy for a periodicity description to be considered as interesting, and thus factorized. | Pattern quality assessment | 0 – 100 % (50 %) |
| Long empty interval $\Delta T_{max}$ | If there is a gap $> \Delta T_{max}$ between two occurrences of an episode, the occurrences before and after the gap are split (different validity intervals). | Periodicity analysis | $> 0.0$ (3× the period) |

* the suggested / default value is given between brackets. These value depend on the application field, and are here adapted to SHAAL systems.

### 3.4.1 Candidate episode search in xED

Habits are by definition frequent behaviors. The candidate episodes are thus the *frequent episodes*: their support needs to be greater than a support threshold $S_{min}$. The early pruning of rare events and episodes also allows faster computation. However, habits are not necessarily the *most* frequent behaviors. The minimal support threshold $S_{min}$ should thus be set to a rather low value.

Looking for candidate episodes is actually the same as looking for frequent itemsets (the sets of event labels describing the episodes) in a list of transac-

tions (the events happening in an event-folding window of length $T_{ep}$). The classic frequent itemset mining algorithms can thus be used to find the frequent episodes. These algorithms are then guaranteed to find all frequent itemsets, i.e., all the frequent episodes.

We used FP-growth, since it is very scalable and efficient (Han et al., 2000). FP-growth stores the transactions in an efficient FP-tree structure. This structure contains a header table, referencing the frequent items, and a prefix tree, containing the information about the item associations in the transactions. FP-growth first builds a first instance of the FP-tree, and then mines it recursively. The core strategy is to reduce complexity by dividing the dataset into conditional datasets (projected databases), which are then processed separately. The concatenation of the results for each projected database gives the set of frequent patterns. The initial tree construction requires two passes over the dataset, and costs $O(N_{events})$, where $N_{events}$ is the number of events in the dataset. Each path in the tree is, at least partially, traversed $N_{items,path} * N_{items,HT}$ times, where $N_{items,path}$ is the number of items in that path (the depth of the path in the tree), and $N_{items,HT}$ is the number of different items in the tree. The cost of searching through all paths is thus bounded by $O(N^2_{items,HT} * D_{tree})$ where $D_{tree}$ is the depth of the tree. For each frequent item in the FP-tree, a recursive call to FP-growth is made. The parameters having the greatest influence on the search are thus the episode length $T_{ep}$ (which is going to determine the length of the transactions) and the minimal support threshold $S_{min}$.

### 3.4.2 Periodicity and variability characterization

The periodicity of each episode is searched as a Gaussian Mixture model (GMM, see chapter 2), that is to say a list of Gaussian components, which are characterized by their means and standard deviations. The episode is expected to occur around the means of the GMM, and the quality of the periodicity descriptions is then assessed on their accuracy.

In this section, we discuss a strategy for the tuning of the GMM to the observed occurrences of the episode. For each occurrence (happening at timestamp $t$) of the considered episode, the *relative* occurrence time ($t_r = t$ modulo $T$, where $T$ is the considered period) is computed. The GMM is trained on these occurrence times, in two steps (see pseudo-code in algorithm 2):

**Algorithm 2** The candidate study step: find the periodicity of a candidate episode

**Input:** occurrences of the considered episode, $\Delta T_{max}, S_{min}, SD_{max}, A_{min}$

**for all** candidate period $T$ **do** /* *Typically* $T = 24hours$, $T = 7days$ */
  Compute the intervals between consecutive occurrences
  **if** there is an interval which is too long ($> \Delta T_{max}$) **then**
    /* *There is a gap without any occurrence of the episode* */
    /* *The habits are likely to be different before and after this gap* */
    Split the occurrences in groups, and process each subset separately
  **for all** group of occurrences $O'$ **do**
    $L \leftarrow [\,]$ /* *Stores the relative occurrence times* */
    **for all** occurrence $o \in O'$ **do**
      Append $o.timestamp \mod T$ to $L$
    $N_{comp} \leftarrow$ Cluster $L$ with DBSCAN$(S_{min}, SD_{max})$
    $\{(\mu_1, \sigma_1), ...(\mu_{N_{comp}}, \sigma_{N_{comp}})\} \leftarrow$ Fit a GMM on $L$ using EM$(N_{comp})$
    Periodicity $P = (T, \{(\mu_1, \sigma_1), ...(\mu_{N_{comp}}, \sigma_{N_{comp}})\})$
    $\Delta t \leftarrow$ the earliest and latest occurrence in $O'$ expected by $P$
    Compute the accuracy $A$
    **if** $A \geq A_{min}$ **and** the description is the best so far **then**
      $\Delta t_{best} \leftarrow \Delta t$
      $Descr_{best} \leftarrow \{(\mu_1, \sigma_1), ...(\mu_{N_{comp}}, \sigma_{N_{comp}})\}$
**return** the best description found: $\Delta t_{best}, Descr_{best}$

1. Determination of the number of components
2. Training of the GMM

**Determination of the number of components in the mixture model**
The occurrences need to be separated between the several components of a given activity. Indeed, a *general* activity (e.g. prepare a meal, figure 2.1) is likely to generate different clusters when considering its *specific* sub-activities (breakfast and dinner). This separation is reflected in the occurrence density throughout the period $T$, and can thus be detected thanks to a density-based clustering algorithm.

The relative occurrence times are thus clustered with DBSCAN (Ester et al., 1996). This algorithm partitions the data points based on the estimated density of the clusters: a point $t_r$ belongs to a cluster $C$ when there are at least $S_{min}$ points from $C$ in the neighborhood of $t_r$. The neighborhood

size controls the desired precision and tolerance for the periodic episodes (controlled by the maximal standard deviation considered as normal $SD_{max}$). The complexity of DBSCAN is of the order of $O(N_{occ}^2)$, where $N_{occ}$ is the number of occurrences.

DBSCAN is used here as a heuristic for the estimation of the number of components. The number of clusters discovered by DBSCAN is used as the number of components in the GMM. This approach was preferred to the standard estimation via information criteria such as AIC (Akaike Information Criteria, Akaike (1998)) or BIC (Bayesian Information Criteria, Schwarz et al. (1978)) for execution speed matters. The use of AIC or BIC criteria allows the comparison of several probability distribution models in order to find a trade-off between the quality of the estimation and the complexity of the model. It thus requires the computation and evaluation of several GMM models. On the contrary, DBSCAN allows a fast estimation of the number of components, making the periodicity analysis faster.

**Training of the GMM** The periodicity descriptions of the episodes are then searched in the form of a Gaussian Mixture Model (GMM) with as many components as the number of clusters discovered by DBSCAN. GMMs are traditionally trained using the Expectation-Maximisation (EM) algorithm (Dempster et al., 1977). EM initializes the means for the latent components of the mixture model using the centers of the DBSCAN clusters. Then for every occurrence time $o$ of the episode and every component $c$ in the mixture model, the probability that $o$ was generated by $c$ is computed. The parameters of the components are then tweaked to maximize the likelihood of the data point / component assignment. The complexity of EM is of the order of $O(N_{iter} * N_{components} * N_{occ})$, where $N_{iter}$ is the number of iterations before the convergence of EM (bounded by the user), and $N_{components}$ is the number of components in the mixture model. The execution of this step depends thus highly on the number of occurrences $N_{occ}$ for the episode. The candidate study step is mostly impacted by the number of frequent episodes (each has to be analyzed separately). However, the analysis of each candidate is independent, and can thus be parallelized.

### 3.4.3 Dataset rewriting

Dataset rewriting is guided by the Minimum Description Length (MDL) principle: the episodes allowing the most compact representation of the dataset are considered as the most relevant. From an information theory point-of-view, such episodes carry more information. Following the MDL principle, xED iteratively rewrites the dataset for each interesting periodic episode, processing as follows:

1. Insertion of a header describing the episode, its validity interval, and its periodicity;
2. Removal of the occurrences observed as expected, that is to say the occurrences matching the periodicity description;
3. Creation of new events for the characterization of expected but missing occurrences. These new events can themselves become part of periodic episodes in the subsequent iterations of the algorithm;
4. The other events and episodes are left untouched.

**Definition 3.3** (Compression power)**.** The compression power of a periodic episode is the ratio between the *sizes* of the dataset before and after the compression of the periodic episode.

The database size accounts for the description of the event labels in the factorized periodic episodes, their validity intervals and periodicities, and all the other events in the dataset (including those introduced when an occurrence was missing). Compression power is thus a measure that favors long episodes, periodic episodes having many occurrences following the periodicity description, and those with few missing occurrences. It makes a rather natural compromise between the three aspects.

Thus, the periodic episodes enabling the greatest compression are described first, and used for the factorization of the dataset. As long as the factorizations remove disjoint occurrences, they can be carried out, thus saving a few iterations of the algorithm. The others periodic episodes are factorized on subsequent iterations, if they are still deemed interesting. When two periodic episodes have the same compression power, the one with the greatest accuracy is compressed first. If they also have the same accuracy, the episode that was first discovered is factorized first.

Table 3.3 – Analysis of the episodes for the dataset from figure 3.1 using xED

| Episode | Description | Errors | N | A |
|---|---|---|---|---|
| kitchen, breakfast, coffee | $\mu_1$=8:03, $\sigma_1$=13min | $\emptyset$ | 12 | 100% |
| kitchen | $\mu_1$=8:03, $\sigma_1$=13min<br>$\mu_2$=12:30, $\sigma_2$=29min | $\emptyset$ | 8 | 100% |
| kitchen, lunch, coffee | $\mu_1$=12:25, $\sigma_1$=25min | `coffee`<br>missing on $d_3$ | 10 | 75% |
| kitchen, lunch | $\mu_1$=12:30, $\sigma_1$=29min | $\emptyset$ | 8 | 100% |
| kitchen, coffee | $\mu_1$=8:03, $\sigma_1$=13min<br>$\mu_2$=12:25, $\sigma_2$=25min | `coffee`<br>missing on $d_3$ | 14 | 87.5% |

N: Number of factorized events = as expected − missing  A: Accuracy

When the dataset is rewritten thanks to the discovered interesting periodic episodes, another iteration of xED starts. The candidate search finds frequent episodes, which are then analyzed for periodicity. If the periodic episodes allow a dataset compression, the dataset is rewritten once again, until no new interesting episode is found.

### 3.4.4 Illustration with example 3.1

Table 3.3 illustrates the results of the episode analysis of xED, and the iterative rewritings of the dataset are given in figure 3.3. Episode `{kitchen, coffee}` has the greatest compression power, with a compression by 14 events (15 events are removed, 1 missing event is added): it is thus factorized first: a header is inserted, the expected occurrences of `{kitchen, coffee}` are removed, the missing occurrence of `coffee` is inserted on day $d_3$. The occurrences of the second most interesting episode (`{kitchen, breakfast, coffee}`) overlap with the already factorized episode: it is thus not factorized in this iteration.

The next iteration then starts. The frequent episodes are `{breakfast}` and `{lunch}`, both occurring four times. Since their periodic occurrences do not overlap, both are factorized during this iteration. The resulting database contains three episode descriptions in its headers, and one event: the missing occurrence of coffee on $d_3$. There are no frequent episode left in the database: the compression is maximal, and xED finishes.

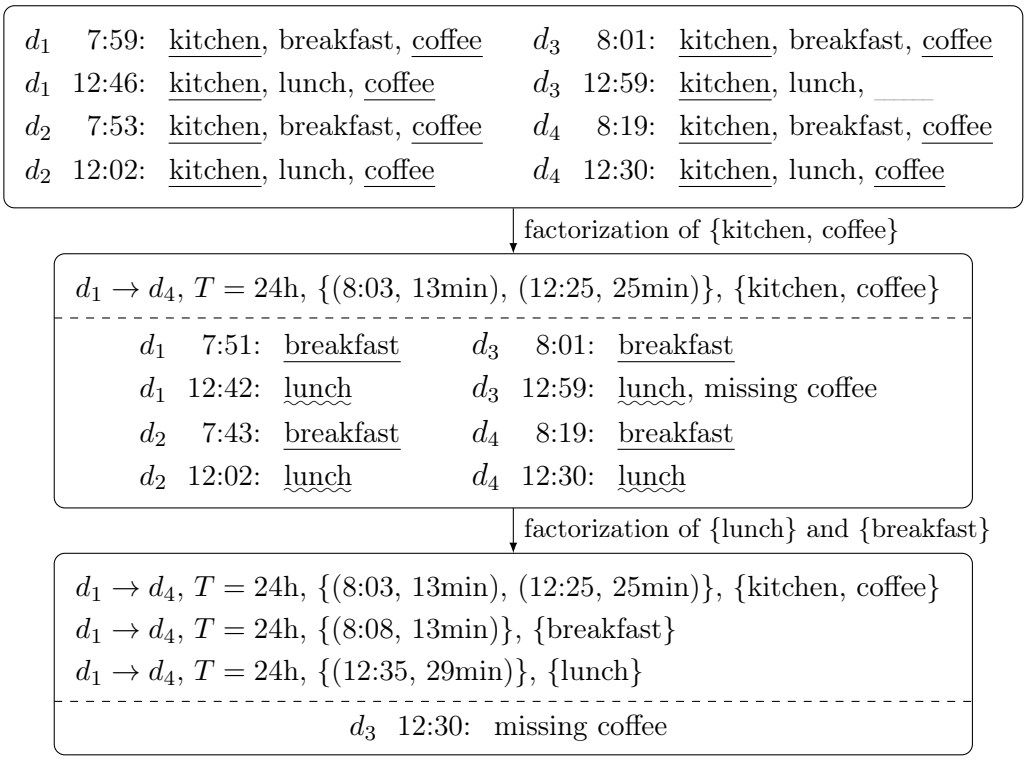


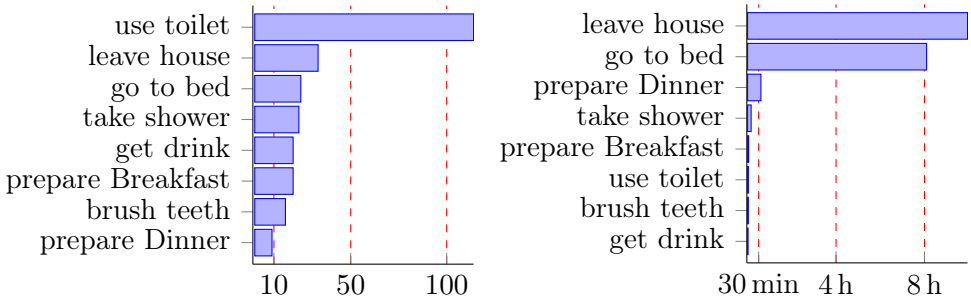

Figure 3.3 – Compression of the dataset from table 3.3

Next section details the experiments on six real-life datasets, and shows xED's ability to discover interesting periodic episodes in the data.

## 3.5 Case studies

xED is implemented in Python. It uses components from the Scikit-learn library (Pedregosa et al., 2011). xED was tested on both simulated and real-life data. The generated datasets were designed to validate xED's behavior in various typical situations (daily and weekly patterns, following uniform and Gaussian distributions, absence of expected events, presence of non-periodic noise patterns, etc.). We describe next into details the experiment on the real-life dataset Kasteren House A, subsequently referred to as KA (van Kasteren et al., 2010b). Five other datasets are also investigated. The synthetic results are explained and discussed in section 3.5.2. Comparison with ED and discussion are provided in section 3.5.3.

Figure 3.4 – Plan of the KA house (figure from van Kasteren et al. (2010b))




(a) Number of occurrences (b) Average activity duration

Figure 3.5 – Characteristics of KA dataset

### 3.5.1 KA dataset

The KA dataset was recorded between February 25th, 2008 and March 21st, 2008, and is available online[1]. Fourteen binary sensors were installed in a home, where a single 26-year-old man lived. The sensors include motion detectors between the rooms, and contact switches on the kitchen appliances: on the fridge door, the cupboards, the dishwasher, etc., as well as on the toilet flush. The 25-day experiment resulted in 2458 on / off sensor events.

The user annotated the activities being carried out using a Bluetooth headset. Each activity has a start and an end time, and these records are available with the sensor data. We focus on finding habits in these activities (518 recorded events), presented in figure 3.5. Figure 3.5a summarizes the occurrence count

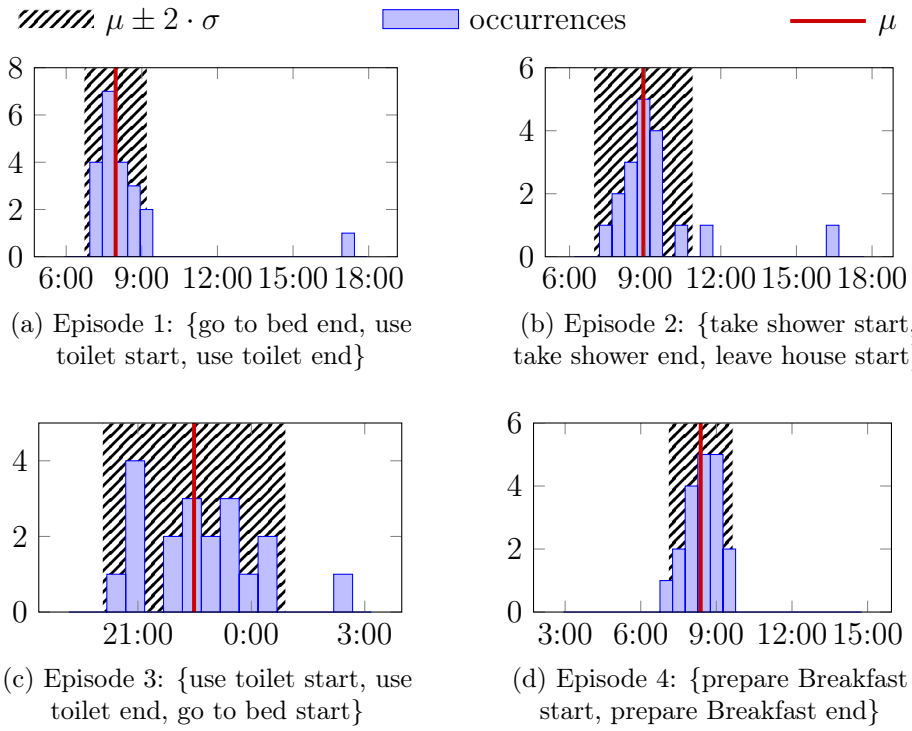
[1] https://sites.google.com/site/tim0306/datasets, last consulted September 16th, 2015

Table 3.4 – Periodic episodes discovered by xED in the KA dataset

| # | Episode | Periodicity | | | A |
|---|---|---|---|---|---|
| | | $\Delta t$ | $T$ | $(\mu, \sigma)$ | |
| 1 | go to bed end, use toilet start, use toilet end | Feb 26$^{\text{th}}$–Mar 22$^{\text{nd}}$, | 1 day , | (7:58, 0:38) | 71 % |
| 2 | take shower start, take shower end, leave house start | Feb 26$^{\text{th}}$–Mar 22$^{\text{nd}}$, | 1 day , | (8:56, 0:59) | 62 % |
| 3 | use toilet start, use toilet end, go to bed start | Feb 26$^{\text{th}}$–Mar 21$^{\text{st}}$ , | 1 day , | (22:29, 1:13) | 74 % |
| 4 | prepare Breakfast start, prepare Breakfast end | Feb 26$^{\text{th}}$–Mar 21$^{\text{st}}$ , | 1 day , | (8:23, 0:39) | 74 % |
| 5 | leave house end | Feb 26$^{\text{th}}$–Mar 20$^{\text{th}}$, | 1 day , | (17:57, 1:27) | 70 % |
| 6 | use toilet start, use toilet end, leave house start | Mar 11$^{\text{th}}$–Mar 19$^{\text{th}}$, | 1 day , | (18:11, 0:59) | 60 % |
| 7 | brush teeth start, brush teeth end | Feb 25$^{\text{th}}$– Mar 7$^{\text{st}}$ , | 1 day , | (22:33, 0:27) | 60 % |
| 8 | use toilet start, use toilet end, prepare Dinner start | Mar 5$^{\text{th}}$ –Mar 11$^{\text{th}}$, | 1 day , | (18:22, 0:24) | 100 % |
| 9 | get drink start, get drink end | Feb 25$^{\text{th}}$–Feb 28$^{\text{th}}$ , | 1 day , | (19:27, 0:57) | 100 % |
| 10 | brush teeth start, brush teeth end | Mar 17$^{\text{th}}$–Mar 21$^{\text{st}}$ , | 1 day , | (23:29, 0:23) | 100 % |
| 11 | Missing occurrence of episode #4 | Feb 28$^{\text{th}}$–Mar 13$^{\text{th}}$, | 1 week, | (Sun 8:23, 0) (Tue 8:23, 0) | 75 % |

of the activities over the dataset (the least frequent event appears 9 times). We set $S_{min} = 3$ in order to enable the discovery of weekly episodes (the dataset lasts slightly less than four weeks). The episode length $T_{ep}$ is set to 30 minutes, which is long enough to consider as part of one episode both the beginning and end of short activities, like preparing breakfast, or using the toilet, as described by figure 3.5b. The tolerance ratio $a$ is set to 2: if the time distribution of an episode follows a normal distribution (reasonable assumption when considering habits), an average of 95% of the occurrences will be classified as expected occurrences.

Out of the 56 frequent episodes present in the dataset, 11 were deemed




(a) Episode 1: {go to bed end, use toilet start, use toilet end}

(b) Episode 2: {take shower start, take shower end, leave house start}

(c) Episode 3: {use toilet start, use toilet end, go to bed start}

(d) Episode 4: {prepare Breakfast start, prepare Breakfast end}

Figure 3.6 – Histograms for the top-four periodic activities discovered in KA (period $T = 1$ day)

interesting thanks to their good accuracy and compression power. Table 3.4 summarizes all the periodic episodes that were discovered. Figures 3.6 and 3.7 show respectively for the top-4 most interesting episodes: the histograms of their occurrence time, and a time line showing both the times at which an activity is expected to happen, and the events that were actually recorded during the experiment.

The discovered habits are the episodes 1 to 11. One can for instance notice (episode 1) that the user goes to the toilet after waking up, which happens around 07:58. Other morning routines are also found, including taking a shower and leave house (episode 2) and preparing breakfast (episode 4). Episode 4 is modified by episode 11. Indeed, episode 4 was not observed as expected on Sundays and Tuesdays in the first half of the dataset: the missing occurrences are themselves regular, hence the discovery of episode 11. The standard deviation of the components may grow to be quite large (e.g. episode 5, with a standard deviation of 5234s = 87 minutes). During the clustering of the occurrence times of this episode by DBSCAN, most oc-

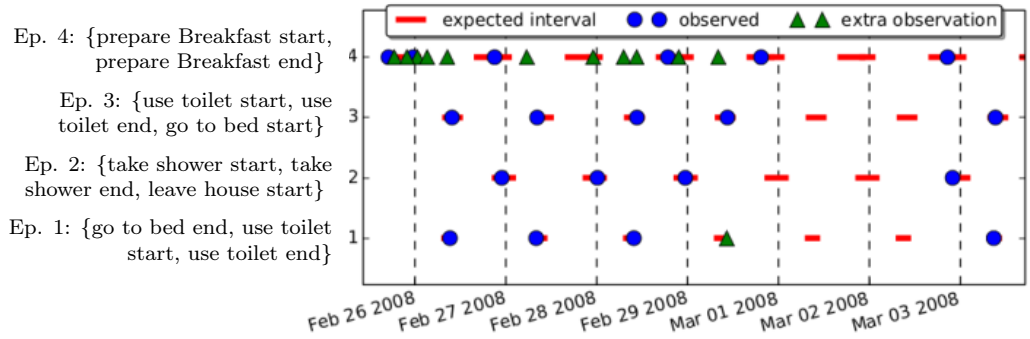



Figure 3.7 – The expected vs the observed occurrences, for the top-4 best periodic episodes in KA, during the first week of recorded data

currences are kept in a single cluster: the standard deviation is high. With a smaller value for the maximal standard deviation parameter $SD_{max}$ (here the default value, 10% of the period, i.e. a bit less than 2.5 hours), this episode might not have been considered periodic.

In the histograms figure 3.6, the hashed area corresponds to the time interval $[\mu - 2 \cdot \sigma,\ \mu + 2 \cdot \sigma]$ (where $\mu$ and $\sigma$ are the mean and standard deviation of the components in the periodicity description). Some habits occur only in the expected interval (episode 4, bottom right histogram), when others happen also at other times (episodes 1, 2, 3).

Figure 3.7 allows day by day comparison, and highlights for each episode the actually observed occurrences. In particular, the expected but missing occurrences can be spotted where one can see an expected interval, but no (round dot) observation occurs in the interval.

### 3.5.2 Synthetic results for six real-life datasets

Experiments were done with six datasets coming from real-life experiments, where one person carried out daily life routines in a smart home, without any kind of supervision or scripted scenarios. The used datasets are:

- the three datasets presented in (van Kasteren et al., 2010b), subsequently referred to as KA (described in the previous paragraph), KB, and KC;

- the two datasets presented in (Tapia et al., 2004), referred to as T1 and T2 (consulted on September, 16$^{\text{th}}$ 2015 http://courses.media.mit.edu/2004fall/mas622j/04.projects/home/);

- the Aruba dataset (Cook, 2012), referred to as CA (http://ailab.wsu.edu/casas/datasets/index.html, consulted on September, 16$^{\text{th}}$ 2015).

Table 3.5 summarizes the characteristics of the six tested datasets.

Each dataset was processed separately, with an episode length of 30 minutes, and $S_{min}$ such that episodes are frequent if they happen, on average, at least once a week ($S_{min} = 3$ for KA, $S_{min} = 2$ for KB, KC, T1 and T2 and $S_{min} = 30$ for CA). The discovered episodes are different from one dataset to the next. This is normal: the sensors are different, and the users have different habits. Nevertheless, some ADL patterns appear in several datasets, notably those describing eating and sleeping patterns. As already noted, the ADLs determine one's ability to live independently, and as thus very interesting patterns in activity monitoring.

In order to allow a cross-dataset summarized visualization, and a comparison of the habits on standard ADLs, the discovered periodic episodes were manually investigated, and classified in five categories thanks to the available annotations: bedtime, waking up, eating, hygiene-related or other. The daily habits matching the first four categories are represented in figure 3.8. The represented band widths are the discovered time intervals where the episodes are expected to happen ($\pm 2 \cdot \sigma$, where $\sigma$ is the standard deviation of the considered periodic episode). These results highlight the high inter-subject variability when it comes to habits, and emphasize the need for algorithms which do not require a priori knowledge on the subject.

### 3.5.3 Comparison between ED and xED

ED and xED are compared on the 6 activity datasets. Both algorithm discover interesting episodes, though not necessarily the same, since the periodicity descriptions are different (see overview of the two periodicity definitions table 3.6). Figure 3.9 presents some characteristics of the candidate and periodic episodes discovered by ED and xED for the 6 datasets. The following paragraphs highlight the main differences in the results for both algorithms.

Table 3.5 – Characteristics of the six datasets used for the validation of xED

| Dataset | Age | Gender | Duration | # activity labels | # events |
|---|---|---|---|---|---|
| KA | 26 | Male | 25 days | 8 | 518 |
| KB | 28 | Male | 14 days | 8 | 224 |
| KC | 56 | Male | 19 days | 8 | 468 |
| T1 | 30 | Female | 16 days | 22 | 590 |
| T2 | 80 | Female | 15 days | 24 | 416 |
| CA | Unknown (has grandchildren) | Female | 220 days | 11 | 12953 |

Table 3.6 – Comparative summary: periodicity description for ED and xED

| ED | xED |
|---|---|
| • Granularity (fine-/coarse-grained)<br>• Date of the first occurrence<br>• List of periodicity cycles (list of length-$l$ lists of intervals)<br>• Mistake list: the observed occurrences that did not respect the expected intervals, since they came either too late (missing occurrence) or too early (extra). The periodicity of an episode at time $t$ is the $i^{\text{th}}$cycle, with $i$ such that $t_{mistake_i} < t < t_{mistake_{i+1}}$. | • Validity interval $\Delta t$<br>• Period (1 day, 1 week)<br>• List of $(\mu, \sigma)$ pairs describing the statistical distribution of the periodic occurrences<br>• Mistake list (expected but missing occurrences): events are created and introduced in the dataset, e.g. `2014-12-20 12:34:56 Missing occurrence of episode #12`. They are thus available, but not required for the understanding of the periodicity. |

**Candidate and periodic episode count (figure 3.9a)** The minimal support threshold in xED reduces the number of candidates. The greater tolerance to extra occurrences increases the number of interesting periodic episodes (denoted "factorisation count" in the figure).

**Episode support (figure 3.9b)** Since xED uses a minimal support threshold, xED candidate episodes are on average slightly more frequent than ED's: amongst the candidates for ED, some episodes are rare. The support of factorized episodes tends to be higher in ED than in xED. This comes from the way both algorithms compute the periodicities: xED penalizes more the missing occurrences, and favors more the longer episodes with a clear periodic

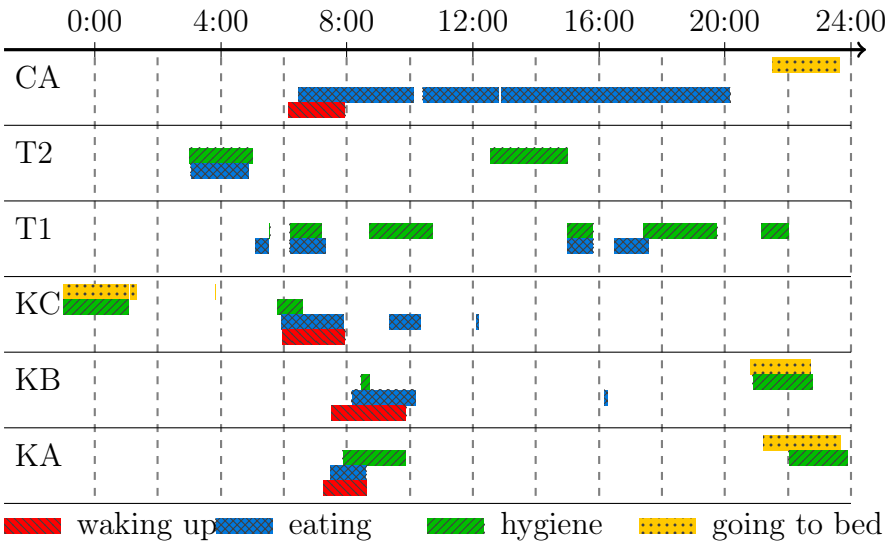



Figure 3.8 – Overview of the periodic episodes discovered by xED in the six activity datasets

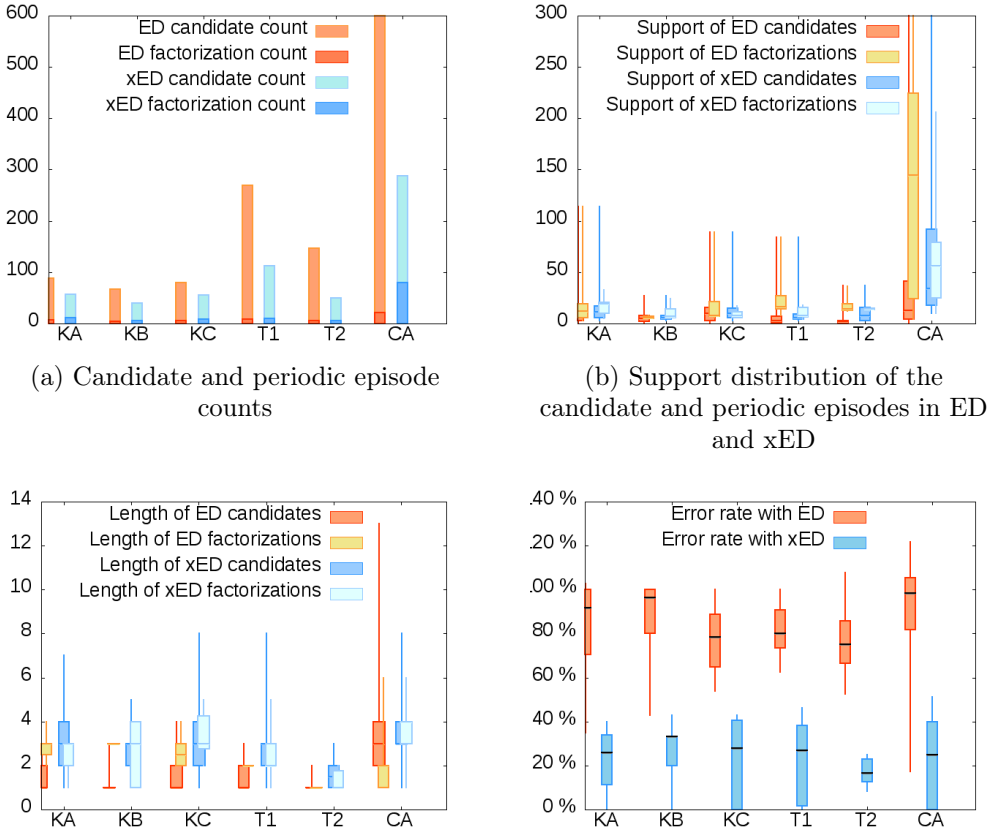
component, rather than the very frequent ones.

For example in the KA dataset, ED factorizes first {`use toilet begin`, `use toilet end`} (table 3.7 presents all periodic episodes in KA), and removes all the occurrences before looking for other interesting episodes. On the contrary, xED differentiates several contexts in which this activity occurs, and outputs episodes such as {`go to bed end`, `use toilet`, `use toilet end`} (i.e. wake up and go to the bathroom, in the morning), {`use toilet`, `use toilet end`, `go to bed`} (i.e. go to the bathroom and go to bed, which occurs in the evening), {`use toilet`, `use toilet end`, `leave house start`}, etc.

Table 3.7 – Interesting periodic episodes discovered by Episode Discovery

| **Episode** | **CP** | **Granularity** |
|---|---|---|
| use toilet start/end | 1.08 | coarse |
| take shower start/end; leave house start | 1.07 | coarse |
| prepare Breakfast start/end | 1.05 | coarse |
| brush teeth start/end; go to bed start | 1.03 | fine |
| leave house end; get drink start/end | 1.02 | coarse |
| leave house end; brush teeth start/end; leave house start | 1.01 | fine |
| prepare Dinner end; get drink start/end | 1.01 | fine |

CP: compression power




(a) Candidate and periodic episode counts

(b) Support distribution of the candidate and periodic episodes in ED and xED

(c) Length distribution of the candidate and periodic episodes

(d) Distribution of the error rates for the periodic episodes ED and xED discovert

Figure 3.9 – Quantitative comparison of the results of ED and xED over the 6 datasets

**Episode length (figure 3.9c)** The candidates are longer for xED, and the factorized episodes are as long, or longer. This phenomenon is linked with the lower frequency of the xED's periodic episodes: longer episodes characterize better activity contexts. They are thus more likely to follow a periodic pattern, in the meaning of xED, but they are less frequent than their shorter sub-episodes.

**Description relevance and uncovered mistakes** Both algorithms do not focus on the same periodicity definition (table 3.6), and discover thus fairly different episodes and periodicity descriptions. Each time ED records

a mistake, a new periodicity cycle is computed. The main advantage is that it takes into account the evolution of habits with time. But in practice, many such lists are created: a lot of mistakes are reported (figure 3.9d shows a very high error rate). This makes the interval lists harder to use for the supervisor (physician, family member).

When ED includes all the occurrences of an episode in the periodicity description, xED distinguishes the periodic occurrences from the other occurrences. These extra occurrences are not considered as abnormal, and are left untouched in the dataset. They can participate in the periodicity analysis of other episodes. xED reports thus less events as anomalous (figure 3.9d). In practice, this makes xED periodicity descriptions more readable, and easier to understand by the supervisor.

### 3.5.4 Discussion

xED allows the discovery of activity patterns in event databases. As of now, it relies a lot on the choice of the parameters. These parameters are primarily based on expert knowledge, as well as statistical analysis and empirical observations. An effort was produced in order to use parameters that are in direct link with the searched behaviors: the duration of the activities, their frequencies, by how much their occurrence times may vary, etc. All these parameters are understandable and can be set by a caregiver (a physician in particular). Preliminary results were presented to experts in the field of functional rehabilitation (Kerpape rehabilitation center, in Brittany, France), and they agree on the relevance of the discovered patterns.

However, an unsuitable parameter setting may lead to the missing of interesting information, such as occurrences of episodes slightly longer than the episode length parameter $T_{ep}$. $T_{ep}$ acts as a good heuristic for the candidate episode generation, since it reduces the complexity of the search space, and emphasizes on the importance of temporal proximity. But other candidate episode generation techniques should also be investigated in the future, such as automated segmentation, or change point detection.

The comparison with ED highlights the differences between ED and xED, both in the theoretical approach and in the obtained results. The main difference comes from the way extra unexpected events are handled: when ED fits all the occurrences of episode in the periodicity model, xED focuses on the periodic components of episodes. Some occurrences may thus remain

out of the periodicity description. A first consequence is that xED adapts better to the variability inherent in real-life datasets. Another is that the periodicity descriptions are more informative, and are useful to monitor the habits.

## 3.6 Conclusion

The discovery of habits is of importance for the health monitoring of elderly people. In particular, when linked to anomaly detection tools, it enables the raise of alarms when unusual and possibly dangerous situations are detected. Knowing the habits also allows the observation of change in the habits, which can point out the onset of a disorder. We propose and analyze xED, which enables the efficient discovery of habits. This unsupervised algorithm characterizes periodic episodes, and quantifies the customary variability of these habits.

The experiment with six datasets shows that xED is able to discover habits from event logs. The habits differ a lot from one subject to the other. The variability attached to each habit is also user-dependant and episode-dependent. This highlights the need for personalized, adaptive monitoring tools, which justifies our approach. These results show the ability of xED to consistently extract ADLs in data coming from various houses, with different inhabitants.

xED has also been extended for the handling of data streams (chapter 4), in order to enable the real-time discovery and update of our knowledge of the habits, as well as the analysis of their evolution. Interesting perspectives also include the investigation of outlier detection, of an evaluation framework, as well as data visualization tools.

# Chapter 4

# Periodic frequent episode discovery in event streams

## 4.1 Introduction

As highlighted in the previous chapter, the discovery of habits is an interesting subproblem of activity monitoring in SHAAL systems. xED provides a first approach for the discovery of such patterns. However, it also shows drawbacks that limit its ability to perform efficiently in real-world scenarios:

- The iterative process allows the discovery of refined episodes, and most particularly episodes involving expected but missing behaviors (such as episode #11, table 3.4), but this process is time-consuming: the complexity increases proportionally with the number of iterations,
- xED focuses mostly on the definition of ways to define, evaluate and compute the periodicity of patterns. The candidate episodes are generated using basic itemset mining techniques. This naive approach is however impractical in the context of stream mining, and calls for improvements in the formalisms and techniques used for the selection of the candidates,
- xED considers static datasets, and habits are assumed to be stable. However, in practice, habits evolve with time.

**Habit evolution** Habits are not stationary. They may change because of seasonal change: the daylight hours and the weather vary, and thus also the activities outside of the house. The habits may also change because someone else's planning changes (e.g., the caregiver changes his/her visit schedule), or because of the user chooses to change his/her routine. Some changes might also not be voluntary. They may be linked to a greater difficulty to carry out every day activities, or to the forgetting of some habits. Monitoring algorithms should thus also be capable to adapt to concept drifts, and update their knowledge when new events are recorded.

**Stream processing** Moreover, monitoring needs to produce results in a real-time setting: the processed events form a data stream. With the rapidly increasing amount of data recording devices (network traffic monitoring, smart houses, sensor networks,...), stream data mining has gained major attention. This evolution leads to paradigm shifts. For an extensive problem statement and review of the current trends, see Gama (2012). In particular, data streams tend to be:

- Voluminous: the complete dataset may not fit in the main memory,
- Unbounded: new data points keep arriving, and
- Evolving: the data distribution is generally not stationary.

These observations set new constraints on the algorithms used for the mining of such data sources (Gama, 2012):

- The model should be built using a single pass over the data. Old data points cannot be processed again.
- The integration cost of a new data point in the model should be small, constant, and independent from the number of data points already used in the model.
- The memory usage should be constant, and independent from the number of data points already used in the model.
- The models should be capable of adapting to concept drifts. Here in the context of habit monitoring, habits may change with time, new habits may emerge, some habits may disappear, and the periodicity descriptions can evolve.

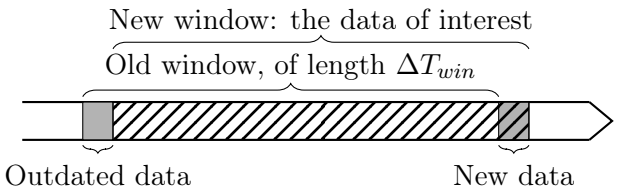



Figure 4.1 – Sliding window framework

Most traditional data mining algorithms do not satisfy these constraints. Several frameworks have been proposed to cope with this situation. In particular, sliding windows are very popular (see the explanatory schema figure 4.1). In this framework, only a subset of the data points is kept in the main memory: the most recent ones. When new data points arrive, the oldest points in the window become outdated. They are either just removed from the window, and/or the model is also updated to cancel their influence.

**Contribution** In order to adapt habit monitoring to a changing streaming context, we propose sxED (streaming extended Episode Discovery) for the discovery of frequent and periodic episodes over event streams (Soulas and Lenca, 2015). The main contributions of sxED are:

- A new structure and algorithm for the discovery and update of the frequent parallel episode data streams,
- A heuristic for the online estimation of the periodicity of the episodes.

**Chapter outline** The rest of this chapter is organized as follows: section 4.2 introduces episode mining, presents the formalisms that are in use and links them with some prominent related work. Section 4.3 details our proposition for frequent periodic pattern mining and updating in an event stream. Experiments on real-life datasets illustrate the interest of this approach in section 4.4. Finally, some conclusions are drawn, and ideas for future work are presented (section 4.5).

## 4.2 Mining event sequences: formalisms and related work

This section introduces episode mining, and reviews the approaches that have been used for episode mining so far. Frequent episode mining is a data mining task introduced by Mannila et al. (1995), for the discovery of interesting patterns in sequences of events (see definition 4.1).

**Definition 4.1.** An *event* is a pair $(e, t)$, where $e$ is the event *label* and $t$ is the *timestamp*.

The event label usually corresponds to a sensor reading, an identifier, an alarm code, etc., and takes values in a finite alphabet $\mathcal{A}$. The timestamps induce a total order on the events, and allow to compute a distance on the events: the time gaps / time intervals.

Episode mining focuses on the discovery of episodes, that is to say relevant collections of event labels in an event sequence. Contrarily to traditional itemset mining approaches, the temporality (order and distance) of the events is a most important feature.

Over the last twenty years, episode mining has attracted a lot of attention, and was refined into several subproblems, depending on the type of episodes that are searched, the episode support measures, or the behaviors of interest that are mined. These different concepts are introduced and discussed in the following subsections.

**Example 4.1.** Figure 4.2 presents a sample event sequence, which is used as a running example to illustrate the different definitions and concepts in this section (4.2) and the next (4.3).

The displayed window of example 4.1 contains 12 events, with labels taking values in the alphabet $\mathcal{A} = \{A, B, C, D\}$. The last seen event is (A, 62). The timestamps are represented by integer numbers and are regularly sampled in this example, but it is usually not true for all event sequences. In the context of sensor networks, the timestamps are composed of a date and a time, and sensors are triggered by activity in the home, and are thus not evenly spread over time.

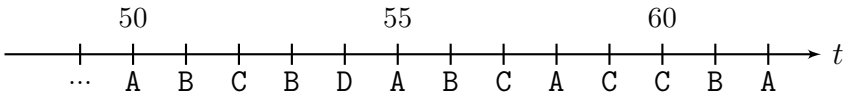
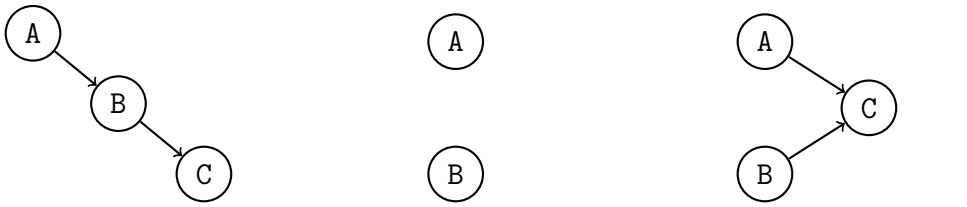



Figure 4.2 – Example: a segment of an event stream




(a) `EP1` Serial episode: `A` occurs, then `B`, then `C`

(b) `EP2` Parallel episode: `A` and `B` occur, in any order

(c) `EP3` General episode: `A` and `B` occur, in any order, followed by `C`

Figure 4.3 – Examples of episodes

### 4.2.1 Episode families

Mannila et al. (1995) define episodes as partially ordered collections of event labels. When the order is total, the episodes are called *serial*. When there are no order constraints, the episode is called *parallel*. Episodes are traditionally assimilated to oriented graphs, where each node represents a label in the episode, and edges denote the "happens before" relationship. Figure 4.3 presents examples of serial (4.3a), parallel (4.3b) and general (4.3c) episodes.

**Definition 4.2** (Episode occurrence)**.** An episode $E$ *occurs* in an event sequence $S$ if there are events in $S$ having the same labels as those of $E$ and respecting the order constraints. These events then form an *occurrence* of $E$.

In example 4.1, `EP1` (figure 4.3a) occurs for example on ⟨ (`A`, 50), (`B`, 51), (`C`, 52) ⟩ or on ⟨ (`A`, 50), (`B`, 53), (`C`, 57) ⟩, etc. `EP2` (figure 4.3b) occurs on ⟨ (`A`, 50), (`B`, 51) ⟩ or on ⟨ (`B`, 53), (`A`, 55) ⟩. `EP3` (figure 4.3c) occurs for instance on ⟨ (`A`, 50), (`B`, 53), (`C`, 57) ⟩, or on ⟨ (`B`, 53), (`A`, 55), (`C`, 57) ⟩.

If Mannila et al. (1995), as well as subsequent studies propose methods for the discovery of all kinds of episodes, they also acknowledge that the complexity of mining general episodes can be prohibiting: the general episodes are built by testing all the combinations of serial and parallel episodes. Most studies thus focus either on serial episodes or parallel episodes.

### 4.2.2 Episode support measures

If it is fairly straightforward to count the support of an itemset in a transactional database, it is however harder to settle on a single support definition when it comes to episodes. Three main trends have emerged:

- Window-based support,
- Minimal occurrence count, and
- Maximal count of distinct occurrences.

**Window-based methods** They use a sliding window of fixed size (the maximal duration of the occurrences), and count the number of windows in which the episode occurs (Mannila et al., 1997; Casas-Garriga, 2003; Tatti and Cule, 2011). This means that even if an episode occurs several times in a window, it is counted only once. When a same occurrence appears in several windows (which is often true), it also means that it is counted several times. This measure thus tends to favor short episodes: a short episode occurrence is more likely to appears in many windows than an occurrence of a longer episode.

**Minimal occurrences** Another popular measure is the count of minimal occurrences (Mannila and Toivonen, 1996; Zhu et al., 2010; Zhou et al., 2010; Lin et al., 2014), see definition 4.3.

**Definition 4.3** (Minimal occurrence - MO)**.** Let $E = \{e_1, ...e_n\}$ be an episode, and $o$ an occurrence of $E$ starting at timestamp $t_1$ and finishing at timestamp $t_n$. $o$ is a *minimal occurrence* if there is no other, shorter occurrence that occurs within the time interval $[t_1, t_n]$. That is to say, there cannot be an occurrence $o'$ starting in $t'_1$ and finishing in $t'_n$ such that $t_1 \leq t'_1$, $t'_n \leq t_n$ and $t'_n - t'_1 < t_n - t_1$.

Counting minimal occurrences does not have the same bias as window-based support counting: each occurrence is counted exactly one time. However, each event can participate to several occurrences: the minimal occurrences may overlap.

**Distinct occurrences** With both previous methods, some events participate in several occurrences of an episode. This lead to the creation of other support measures: the counting of the maximal number of distinct occurrences (Patnaik et al., 2012), or the counting of non-overlapping occurrences. Two distinct occurrences are occurrences such that they do not share common events. Two non-overlapping occurrences span over non-overlapping intervals: the first occurrence finishes before the second one starts.

This last support family presents a natural definition: each occurrence is counted at most once, and each event may participate in only one occurrence. However, computing the maximal count of distinct occurrences is more complex.

**Hybrid measures** In order to benefit from the properties of different support families, the measures have also been combined. For example, minimal occurrences are sometimes counted only if they are distinct from the already counted minimal occurrences (Lin et al., 2014), non-overlapping (Zhu et al., 2010), or if they last for less than a maximal duration bound, which is the approach used here in sxED.

**Illustration** Depending on the definition used to count support, a same episode may have different support values. Let us now consider the different support values for episode `EP3` (figure 4.3c) on example 4.1:

- Window-based support, with a window of size 4: we slide a window over the example data. Episode `EP3` occurs in the window starting at timestamps 50, 53, 54, 55 and 56. It's support is thus 5.
- Minimal occurrences-based support: episode `EP3` has a support of 3, and its minimal occurrences are:
  - ⟨ (`A`, 50), (`B`, 51), (`C`, 52) ⟩,
  - ⟨ (`A`, 55), (`B`, 56), (`C`, 57) ⟩, and
  - ⟨ (`B`, 56), (`A`, 58), (`C`, 59) ⟩.
- Distinct occurrences: the support of episode `EP3` is 3, and the distinct occurrences are:
  - ⟨ (`A`, 50), (`B`, 51), (`C`, 52) ⟩,

  - ⟨ (B, 53), (A, 55), (C, 57) ⟩ and
  - ⟨ (B, 56), (A, 58), (C, 59) ⟩.

- Non-overlapping occurrences: the support of EP3 falls back to 2 when considering non-overlapping occurrences:
  - ⟨ (A, 50), (B, 51), (C, 52) ⟩,
  - ⟨ (B, 53), (A, 55), (C, 57) ⟩.

### 4.2.3 Types of searched patterns

The most common task in episode mining is to find all the frequent episodes, that is to say all the episodes having a support higher than a user-defined support threshold. But similarly to the developments that were proposed in the itemset mining community, other interesting tasks have also been investigated. Such tasks include for instance the search for:

- Emerging/submerging patterns (Gan and Dai, 2014), i.e. episodes whose support changes significantly with time,
- The top-$k$ most frequent patterns (Patnaik et al., 2012), where instead of constraining the minimal support of the frequent episodes, the user chooses how many results the episode should yield,
- Closed episodes (Zhou et al., 2010; Tatti and Cule, 2011), i.e. episodes such that no super-episodes has an equal or greater support.

### 4.2.4 Conclusion

Table 4.1 summarizes the characteristics of some prominent episode mining algorithms. While most combinations of episode types, support measures, search strategies and constraints have been explored, most approaches do not adapt well to data streams, nor to a deeper periodicity analysis.

**Positioning** With sxED, we search the periodic parallel episodes in event streams. The support is computed via the count of the non-overlapping minimal occurrences. We also introduce a maximal occurrence duration bound.

Table 4.1 – Characteristics of some prominent episode mining algorithms

| Algorithm | Searched episodes | Support measure | Strategy | Stream |
|---|---|---|---|---|
| Mannila and Toivonen (1996) | General | MO | A-priori | No |
| Mannila et al. (1997) | General | WF | A-priori | No |
| Casas-Garriga (2003) | Serial + Parallel | WF | A-priori | No |
| Zhu et al. (2010) | Serial | Non-overlapping MO | A-priori | No |
| Zhou et al. (2010) | Closed serial | MO | A-priori | No |
| Tatti and Cule (2011) | Closed general | WF | PG | No |
| Patnaik et al. (2012) | General | Distinct | A-priori | Yes |
| Lin et al. (2014) | Serial | Distinct MO | PG | Incremental |
| Gan and Dai (2014) | Serial | Custom WF | PG | Yes |
| sxED (Soulas and Lenca, 2015) | Parallel | Non-overlapping MO | PG | Yes |

MO: minimal occurrences WF: window frequency PG: pattern growth

We propose in the next section an episode mining strategy for the discovery of the frequent episodes, that also allows us to maintain the necessary information for the computation and update of the periodicity of the episodes.

## 4.3 Frequent periodic pattern discovery and update: the sxED algorithm

Ambient sensors and probes register events, forming a stream, where new events arrive continuously, but not necessarily regularly. We process the event stream using a sliding window, whose duration is denoted $T_W$.

Habits are searched in the form of frequent and periodic episodes. In a smart home setting, there may be delays with the activation of sensor events. For example in a smart kitchen, the motion detector might detect the person entering in the room either before the person starts using some kitchen appliances, or just after. It is usually just a matter of a couple seconds. In such

situation, the relative order between the sensor readings is not relevant. We thus focus on the discovery of *parallel episodes*, which we subsequently simply refer to as *episodes* (definition 4.4), and measure the support (definition 4.5) as the count of the non-overlapping occurrences.

**Definition 4.4** (Episode, episode length)**.** An *episode* $E$ is a set of $n$ distinct event labels $\{e_1, ..., e_n\}$. The order of the labels is not constrained. The *length* of episode $E$ is $n$.

Similarly to xED, described in chapter 3, a constraint $T_{ep}$ on the *maximal occurrence duration* can thus be set. Occurrences with a duration longer than $T_{ep}$ are then discarded. $T_{ep}$ exploits expert or statistical knowledge regarding the expected duration of the activities of daily living. It also serves as a heuristic for the reduction of the search space (see section 4.3.4.1 for more detail).

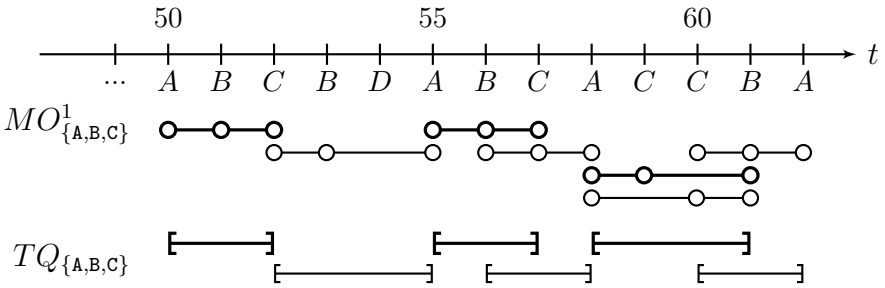
**Definition 4.5** (Support)**.** The *support* of an episode is the count of its non-overlapping minimal occurrences that last for less than $T_{ep}$.

A detailed overview of the measures and algorithms used to discover and characterize habits was discussed in chapter 2. Since the periodicity description used in xED is satisfactory for the discovery of the habits of the inhabitant of a smart house, sxED uses the same definitions and models too. That is to say, that the periodicity of an episode is described via a distribution of its *relative* occurrence times within the period of interest (e.g 1 day, 1 week), thanks to a Gaussian Mixture Model (GMM). The quality of a periodicity description is evaluated on its *accuracy*, that is to say the proportion of the expected occurrences that were actually observed.

Next sections (4.3.1, 4.3.2 and 4.3.3) present the data structures used for episode mining with sxED. The frequent episode update strategy when a new event is recorded is detailed in section 4.3.4. The methods proposed for periodicity computation and update are described in section 4.3.5. Finally, section 4.3.6 summarizes how sxED works.

### 4.3.1 Time queues

We define the time queue of an episode (definition 4.6) following the original definition from Mannila et al. (1995) (who call it then the *occurrence list*) and the formalism of (Lin et al., 2014).




[1] The non-overlapping minimal occurrences are marked with a bolder stroke

Figure 4.4 – Minimal occurrences and Time queue of episode {A, B, C}

**Definition 4.6** (Time queue – TQ)**.** The time queue of an episode $E$, noted $TQ_E$, is the list of the distinct pairs of start and end timestamps of its minimal occurrences. The time queue entry corresponding to an occurrence $o$ is noted : $[o.start, o.end]$.

**Illustration** on the example 4.1 (first described page 4.1) with a maximal occurrence duration $T_{ep} = 3$, let us consider episode {A, B, C}:

- The events composing an occurrence are not necessarily contiguous: (D, 54) occurs in the middle of occurrence ⟨ (C, 52), (B, 53), (A, 55) ⟩.
- ⟨ (A, 50), (B, 51), (C, 52) ⟩ is minimal, but ⟨ (A, 50), (C, 52), (B, 53) ⟩ is not.
- {A, B, C} has 7 minimal occurrences, spanning over 6 different intervals (⟨ (A, 58), (C, 59), (B, 61) ⟩ and ⟨ (A, 58 ), (C, 60), (B, 61) ⟩ span over the same interval [58, 61]). For the complete list of minimal occurrences and time queue entries, see the schema figure 4.4.
- Among these 7 minimal occurrences, we can pick a maximal of 3 non-overlapping occurrences, or also 3 non-overlapping intervals in the time queue. The support of {A, B, C} is thus 3.

**Properties of time queues.** Minimal occurrences and time queues have convenient properties for the mining of interesting episodes. Namely:

**Property 4.1.** Every event with label $e$ corresponds to a minimal occurrence of the length-1 episode $\{e\}$. In every such occurrence $o$, $o.start = o.end$.

**Property 4.2.** An episode $E$ has at most one time queue entry that starts at a given timestamp $t$.
*Proof:* If $tq_1 = [t_s, t_e]$ and $tq_2 = [t_s, t'_e]$ are distinct time queues entries, then $t_e \neq t'_e$. If $t_e < t'_e$, then $tq_2$ cannot correspond to a minimal occurrence (contradiction with definition 4.3). Otherwise, then it is $tq_1$ that cannot correspond to a minimal occurrence.
Similar observations show that an episode has at most one time queue entry that finishes at a given timestamp.

**Property 4.3** (Downward closure of the support measure)**.** For $E$ an episode and $E'$ one of its sub-episodes ($E' \subset E$), the support (i.e. the count of non-overlapping minimal occurrences) of $E'$ is greater or equal to this of $E$. The support verifies the downward closure property (Zhu et al., 2010).

**Property 4.4.** Knowing the time queues of two episodes $E_1$ and $E_2$, we can easily build the time queue of the union episode $E = E_1 \cup E_2$. The procedure to build the time queue of $E$ is detailed in algorithm 3. This property is used for the construction of the candidate frequent episodes.

**Property 4.5.** A new event $(e, t)$ can be part of an occurrence of episode $E = \{e\} \cup E'$ (where $e \notin E'$) if the latest entry $[t'_s, t'_e]$ in the time queue of $E'$ started less than $T_{ep}$ prior to the arrival of the new event (i.e. $t - t'_s \leq T_{ep}$). Moreover, this occurrence is minimal if the beginning of the last entry $TQ_{E'}$ starts strictly after the latest entry $[t_s, t_e]$ in the time queue of $E$ ($t'_s > t_s$). Then, the time queue entry $[t'_s, t]$ can be added to the time queue of $E'$.
*Proof:* Otherwise, it means that there is already an occurrence of $E$ starting at timestamp $t'_s$. There would then be two time queue entries starting at $t'_s$, which is not possible according to property 4.2.

Property 4.5 gives a particular importance to the recently observed episodes for the construction of the occurrences of the longer episodes. Recently observed episodes are episodes whose last minimal occurrence started less than $T_{ep}$ before the currently observed event.

### 4.3.2 Episode Lattice

The frequent episodes and their time queues are stored in a frequent episode lattice (FEL). Each node in the lattice corresponds to an episode. However,

**Algorithm 3** Computation of the time queue $TQ$ and support $s$ of episode $E = E_1 \cup E_2$ from the time queues of $E_1$ and $E_2$

**Input:** $TQ_1$ and $TQ_2$ the time queues of $E_1$ and $E_2$, indexed respectively on $i$ and $j$

$i \leftarrow 0$; $j \leftarrow 0$
$TQ \leftarrow [\,]$; support $s \leftarrow 0$
**while** $i < |TQ_1|$ and $j < |TQ_2|$ **do**
    **if** $TQ_1[i]$ finishes after $TQ_2[j]$ **then**
        Increment $j$ as long as $TQ_2[j]$ ends before $TQ_1[i]$
    **else**
        Increment $i$ as long as $TQ_1[i]$ ends before $TQ_2[j]$
    $start \leftarrow \min(TQ_1[i].start, TQ_2[j].start)$
    $end \leftarrow \max(TQ_1[i].end, TQ_2[j].end)$
    **if** $end - start \leq T_{ep}$ **then** */* New minimal occurrence */*
        Add $(start, end)$ to $TQ$
        $s \leftarrow s + 1$
    **if** $TQ_1[i].start == start$ **then**
        $i \leftarrow i + 1$
    **if** $TQ_2[j].start == start$ **then**
        $j \leftarrow j + 1$
**return** $TQ, s$

not every episode is stored in the lattice, it would otherwise grow impractical. Only the length-1 episodes and the frequent episodes are maintained in the lattice. Length-1 episodes are kept even if they are not frequent in order to keep the necessary information to build longer episodes if and when they become frequent. Each node retains the time queue of the corresponding episode and the Gaussian Mixture Model description that best fits the episode.

The parents of a node (located at depth $d$) correspond to its sub-episodes of length $d-1$, and its children to its super-episodes of length $d+1$. The edges linking two episodes are indexed on the only event label that is present in the child episode but not in the parent: this facilitates the navigation within the episode lattice.

In spite of its possibly big edge count, the lattice structure was chosen over the standard prefix tree because it allows faster episode retrieval and update. The episode lattice corresponding to example 4.1 is given in figure 4.5 (built with minimal support threshold $S_{min} = 3$, maximal occurrence duration

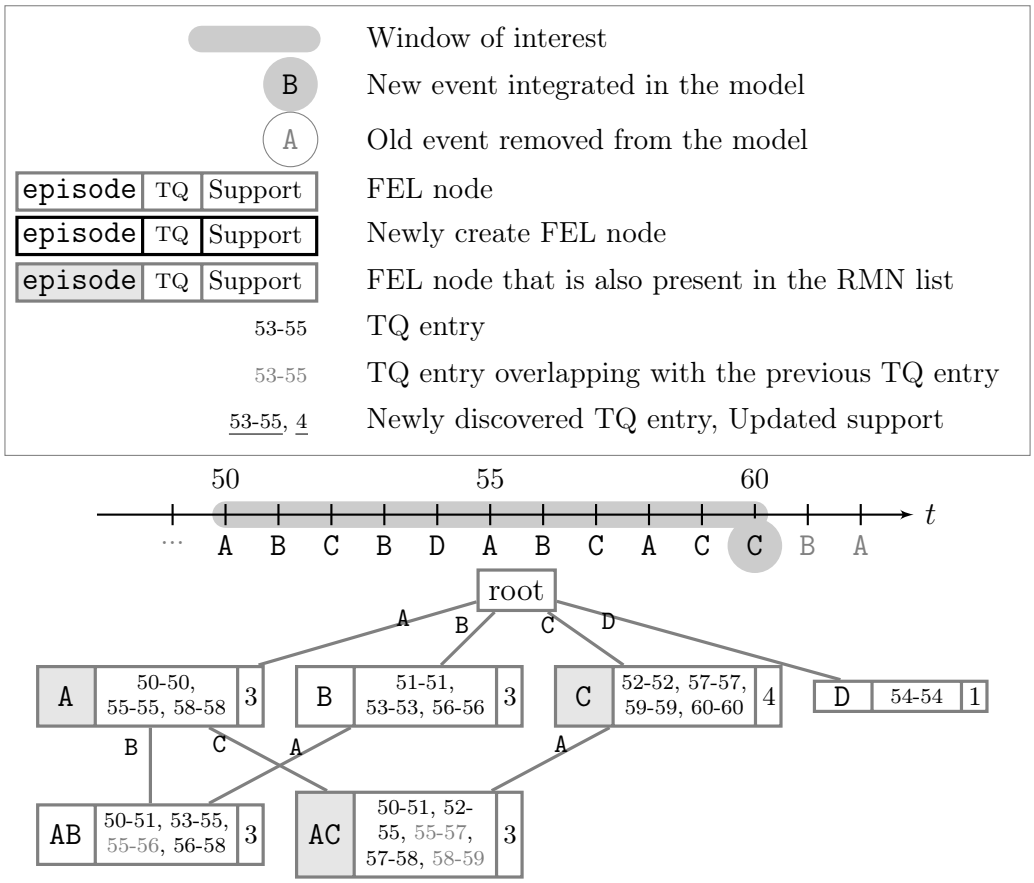



Figure 4.5 – Lattice corresponding to the example stream figure 4.2, when (C, 60) is the last seen event. The top box describes the conventions used to highlight the changes induced by the updates. The same conventions are used in figures 4.6 and 4.7

$T_{ep} = 3$, window length $T_W = 10$).

### 4.3.3 Recently modified nodes

As stated in property 4.5, the recently observed occurrences of an episode may be extended with a new incoming event to form an occurrence of a longer episode. Here, *recently observed* means that the latest occurrence started less than $T_{ep}$ before the timestamp of the current event. This gives a particular importance to the lattice nodes that were recently modified (RMN – recently modified nodes). sxED thus stores the recently modified nodes in a layered

**Algorithm 4** Update of the Frequent Episode Lattice when a new event is recorded

**Input:** new event $(e, t)$, FEL, RMN-list

```
1: if label e is not in the first level of the FEL then
2:     Create a node for episode {e} and link it to the FEL root
3: Update the time queue of episode {e}
4: if {e} is frequent then
5:     Add it to the RMN-list
6:     for all depth d in FEL do
7:         for all node N_E (characterizing episode E) in RMN-list[d] do
8:             Check whether there is a new MO of E ∪ {e} (algorithm 5)
9: Remove outdated information
```

collection of lists if they are frequent (the RMN at depth 1, the RMN at depth 2, etc.) subsequently called the *RMN-list*.

For example, the RMN-list associated with the lattice figure 4.5 is:

$$RMN = [\ [\ \{\texttt{A}\}, \{\texttt{C}\}\ ], [\ \{\texttt{A}, \texttt{C}\}\ ]\ ]$$

$\{\texttt{D}\}$ is not part of the RMN-list, since the episode is not frequent enough (its support is 1, when $S_{min} = 3$). Episodes $\{\texttt{B}\}$ and $\{\texttt{A}, \texttt{B}\}$ where not observed recently enough (last observation at timestamp 56, i.e. 4 clock ticks before the current event, when $T_{ep} = 3$).

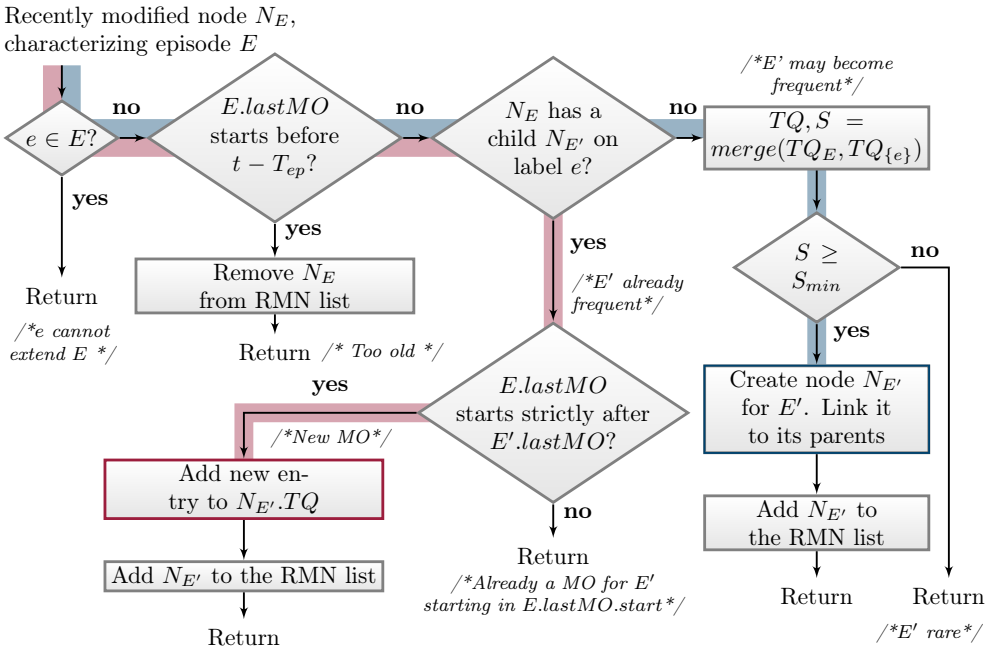
### 4.3.4 Lattice update

#### 4.3.4.1 Update with a new event

When a new event $(e, t)$ arrives, the lattice update follows algorithm 4: the episode of length 1 $\{e\}$ is first updated with its new time queue entry $[t, t]$ (property 4.1). The event can also be part of a new minimal occurrence for a longer episode $E' = E \cup \{e\}$ where $E$ is an episode that was recently observed (property 4.5). The RMN-list is thus traversed and its nodes are candidates for the construction of longer episodes.

For each node $N_E$ in the RMN-list, $E' = E \cup \{e\}$ is investigated thanks to the procedure detailed in algorithm 5. Algorithm 5 updates the Frequent Episode Lattice and the RMN-list by taking advantage of the time queue properties 4.1–4.5. Two critical paths are highlighted:

**Algorithm 5** RMN-based update when a new event $(e, t)$ arrives




- The red path is followed when there is a new minimal occurrence (MO) for the already frequent episode $E' = E \cup \{e\}$,
- The blue path is followed when $E'$ becomes frequent thanks to the newly observed minimal occurrence. When $E'$ becomes frequent, a new node $N_{E'}$ is created, and is linked to its parents in the lattice. The parents are the nodes describing the episodes $E' \backslash \{e'\}$ for each $e' \in E$, and are accessible via $N_E.parent(e').child(e)$, where $N_E$ is the node for the known subset $E$. Since the RMN list is layered, and explored by increasing node depth, $N_E.parent(e').child(e)$ is always created and properly linked in the Frequent Episode Lattice before $N_{E'}$ tries to access it.

#### 4.3.4.2 Removal of outdated information

Events older than $T_W$ are outdated, and their influence in the Frequent Episode Lattice needs to be removed. The time queue construction makes it so that its entries are ordered by increasing start timestamp: the entries that need to be removed are thus at the beginning of the time queue in each node. If a node does not contain outdated time queue entries, then

its children (and thus the complete sub-lattice of its super-episodes) do not have outdated entries in their time queues. The Frequent Episode Lattice can thus be traversed from the root using a breadth-first search algorithm, where nodes are investigated and updated only if at least one of their parents presents outdated occurrences.

If an episode becomes rare, all of its super-episodes become rare too (property 4.3). This node and its children are thus removed from the Frequent Episode Lattice (except the nodes at depth 1, which are still needed for when they become frequent again).

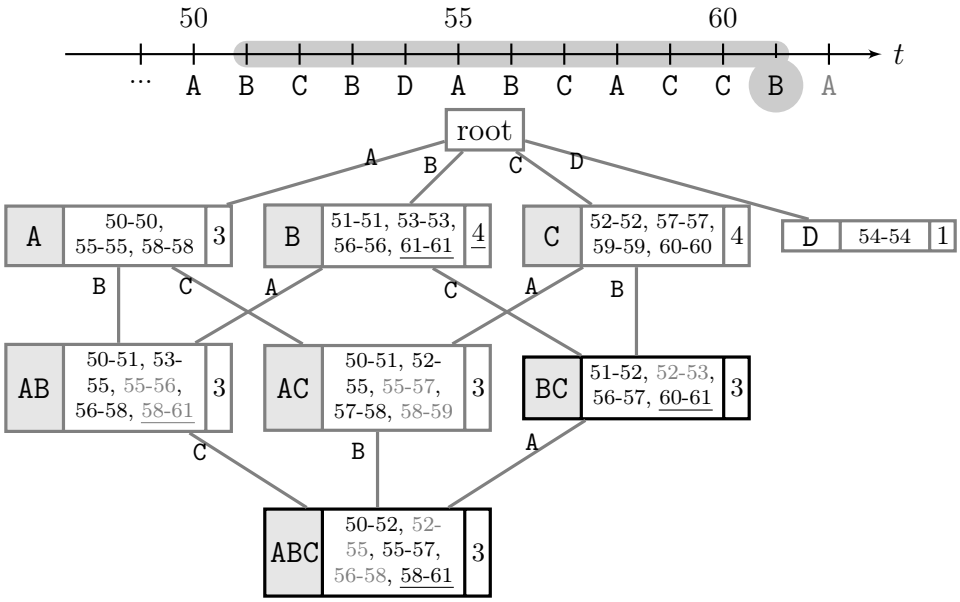
#### 4.3.4.3 Illustration on example 4.1

The update process of the Frequent Episode Lattice is illustrated twice on the events of example 4.1, with first the arrival of a new event (B, 61) in figure 4.6 and this of (A, 62) in figure 4.7. With (B, 61), episodes {B, C} and {A, B, C} become frequent. Their nodes are thus created and inserted in the Frequent Episode Lattice, and in the RMN list. The arrival of (B, 61) also makes (A, 50) outdated. Episode {A} looses an occurrence and its support becomes 2. Since {A} is rare, so are all the episodes in the sub-lattice starting from node {A}. They are thus removed from the lattice. {A} is moreover removed from the RMN list.

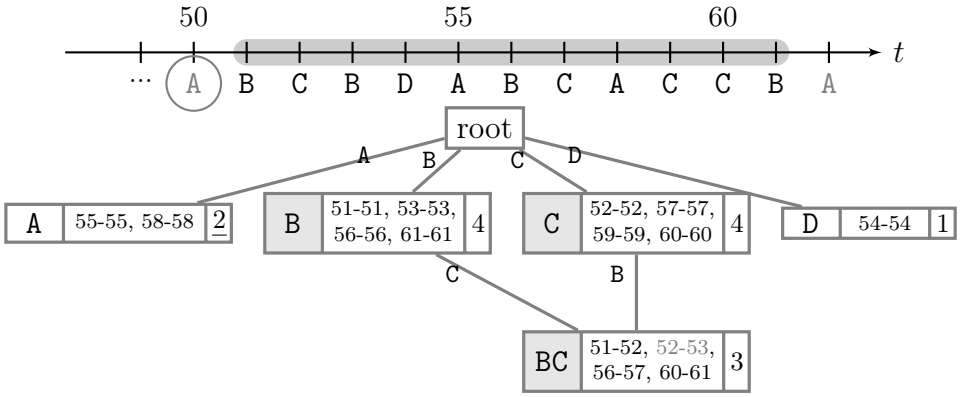
The arrival of (A, 62) makes {A} become frequent again. The nodes in the RMN list are thus investigated to check whether they can be extended with label A. The episodes {A, B}, {A, C} and {A, B, C} are thus introduced in the lattice again. The removal of outdated event (B, 52) updates the nodes for episodes {B} and {B, C} but does not change the structure of the lattice.

### 4.3.5 Periodicity discovery

The periodicity of an episode is described thanks to a Gaussian Mixture Model. Each node in the Frequent Episode Lattice is associated with a Gaussian Mixture Model describing the periodicity of the episode, which is updated when new minimal occurrence are observed or occurrences removed. Usually, a Gaussian Mixture Model is trained with the Expectation-Maximization algorithm (Dempster et al., 1977) (EM), whose behavior was described in the previous chapter, section 3.4.2: the characteristics of the

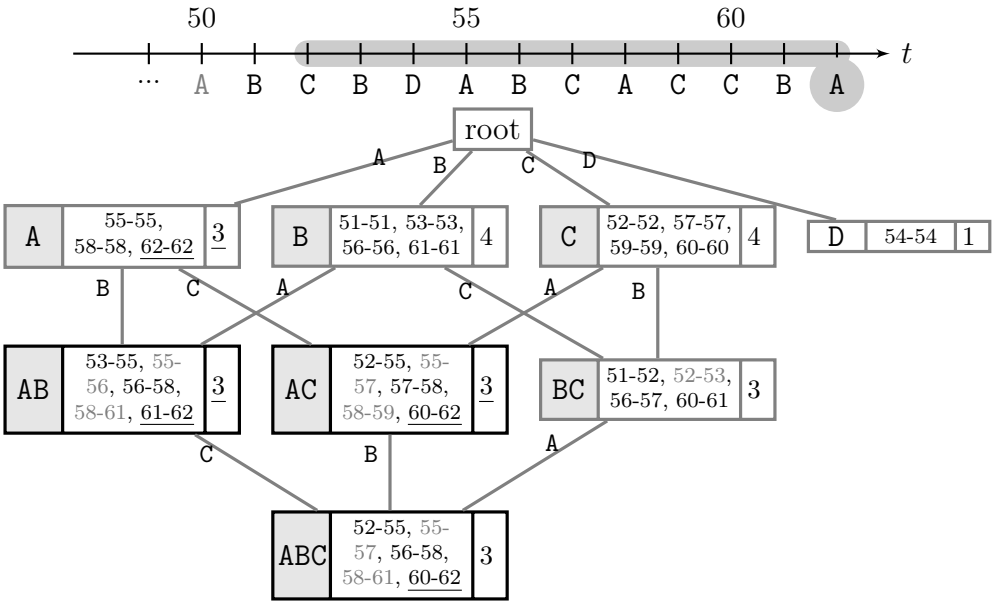



(a) Inclusion of event (B, 61) in the Frequent Episode Lattice: episodes {B, C} and {A, B, C} become frequent.

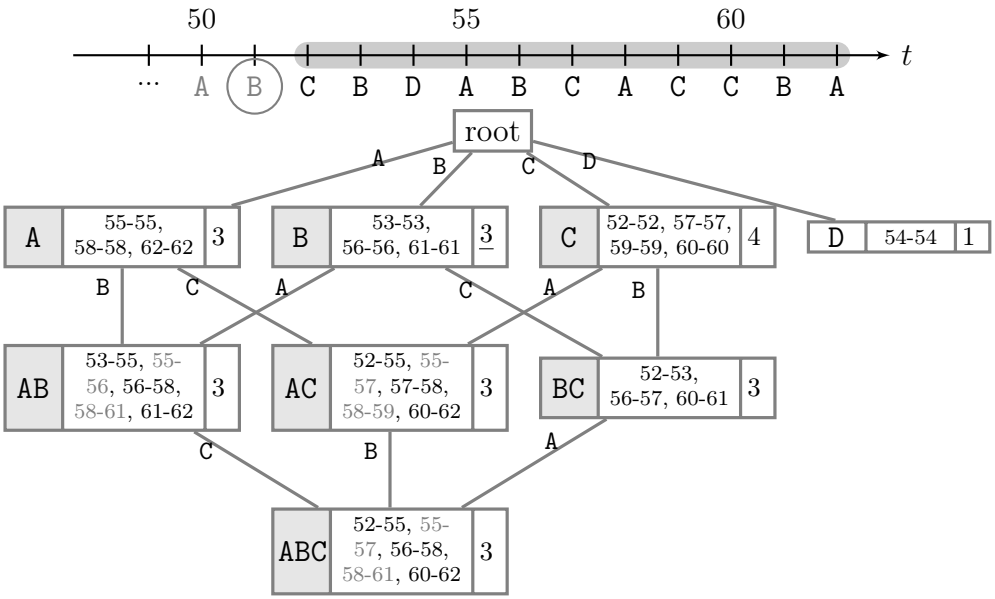



(b) Removing outdated event (A, 50): episode {A} becomes rare, and thus the episodes containing event label A are rare too, and their nodes are removed. See the representation conventions in the legend of figure 4.5

Figure 4.6 – Update with event (B, 61)




(a) Inclusion of event (A, 62) in the Frequent Episode Lattice: {A} becomes frequent again, as well as {A, B}, {A, C} and {A, B, C}




(b) Removing outdated event (B, 52): {B} loses a minimal occurrence but remains frequent, {B, C} loses a minimal occurrence as well, but its support is not modified. See the representation conventions in the legend of figure 4.5

Figure 4.7 – Update with event (A, 62)

Gaussian component are iteratively tweaked to maximize their adherence to the data.

The number of components in the Gaussian Mixture Model is traditionally evaluated by trial and error, using the AIC or BIC information criteria. This requires the training and maintenance of several mixture models. Moreover, since streaming data may be non-stationary, the number of components may evolve, as well as their characteristics. We here extend Expectation-Maximization with heuristics for the addition, removal and merging of components. Algorithm 6 presents the general workflow for the periodicity update. In particular:

- When a new minimal occurrence, starting at timestamp $t$ is detected for the episode, the position of the timestamp in the period $t_r = t$ modulo *period* is computed. If $t_r$ does not match any of the existing components, i.e. for each component $(\mu, \sigma)$, $|t_r - \mu| > \sigma$, a new component is added.

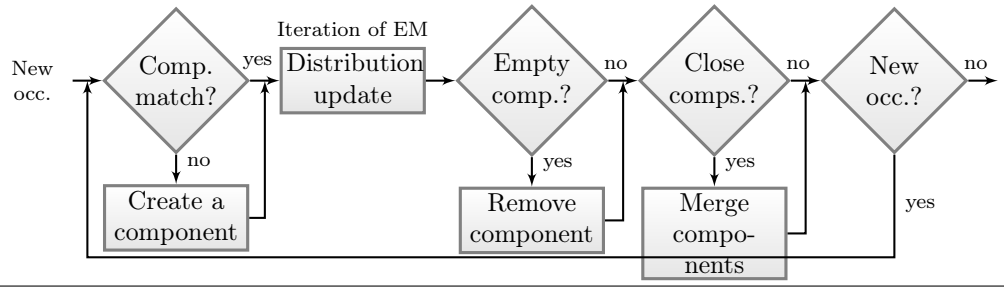
- When outdated data is removed from the sliding window, some components may lose their importance. When a component is not enough represented among the non-overlapping minimal occurrences in the window, it is removed from the Gaussian Mixture Model.

- Finally, when two components $(\mu_1, \sigma_1), (\mu_2, \sigma_2)$ become close to one another, i.e if $|\mu_1 - \mu_2| < a * (\sigma_1 + \sigma_2)$ (with $a = 1.5$ in the experiments), the two components are merged.

- In the general case, Gaussian Mixture Model updates do not change much the model. Thus, when the number of components does not change, a single Expectation-Maximization iteration is necessary to update the characteristics of the components.

The interest of this approach was evaluated on synthetic data, following known mixture of Gaussian models evolving with time. The heuristics allow the detection of the main trends in the data: emergence of new components, loss of old and rare components, shifting in the characteristics of the components.

**Algorithm 6** Overview of the periodicity update ("comp" stands for Gaussian Mixture Model component)




### 4.3.6 General workflow

Algorithm 7 summarizes the main steps of sxED: when a new event is recorded, the Frequent Episode Lattice is updated thanks to the frequent episode discovery algorithm. Whenever a node is created or updated, its periodicity is updated, and it is added in the RMN list. When the Frequent Episode Lattice is up-to-date, the outdated events are removed.

Table 4.2 summarizes the parameters allowing a tuning of the overall algorithm behavior.

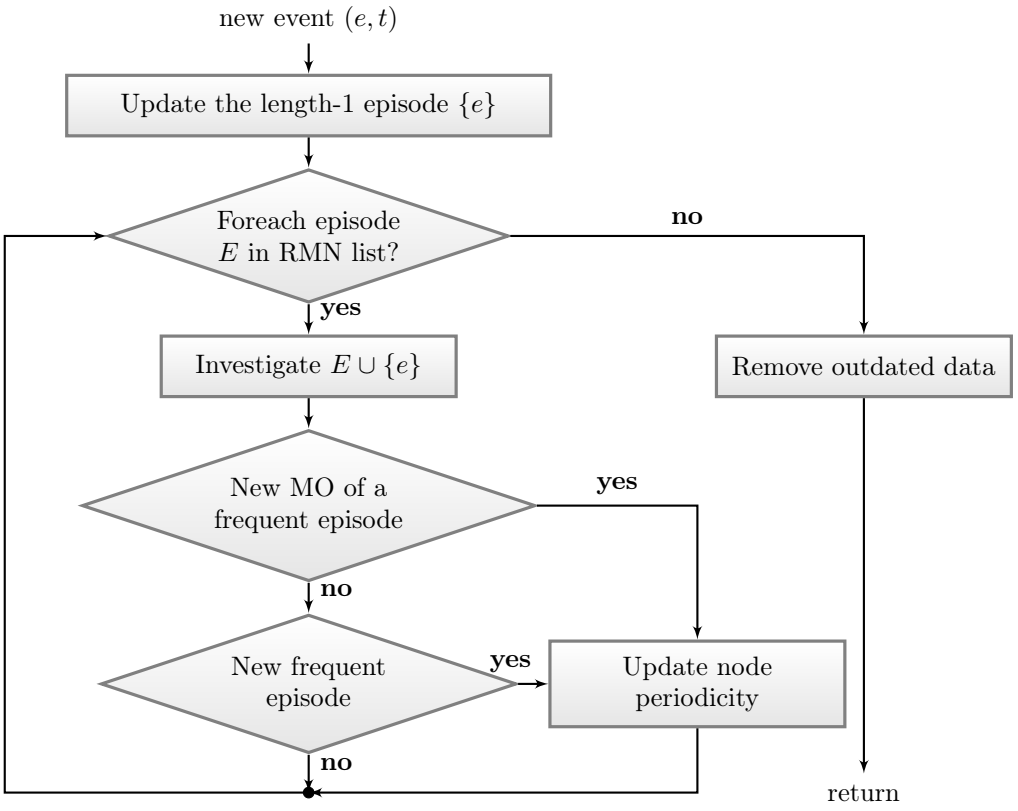
## 4.4 Experimentation

In this section, we evaluate sxED on its ability to disover interesting frequent periodic episodes in three activity datasets (section 4.4.1), and on its scalability (memory usage and runtime, see section 4.4.2). A qualitative comparison between sxED and xED is also proposed in section 4.4.3.

A prototype was implemented in Python. It was also instrumented to record the episodes and lattice updates. The instrumentation slows down the experimentations: the execution times given are over-estimated.

We here describe the experiments on ambient assisted living datasets. An experiment on data coming from an online video game was also conducted, and is available in appendix C.

**Algorithm 7** General workflow for sxED

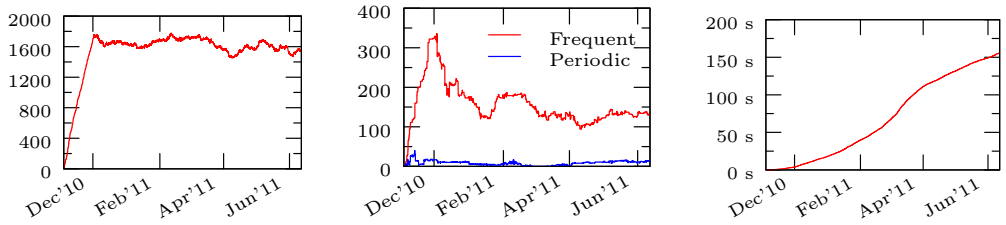



### 4.4.1 Experiments with three activity datasets

The CASAS project (Cook et al., 2013a) uses home automation devices to improve aging at home. Over the years, they have collected and published several datasets. We present here our experimentations on the Aruba, Cairo and Twor datasets[1]. These datasets record the daily living patterns over a period ranging from 2 months (Cairo and Twor datasets) to 6 months (Aruba dataset), using motion detectors. The obtained information was annotated with activities labels (such as *Sleeping*, *Housekeeping*, etc.). These annotations are our events. Table 4.3 summarizes the characteristics of the datasets.

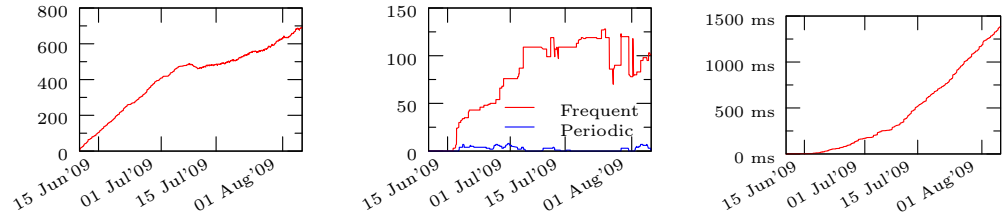
The datasets were processed using a period of one day, a window $T_W$ of 4 weeks, a minimal support $S_{min}$ of 8, a maximal episode duration $T_{ep}$ of

[1] available online http://wsucasas.wordpress.com/datasets/, last consulted on October 5th, 2015



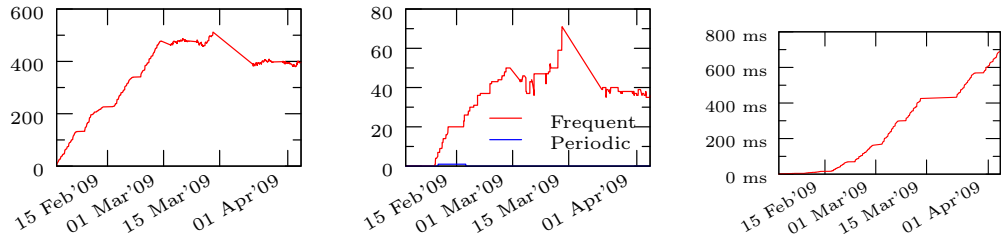

(a) Event count in the window (b) Interesting episode count (c) Cumulative execution time

Figure 4.8 – Execution log for the Aruba dataset




(a) Event count in the window (b) Interesting episode count (c) Cumulative execution time

Figure 4.9 – Execution log for the Cairo dataset




(a) Event count in the window (b) Interesting episode count (c) Cumulative execution time

Figure 4.10 – Execution log for the Twor dataset

Table 4.2 – Overview of the parameters available to tune the online periodic episode discovery algorithm

| **Parameter** | **Description** | **Area of influence** | **Value** |
|---|---|---|---|
| Maximal occurrence duration $T_{ep}$ | Maximal time interval between events in an episode occurrence | FEL | $> 0$ (e.g. 30 min) |
| Minimal support $S_{min}$ | Minimal number of occurrences for an episode to be considered as frequent. Only frequent episodes are in the FEL. | FEL | $> 0$ (e.g. 1 occurrence per week) |
| Window length $T_W$ | Duration of the window. Only the events that occurred less than $T_W$ ago influence the results. | FEL, Periodicity description | $> 0$ (e.g. 4 weeks) |
| Period | Used for periodicity computation. | Periodicity description | $> 0$ (usually 1 day, 1 week) |
| Minimal accuracy $A_{min}$ | Minimal accuracy for a periodic description to be considered as interesting. | Periodicity description | $0 - 100\%$ (e.g. 60%) |

30 minutes, and an accuracy threshold of 70%. The parameter setting was reinforced by a descriptive analysis of the data (e.g., it showed that most activities last less than 30 minutes). The results obtained throughout the course of the execution are given in figures 4.8 for the Aruba dataset, 4.9 for the Cairo dataset and 4.10 for the Twor dataset.

During the first 4 weeks, the sliding window fills with the incoming events (figures 4.8a, 4.9a and 4.10a), and the first frequent and periodic episodes appear (figures 4.8b, 4.9b and 4.10b). Since the datasets are different, the

Table 4.3 – Characteristics of the ambient assisted living datasets

| | **Aruba** | **Cairo** | **Twor** |
|---|---|---|---|
| **Recorded** | Nov $4^{\text{th}}$, 2010 – June $6^{\text{th}}$, 2011 | June $10^{\text{th}}$ – Aug $5^{\text{th}}$, 2009 | Feb $2^{\text{nd}}$ – Apr $4^{\text{th}}$, 2009 |
| **Duration** | 220 days | 57 days | 61 days |
| **# Labels** | 22 | 26 | 32 |
| **# Events** | 12 954 | 1 200 | 1 004 |

observations are different too. Here are some remarks:

- The execution times (figures 4.8c, 4.9c and 4.10c) shows the scalability of the approach for this kind of application. The longer execution time for the CASAS dataset comes from the greater amount of events in this dataset: the window is more dense, and the data structures are updated more.

- In the Aruba dataset, the size of the window remains quite stable after its initial filling (figure 4.8a). That is not true for the Cairo (4.9a) and Twor (4.10a) datasets, hence the abrupt jumps in the episode counts. In particular for the Twor dataset, there is a gap without event recording between March 13$^{\text{th}}$ and March 23$^{\text{rd}}$.

- The number of frequent periodic episodes is much smaller than the number of frequent episodes. This makes the habits (i.e. the periodic episodes) easier to track and analyze by the caregiver or physician.

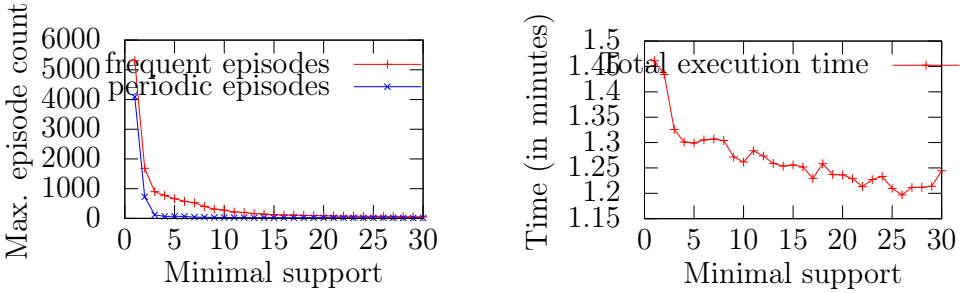
**Examples of periodic patterns** The contents of the FEL in the last window is investigated for the Aruba dataset. Some of periodic episodes with the highest accuracy $A$ are:

- {`Sleeping end`}: 50 non-overlapping minimal occurrences, 1 component, $\mu$ = 6:00, $\sigma$ = 2 hours, $A$ = 100%

- {`Enter_Home begin`, `Enter_Home end`}: 61 non-overlapping minimal occurrences, 1 component, $\mu$ =14:00, $\sigma$ =3 hours, $A$ = 88%

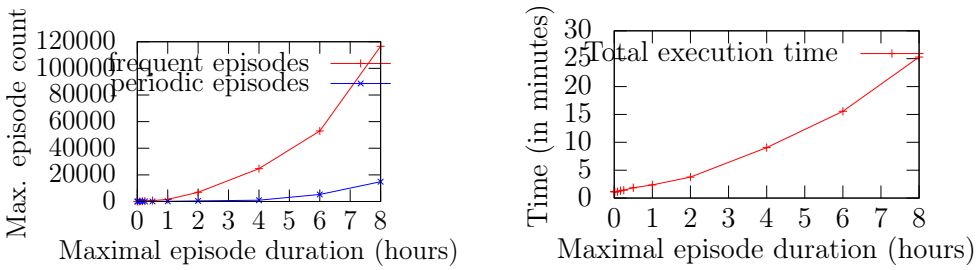
- {`Sleeping end`, `Meal_Preparation begin`, `Meal_Preparation end`, `Relax begin`}: 26 non-overlapping minimal occurrences, 1 component, $\mu$ =6:00, $\sigma$ =1.45 hours, $A$ = 82%

These patterns can be interpreted as habits: the person woke up every morning around 6:00, and also had breakfast in 82% of the mornings. The third episode describes a movement pattern: the inhabitant usually goes out of home at some time (it is another episode), and comes back in the early afternoon.



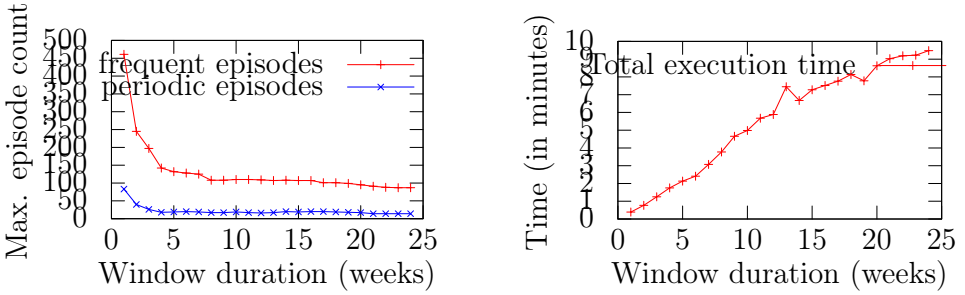

(a) Size of the FEL (b) Cumulative execution time

Figure 4.11 – Influence of the minimal support threshold




(a) Size of the FEL (b) Cumulative execution time

Figure 4.12 – Influence of the maximal occurrence duration




(a) Cumulative execution time (b) Size of the FEL

Figure 4.13 – Influence of the window duration

### 4.4.2 Parameter influence

In order to assess the influence of the parametrization on the scalability of sxED, the CASAS Aruba dataset was mined using different settings. For each setting, the maximal size of the FEL (maximal number of frequent episodes and maximal number of frequent periodic episodes) and the total execution time was recorded. Figure 4.11 presents the influence of the minimal support $S_{min}$, which we set to values ranging from 2 to 30, with a step of 1. The other parameters remained constant ($T_{ep}$ = 30min, $T_W$ = 3 weeks). Figure 4.12 draws the influence of the maximal occurrence duration $T_{ep}$ (ranging from 1s to 8hours, $S_{min} = 5$ and $T_W = 4$ weeks). Figure 4.13 shows the influence of the window duration $T_W$ (ranging from 1 to 24 weeks, with $T_{ep}$ = 30min and $S_{min}$ = 4 occurrences per week, i.e. $S_{min} = 4$ for $T_W = 1$ week and $minimalsupport = 96$ for $T_W = 24$ weeks).

In particular, it shows that the execution time remains reasonable and scalable. The maximal episode duration parameter (figure 4.12) remains however necessary to prevent an explosion of the complexity of the FEL. For habit monitoring, 30 minutes is quite reasonable.

### 4.4.3 Qualitative comparison with xED on KA dataset

In order to compare sxED with its static dataset counter part, xED, the periodic episodes discovered by xED and sxED on the KA dataset are compared. In the previous chapter, table 3.4 lists the periodic episodes discovered by xED. Table 4.4 lists all the periodic episodes discovered by sxED on the same dataset, and with a similar setting: same episode maximal duration $T_{ep}$ = 30 minutes, same support threshold $S_{min} = 3$, and a window $T_W = 4$ weeks in order to cover the complete dataset. Since some of the most interesting episodes according to xED had an accuracy as low as 60%, the minimal accuracy threshold was lowered to $A_{min} = 60\%$ for sxED as well.

In this configuration, sxED discovers 16 interesting episodes. These episodes are very similar to those xED discovers: they were either discovered by xED, or are sub-episodes of episodes that were discovered by xED. The episodes xED discovered but that were missed by sxED are episodes whose validity interval was really short (#7 and #10 {brush teeth start, brush teeth end}, valid only from Feb 25$^{th}$ to March 7$^{th}$ and from March 17$^{th}$ to 21$^{st}$; #8 {use toilet start, use toilet end, prepare Dinner start}, valid only from March 5$^{th}$

Table 4.4 – Periodic episodes in the KA dataset in the window spanning over the whole dataset

| Episode | S | Periodicity | A |
|---|---|---|---|
| use toilet start | 109 | (6:03:49, 3:21:53), (19:57:13, 2:16:07) | 82 % |
| use toilet end | 109 | (6:05:36, 3:21:57), (19:58:57, 2:16:05) | 82 % |
| use toilet start, use toilet end | 109 | (0:34:34, 1:33:18), (7:49:31, 1:38:53), (20:11:24, 2:06:54) | 77 % |
| take shower start | 22 | (9:29:23, 2:02:41) | 76 % |
| take shower end | 22 | (9:39:03, 2:01:11) | 76 % |
| take shower start, take shower end | 22 | (9:29:23, 2:02:41) | 76 % |
| go to bed end | 22 | (7:57:07, 0:42:53) | 72 % |
| take shower end, leave house start | 19 | (9:27:37, 1:56:10) | 68 % |
| go to bed end, use toilet start | 21 | (7:56:33, 0:41:33) | 68 % |
| take shower start, leave house start | 18 | (9:23:30, 1:59:07) | 64 % |
| take shower start, take shower end, leave house start | 18 | (9:23:30, 1:59:07) | 64 % |
| prepare Breakfast end | 20 | (8:27:10, 0:39:28) | 64 % |
| go to bed end, use toilet end | 20 | (8:00:01, 0:39:37) | 64 % |
| go to bed end, use toilet start, use toilet end | 20 | (7:59:57, 0:39:30) | 64 % |
| prepare Breakfast start | 20 | (8:24:23, 0:39:05) | 60 % |
| prepare Breakfast start, prepare Breakfast end | 20 | (8:24:23, 0:39:05) | 60 % |

to March 11$^{\text{th}}$; and #9 {get drink start, get drink end}, valid from Feb 25$^{\text{th}}$ to 28$^{\text{th}}$). Episode #11 could obviously not be discovered by sxED, since it does not iterate the episode mining process several time.

The periodicity descriptions are also very similar (similar average occurrence time for the morning routines), which is to be expected since the mixture models were fit using the same Expectation Maximization methodology. The main differences are observed for the episodes where the number of components is different (use toilet start, use toilet end} and its sub episodes). sxED thus performs as well as xED, while also adapting to concept drifts and continuously arriving data.

In terms of execution time, the incremental habit discovery by sxED gave results much faster than xED (cumulative execution time of 0.7s for sxED,

18s for the 6 iterations of xED, that is to say an average of 3s per iteration). The better performance for sxED were expected, but should be put in perspective, as neither of the implementations of xED and sxED were thoroughly optimized: it was not the main objective. Although both prototype implementations for xED and sxED could be better optimized, this shows the better performances of sxED.

## 4.5 Conclusion

Behavior pattern (episode) mining over event sequences is an important data mining problem, with many applications, in particular for ambient assisted living. Several frequent episode mining algorithms have been proposed for both static data and data streams. But while periodicity was also shown to be an interesting characteristic for the study of behaviors, very few algorithm have addressed both frequent and periodic patterns, in a streaming context. In this chapter, we have thus proposed sxED, an algorithm to mine frequent periodic episodes in data streams.

Future work on this topic includes the automatic determination of the period to consider. Due to the use of time queues, sxED does not scale well when the maximal episode duration $T_{ep}$ threshold is high, or when the event density increases. It would thus be very interesting to devise new ways to compute periodicity and to evaluate its accuracy, that would not require the occurrences timestamps. In particular, approximate support counting and decay factors (instead of sliding windows) seem to be promising alternatives. Solutions involving less parameters would also be easier to tune, especially for the supervisor, who is in general a physician or a caregiver, and not an algorithm expert.

# Chapter 5

# Top-$k$ regular episodes

## 5.1 Introduction

The description of periodicity via Gaussian Mixture Models is a powerful descriptive tool. It provides the supervisor with a detailed description of the episodes, their periodicity and its components, a description of their variability and of how often they occur, thanks to the accuracy measure. However, the computation, maintenance and analysis of such periodicity descriptions requires the storing of the occurrence times of the episodes of interest, which may be resource-consuming. They can moreover be "cheated": an arbitrarily big standard deviation will tend to produce high-accuracy periodicity descriptions.

We thus investigate another popular periodicity measure: *regularity*, which studies the time intervals between the consecutive occurrences of an episode. Moreover, since the setting of appropriate thresholds may be complicated and overwhelming, we adopt a top-$k$ approach: instead of setting a frequency or a regularity threshold to separate interesting episodes from the non-interesting ones, the supervisor sets the number $k$ of desired results.

Thus, we propose TKRES (**T**op-$k$ **R**egular **E**pisodes in **S**treams) for the discovery and update of the top-$k$ most regular episodes in an event stream.

**Chapter outline** The rest of the chapter is organized as follows: section 5.2 reviews some of the associated literature for regular pattern mining.

Section 5.3 presents the formalisms, defines the problem we address, and proposes a formalism for the discovery of the top-$k$ regular episodes. Section 5.4 presents the TKRES algorithm, and section 5.5 present the performance evaluation of TKRES. Finally, section 5.6 concludes and presents lines of study for future work.

## 5.2 Literature Review: regularity

Regularity was introduced in Tanbeer et al. (2008). It was originately defined for itemset mining in transactional databases, and considers the gaps between the occurrences of an itemset. It was also sometimes called "periodicity", which we do not do, in order to avoid confusions with the periodicity definitions used in chapters 3 and 4.

Formally, let $I = \{i_1, \cdots, i_n\}$ be a set of $n$ items and $TDB = \{t_1, ..., t_m\}$ a transactional database. Each $t_i \in TDB$ is a transaction, identified by a transaction identifier $tid_i$, which contains an itemset $X_i \subseteq I$ ($X_i$ subset of $I$). Let $X$ be an itemset. The transaction $t_i$ *supports* $X$ if $X \subseteq X_i$. The identifiers of the transactions that support $X$ are stored in a list $T^X = \{tid_{i_1}, \cdots, tid_{i_k}\}$ (with $1 \leq i_1 < \cdots < i_k \leq m$). The *support* of $X$ is the length of $T^X$: $k$ (there are $k$ transactions identifiers in $T^X$).

**Regularity** Let $X$ be an itemset, and $T^X = \{tid_{i_1}, \cdots, tid_{i_k}\}$ the identifiers of the transactions that support $X$. The regularity of $X$ is then the maximal gap between two consecutive transactions supporting $X$. Two special cases are also considered: the gap between the beginning of the dataset and the first occurrence, as well as this between the last occurrence of $X$ and the end of the dataset. The regularity is thus the maximal value of:

- $tid_{i_1} - tid_1$ (which corresponds to the number of transactions from the beginning of the data to the first occurrence of $X$)
- $\max_{1 \leq j < k} tid_{i_{j+1}} - tid_{i_j}$ (which corresponds to the maximal number of transactions between two consecutive occurrences of itemset $X$)
- $tid_m - tid_{i_k}$ (which corresponds to the number of transactions between the occurrence of $X$ and the end of the data)

An itemset is then said to be regular if its regularity is lower than a user-defined threshold. One can notice that an itemset is more regular if its regularity value is small.

Moreover, if $Y$ is a sub-itemset of $X$, then it is more regular than $X$ (downward closure property (Tanbeer et al., 2008)). This provides a stop criterion when mining regular itemsets: if an itemset is not regular, then its super-itemsets cannot be regular either.

Regularity has been extensively studied, especially by Tanbeer et al. and Amphawan et al.. In particular, regularity was coupled with frequency for the discovery of frequent-regular itemsets (Tanbeer et al., 2009; Surana et al., 2011). It has been used in top-$k$ approaches (Amphawan et al., 2011; Amphawan and Lenca, 2013), and for the discovery of closed itemsets (Amphawan and Lenca, 2015). The initial algorithms have also been extended to allow an incremental discovery (Tanbeer et al., 2010b), or for the mining of data streams (Tanbeer et al., 2010a).

However, this definition for regularity, and the algorithms used for the discovery of regular itemsets need to be adapted in order to mining event sequences and event streams. We thus propose in section 5.3 an extension of regularity for episode mining in sensor streams.

## 5.3 Problem definition

An event stream is a potientially infinite sequence of events, ordered by their timestamp. The events are gathered in batches of fixed duration (e.g.: one day), and the stream of batches $DS = \langle B_1, \cdots, B_n, \cdots \rangle$ is processed using a sliding window: only the last $m$ batches of data are considered.

TKRES searches for the top-$k$ most regular episodes (definition 5.1) in the recent past, composed of the last $m$ batches of data. Whenever a new batch of data arrives, the interesting episode list is updated.

**Definition 5.1** (Top-$k$ regular episodes)**.** An episode $E$ is a top-$k$ regular episode if there are no more than $k-1$ episodes that are more regular than $E$.

Initially defined in the context of itemset mining in transactional data, regularity is not fit to describe episodes in event streams. We here extend the

definition of regularity to cover episode mining as well. Instead of defining the regularity as the number of transactions separating two transactions that support the itemset, regularity becomes the time between consecutive non-overlapping minimal occurrences (definition 5.2). The maximal duration of the occurrences is bounded by a threshold $T_{ep}$, which corresponds exactly to the support threshold used in xED (chapter 3) and sxED (chapter 4). As in sxED, the occurrences of the episodes are managed thanks to their time queues (TQ, see definition 4.6).

**Definition 5.2** (Regularity)**.** Let $E$ be an episode, and $TQ^{NOMO}$ its time queue restrained to the non-overlapping occurrence intervals. Let $l$ be the length of $TQ^{NOMO}$. Each entry $t$ in $TQ^{NOMO}$ is composed of a start and end timestamp, noted $t.start$ and $t.end$. The regularity of episode $E$ is:

$$r^E = \max_{0 \leq i \leq l}(r_i) \qquad \text{where}$$

- $r_0 = t_1.end - t_{sw}$, with $t_1$ the first occurrence in $TQ^{NOMO}$ and $t_{sw}$ the start time of the window (first regularity);
- $r_i = t_{i+1}.end - t_i.start$ for $1 \leq i < l$, where $t_i$ and $t_{i+1}$ are two consecutive occurrences in $TQ^{NOMO}$;
- $r_l = t_{ew} - t_l.start$, with $t_l$ the last occurrence $TQ^{NOMO}$ and $t_{ew}$ the last timestamp of the window.

**Downward closure** Similarly to the original regularity definition for itemset mining, an episode $E$ is more regular than another episode $E'$ if $r^E < r^{E'}$. This definition for regularity also maintains the downward closure property: an episode is always as regular as or more regular than its super-episodes. This means that if an episode is not a top-$k$ regular episode, then none of its super-episodes may be a top-$k$ episode either.

Put in simple words, regularity measures the maximal time gap between the *start* of an occurrence and the *end* of the next occurrence (*start-to-end*). Other definitions like *end-to-start*, *start-to-start* or *end-to-end* would not satisfy the downward closure property.

**Problem statement** With the user-given set of parameters (table 5.1): a number of desired interesting episodes $k$, a batch duration $|B|$, a number $m$

Table 5.1 – Parameters for the tuning of TKRES

| Parameter | Description | Area of influence |
|---|---|---|
| $k$ | Number of episodes in output. Only the $k$ most regular episodes are returned. | Depth of the $k$-tree, length of the top-$k$ list |
| $m$ Number of batches in the sliding window | The occurrences in the last $m$ batches are used to build episodes and compute their regularity. Older data is discarded. | Defines how long the occurrences remain valid |
| $T_{ep}$ Maximal occurrence duration | Occurrences longer than $T_{ep}$ are not considered in the regularity measures. | Time queue construction and update |
| $\|B\|$ batch duration | Duration of a batch | Update frequency |

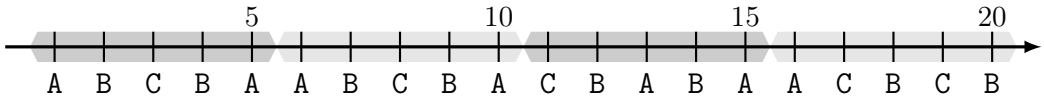



Figure 5.1 – Example event stream, used as a running example in the chapter

of batches in the window, the maximal duration of episodes occurrences $T_{ep}$; we address the problem of mining the top-$k$ regular episodes. That is to say, we discover the $k$ episodes with the lowest regularity values in the window (containing $m$ batches) sliding over a sensor data stream $DS$.

## 5.4 Proposed TKRES algorithm

In this section, we introduce TKRES, an efficient single-pass algorithm for mining the top-$k$ regular episodes in a sensor data stream. The data structures and main steps of the algorithm are illustrated on an example (example 5.1).

**Example 5.1.** Figure 5.1 presents an event stream, containing 20 events, split in 4 batches (batch duration $|B| = 5$). This event stream is processed with the following parameters: $k = 4$, $m = 3$, $T_{ep} = 3$.

TKRES searches the episodes in a sliding window containing $m$ consecutive batches of events, and can be divided in two main steps:

- the *initialization* (section 5.4.2): the dicovery of the top-$k$ regular episodes from the first window (the first $m$ batches of the input stream $B_1$ to $B_m$), and
- the *update with an incoming batch* (section 5.4.3): the update of the knowledge on the top-$k$ regular episodes present in the new window (the previous batches, except the oldest one, plus the new incoming batch).

TKRES uses a list (called top-$k$ list) to maintain the set of top-$k$ regular episodes during data processing, and a tree structure (the $k$-tree) to maintain the occurrence information for the short episodes and the top-$k$ regular episodes. The top-$k$ list is ordered by ascending episode regularity, and is always maintained in order throughout the mining process. The $k$-tree is based on prefix trees.

### 5.4.1 $k$-tree

In order to maintain the necessary information for the episode construction and update, and for the computation of the top-$k$ list, TKRES uses a tree structure (see example figure 5.2). Each node corresponds to an episode and contains:

- a label: the episode corresponding to the node can be retrieved through the list of labels in the path from the node to the root,
- a pointer to its next sibling. These pointers form a linked list of the nodes sharing the same parent, and are used during the construction of longer episodes (merging of two sibling nodes),
- a pointer to its children, indexed on their labels, and a pointer to the first child (head of the sibling linked list),
- a pointer towards the next node located at the same depth, regardless of whether the nodes are siblings or not,
- the time queue of the episode.

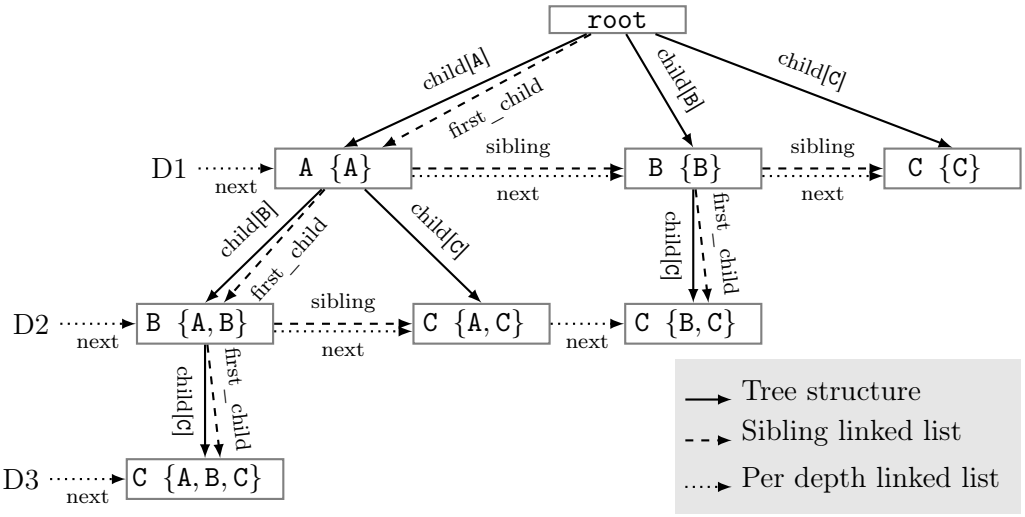



Figure 5.2 – Example of a $k$-tree, if the event alphabet is $\{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$, with a highlight on the navigation possibilities within the tree

When a new node is added in the tree, the mappings are updated: it is linked to its parent, appended to the linked list of its siblings, and appended to the linked list of the nodes located at the same depth. Similarly, when a node is removed from the tree (e.g. because the episode it represents is not in the current window), the linked lists are updated too. Such structure allows a fast tree traversal for the update of the $k$-tree and the retrieval of the top-$k$ episodes.

### 5.4.2 Initial mining

As described in algorithm 8, TKRES first creates nodes in the $k$-tree for all the event labels. The timestamps of the length-1 episodes (the labels) are collected from the events in the batches, and the regularities of these episodes are computed. The $k$ event labels with the lowest regularity are collected and ordered by increasing regularity in the top-$k$ list (lines 1–6). The $k^{th}$ least regular event label gives an upper bound to the maximal regularity a top-$k$ episode may have.

TKRES then builds the $k$-tree to generate longer episodes. It reduces its memory consumption by limiting the depth of the tree to $D_{k-tree}$ (line 7). This threshold is defined as the smallest possible depth enabling to hold $k$

**Algorithm 8** TKRES–initial mining

**Input:** $k$: number of episodes to be discovered,
$m$ batches of sensor data $\langle B_1, \dots B_m \rangle$
**Output:** top-$k$ list, $k$-tree

1: **for all** batch $B_i$ **do**
2: **for all** event $(e_j, t_j)$ in $B_i$ **do**
3: **if** the root has no child node for $e_j$ **then** Create the node
4: Add $\{e_j\}$'s new occurrence $t_j$
5: Compute the regularity of each event label
6: Collect the $k$ labels with the lowest regularity into the sorted top-$k$ list
7: Compute the depth $D_{k-tree}$ of the $k$-tree to be created
8: **for** depth $d = 1$ to $D_{k-tree} - 1$ **do**
9: **for all** node $E_X$ at depth $d$ **do**
10: **for all** $E_Y$ siblings of $E_X$ **do**
11: Merge the time queues of $E_X$ and $E_Y$ to form $E_Z$
12: Create a node for $E_Z$ as the child of $E_X$
13: Compute the regularity $r^{E_Z}$ of $E_Z$
14: **if** $r^{E_Z} < r^k$ **then** Update the top-$k$ list with $E_Z$
15: $d = D_{k-tree}$
16: **while** new episodes of length $d$ get in the top-$k$ list **do**
17: **for all** node $E_X$ at depth $d$ such that $E_X$ is in the top-$k$ list **do**
18: **for all** $E_Y$ siblings of $E_X$ such that $E_Y$ is in the top-$k$ list **do**
19: Merge the time queues of $E_X$ and $E_Y$ to form $E_Z$
20: Compute the regularity of $E_Z$
21: **if** $r^{E_Z} < r^k$ **then**
22: Update the top-$k$ list with $E_Z$
23: Create the node for $E_Z$ (child of $E_X$)
24: $d = d + 1$

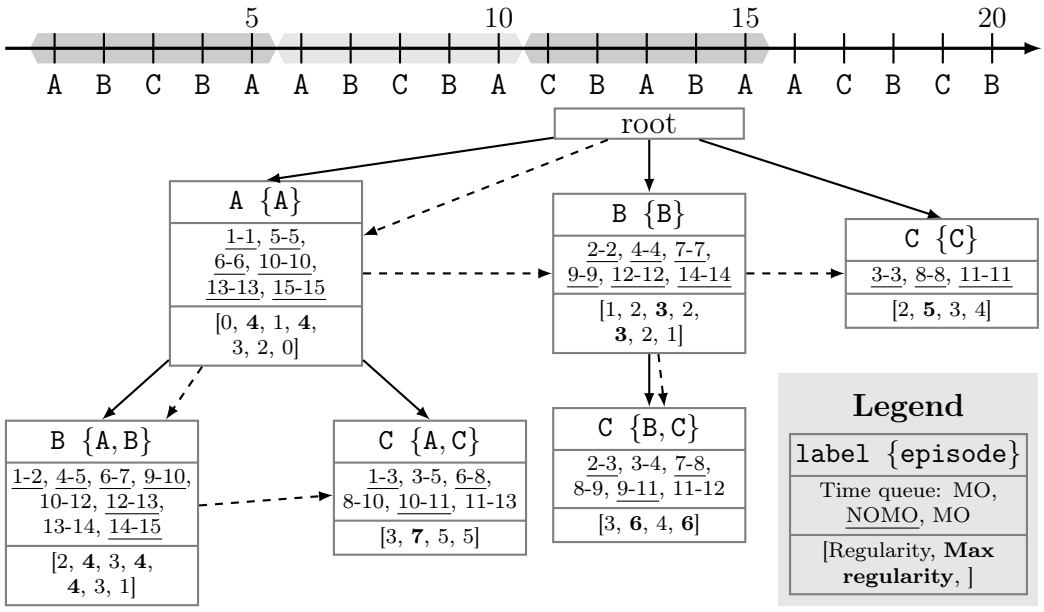



Figure 5.3 – Initial tree construction when the first $m = 3$ batches are read

episodes in the tree. It is computed based on $k$ and on the size of the event label alphabet $\xi$. For example, if the events take values among 5 labels and the value of $k$ is 25, then $D_{k-tree} = 3$: the 5 episodes of length 1, the 10 episodes of length 2, and 10 of the 30 episodes of length 3. In example 5.1, $D_{k-tree} = 2$.

The episode construction step considers pairs of sibling episodes in the $k$-tree (starting from depth 1 to $D_{k-tree} - 1$). For each pair of sibling episodes $E_X$ and $E_Y$, a new entry for $E_Z = E_X \cup E_Y$ is created and linked to its parent $E_X$ (if $E_X$ and $E_Y$ are located at depth $d$, then $E_Z$ is located at depth $d + 1$). Based on the occurrence times of $E_X$ and $E_Y$, the time queue of $E_X \cup E_Y$ is computed and added to the corresponding node in the $k$-tree. The corresponding regularity is computed and compared to the worst regularity in the top-$k$ list. If need be, the top-$k$ list is updated (lines 8–14): $E_Z$ is inserted and the $k^{\text{th}}$ entry (which has just become the $k + 1$ most regular episode) is removed.

There is no guarantee that all the top-$k$ regular episodes are shorter than $D_{k-tree}$. A process similar to the construction of the episodes of length up to $D_{k-tree}$ is thus used to build these longer episodes: the pairs of top-$k$ episodes located at depth $D_{k-tree}$ and higher are combined and their union is investigated. If the new episode belongs to the top-$k$ list, a node is created

in the tree and the top-$k$ list is updated. This new episode is then candidate for further extension with other episodes of the same length (lines 15–24).

The initial tree construction process is illustrated on figure 5.3: the nodes, time queues and regularities of the episodes of length 1 are computed based on the contents of the first 3 batches. The respective regularities of $\{\mathtt{A}\}$, $\{\mathtt{B}\}$, and $\{\mathtt{C}\}$ are 4, 3, 5. Siblings $\{\mathtt{A}\}$ and $\{\mathtt{B}\}$ are then combined to form $\{\mathtt{A},\mathtt{B}\}$ (regularity: 4), $\{\mathtt{A}\}$ and $\{\mathtt{C}\}$ form $\{\mathtt{A},\mathtt{C}\}$ (regularity: 7), $\{\mathtt{B}\}$ and $\{\mathtt{C}\}$ form $\{\mathtt{B},\mathtt{C}\}$ (regularity: 6). The $k = 4$ most regular episodes are thus $\{\mathtt{B}\}$, $\{\mathtt{A}\}$, $\{\mathtt{A},\mathtt{B}\}$ and $\{\mathtt{C}\}$. The constuction of the third level of the $k$-tree is not necessary: there is only one top-$k$ episode of length 2, and can therefore not be merged with another top-$k$ episode of length 2 to form a top-$k$ episode of length 3.

### 5.4.3 Mining new incoming batches

At the end of the first mining step, TKRES has built the top-$k$ list and the $k$-tree, which contains all entries for episodes in depth 1 to $D_{k-tree}$, and the entries of top-$k$ regular episodes at depth higher than $D_{k-tree}$. However, when a new batch of sensor data arrives, the contents of the top-$k$ list might not be up to date anymore. Algorithm 9 details the steps for the maintenance of the data structures.

When a batch $B_{i+m}$ arrives, then the batch $B_i$ becomes outdated, and the information it contains is removed: the top-$k$ list is emptied, the deeper levels of the $k$-tree are removed (these nodes are not maintained, but they will be built again if necessary), and the outdated entries in the time queues of the nodes located at depth lower than $D_{k-tree}$ are revoved (lines 1–3).

After the removal of the outdated information, the update follows a structure similar to this of the initial tree construction. The episodes of length one are first considered, and the new occurrences are registered (lines 4–8). The tree is then updated: the minimal occurrences present in the new batch of data are added to the corresponding nodes (lines 9–15). If necessary, new nodes are created for the new episodes. The construction of the episodes longer than $D_{k-tree}$ occurs only if necessary, i.e. if there are top-$k$ episodes of length $D_{k-tree}$. This construction proceeds exactly like the construction of the longer episodes in the initial mining (lines 16–25).

The update of the $k$-tree with the $4^{\text{th}}$ batch of example 5.1 is illustrated on

**Algorithm 9** TKRES–mining a new incoming batch of sensor data

**Input:** $k$, $B_{i+m}$: the new batch, $k$-tree built when $B_{i+m-1}$ arrived
**Output:** new top-$k$ list, updated $k$-tree

Empty the top-$k$ list
Remove all the nodes at depth higher than $D_{k-tree}$
**for all** node **do** Remove the occurrence times occurring during $B_i$
**for all** event $(e_j, t_j)$ in the new batch $B_{i+m}$ **do**
    **if** the root has no child for $e_j$ **then** Create the node
    Add $\{e_j\}$'s new occurrence $t_j$
Recompute the regularity for the episodes at depth 1
Collect the $k$ labels with the lowest regularity into the sorted top-$k$ list
**for** depth $d = 1$ to $D_{k-tree} - 1$ **do**
    **for all** node $E_X$ at depth $d$ **do**
        **for all** $E_Y$ siblings of $E_X$ **do**
            Merge the time queues of $E_X$ and $E_Y$ (only the entries occurring in $B_{i+m}$)
            **if** $E_X$ has to child for $E_Z$ **then**
                Create the node
            **if** regularity $r^{E_Z} < r^k$ **then** Update the top-$k$ list with $E_Z$
depth $d = D_{k-tree}$
**while** new episodes of length $d$ get in the top-$k$ list **do**
    **for all** node $E_X$ at depth $d$ such that $E_X$ is in the top-$k$ list **do**
        **for all** $E_Y$ siblings of $E_X$ such that $E_Y$ is in the top-$k$ list **do**
            Merge the time queues of $E_X$ and $E_Y$ to form $E_Z$
            Compute the regularity of $E_Z$
            **if** $r^{E_Z} < r^k$ **then**
                Update the top-$k$ list with $E_Z$
                Create the node for $E_Z$ (child of $E_X$)
    $d = d + 1$

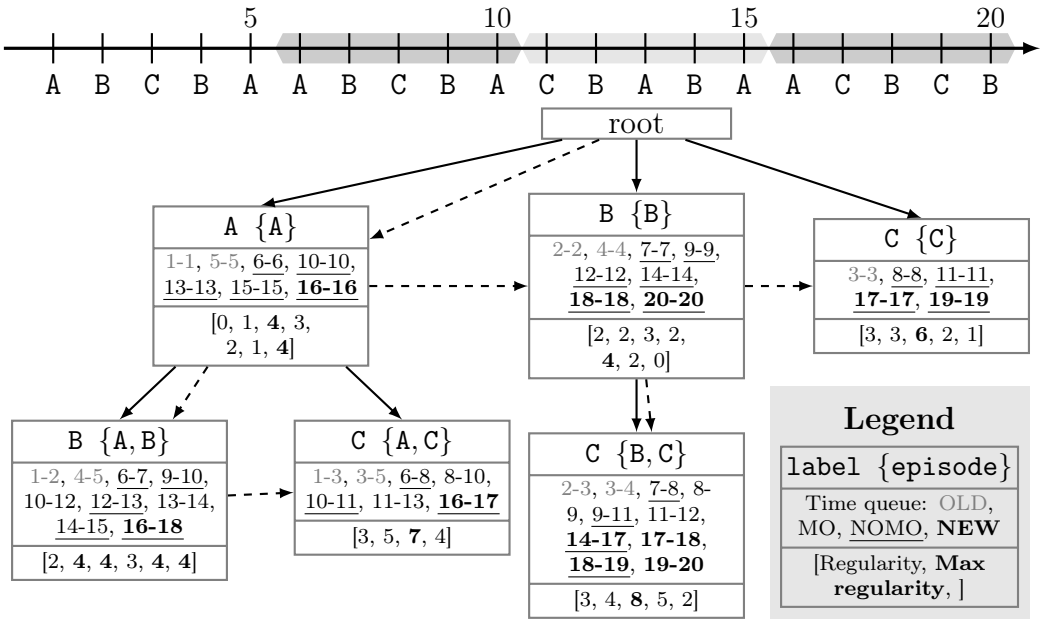



Figure 5.4 – $k$-tree update when a new batch of data arrives

figure 5.4. After the removal of outdated information and the update of the $k$-tree, the four most regular episodes are {A}, {B}, {C} and {A,B}. Here again, the construction of the third level of the $k$-tree is not necessary.

## 5.5 Experimental study on home activity monitoring datasets

### 5.5.1 Performance assessment

In this section, we investigate the output of TKRES on three real-life datasets, the Aruba (#17), Cairo (#14) and Twor (#7) datasets[1] from the CASAS project (Cook and Schmitter-Edgecombe, 2009). Each dataset is composed of two sub-datasets: the dataset of the raw sensor readings, and the corresponding activity annotations. For the three datasets, each sub-dataset is considered separately. They are referred to as the "activity" and "sensor" datasets. All three houses are equipped with motion detection sensors and contact switches on doors. The Aruba and Cairo datasets also have

[1] http://ailab.wsu.edu/casas/datasets/, last consulted on Oct. 23rd, 2015

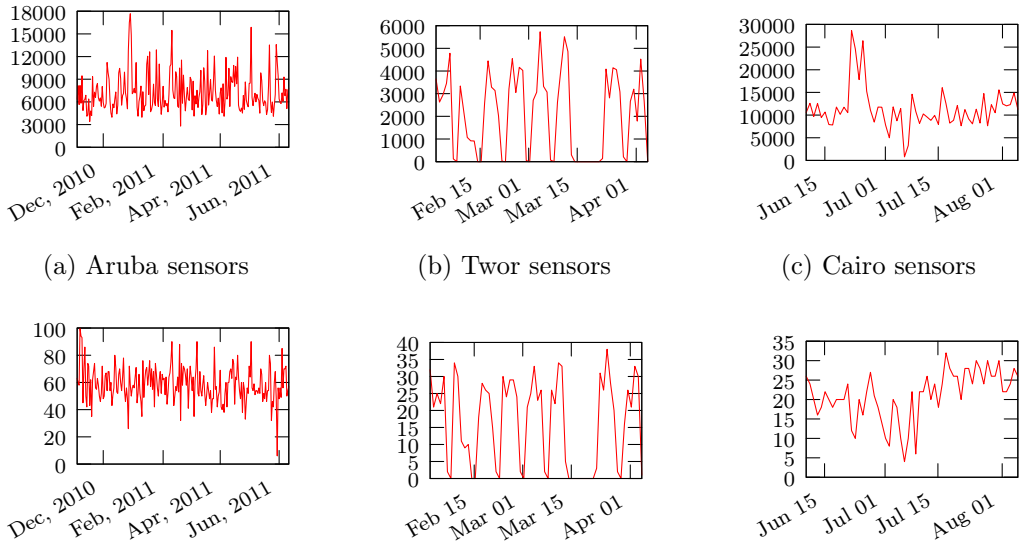
Table 5.2 – Characteristics of the datasets

| | **Aruba** | **Twor** | **Cairo** |
|---|---|---|---|
| **Start** | 2010-11-04 | 2009-02-02 | 2009-06-10 |
| **End** | 2011-06-11 | 2009-04-04 | 2009-08-05 |
| **Duration** | 7 months | 2 months | 2 months |
| # **habitants** | 1 | 2 | 2+pet |
| # **sensor events** | 1 602 986 | 130 347 | 647 487 |
| # **sensor labels** | 72 | 129 | 54 |
| # **activity events** | 12 954 | 1 004 | 1 200 |
| # **activity labels** | 22 | 32 | 26 |

temperature sensors, and the Twor dataset records water usage. The numerical sensor reading (from the temperature and water sensors) were removed: event label recognition is based on string matching, which is not suitable for numerical values. The remaining sensors generate a binary information: the motion sensors are either ON or OFF, and the doors are either OPENed or CLOSEd. Table 5.2 summarizes the characteristics of the datasets. The event distribution in the datasets are different. Figure 5.5 shows the number of events in each batch of data (length of each batch: one day). If the event distribution is fairly homogeneous in the Aruba and Cairo datasets, the Twor dataset contains information for only six days of each week, and there is a period of 9 days without data around March 15$^{\text{th}}$.

In order to assess the performance and scalability of TKRES, we ran experiments with different values for the three key parameters: the size of output $k$, the length of the window $m$, and the maximal occurrence duration $T_{ep}$. When assessing the performance of one parameter, the other two parameters are fixed. Table 5.3 summarizes the experimental setup. For example for the evaluation of $k$, the mining of each dataset was repeated with $k$ ranging from 10 to 300, with $m = 30$ and $T_{ep} = 30$ minutes.

Figures 5.6–5.11 show the average regularity, support, length and stability of the top-$k$ episodes in each experiment and for each dataset. The regularity is counted in hours, the support as the number of non-overlapping minimal occurrences, and the length is the number of labels composing the episode. The stability of the top-$k$ lists, defined for each episode as the percentage of updates where it was part of the top-$k$ list is also plotted. More detailed results, including also the minimal and maximal values for the regularity, support,




(a) Aruba sensors (b) Twor sensors (c) Cairo sensors

(d) Aruba activities (e) Twor activities (f) Cairo activities

Figure 5.5 – Number of events per batch (batch duration = 1 day) for each of the six datasets

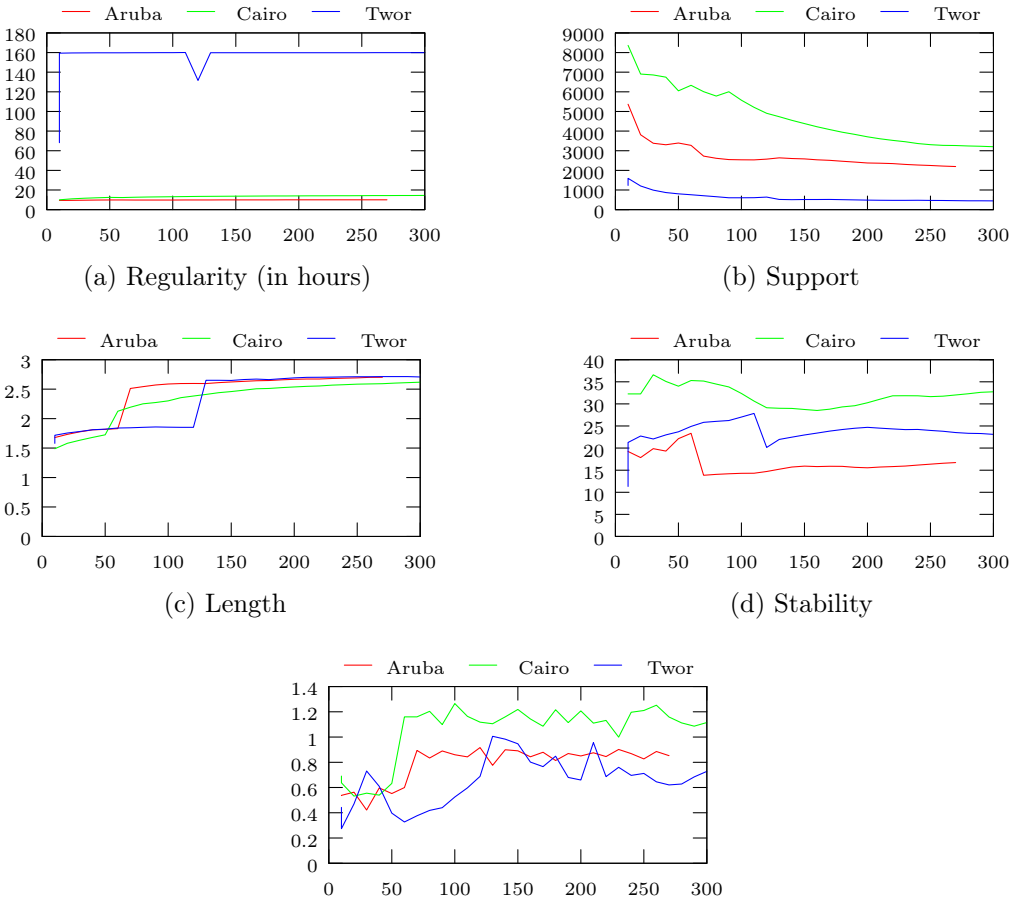
Table 5.3 – Experimental setup for parameter influence assessment

| **Parameter** | **Influence of $k$** | **Influence of $m$** | **Influence of $T_{ep}$** |
|---|---|---|---|
| $k$ | 10 to 300 | 20 | 20 |
| $m$ | 30 | 1 to 60 | 30 |
| $T_{ep}$ | 30 minutes | 30 minutes | 10 minutes to 24 hours |
| **Results** | figures 5.6 and 5.7 | figures 5.8 and 5.9 | figures 5.10 and 5.11 |

length and stability of the top-$k$ episodes are available in appendix D.

Here are some remarks regarding the results:

- Higher values of $k$ tend to result in a deeper $k$-tree (greater episode length, see figures 5.6c and 5.7c), where the episodes are on average less frequent (figures 5.6b and 5.7b).
- The number of batches $m$ in the window mostly influences the regularity (figures 5.8a and 5.9a) of the top-$k$ episodes, which tends to be higher and thus less good (reminder: a higher regularity value means that the episodes occur less regularly). The regularity is indeed computed over




(a) Regularity (in hours)

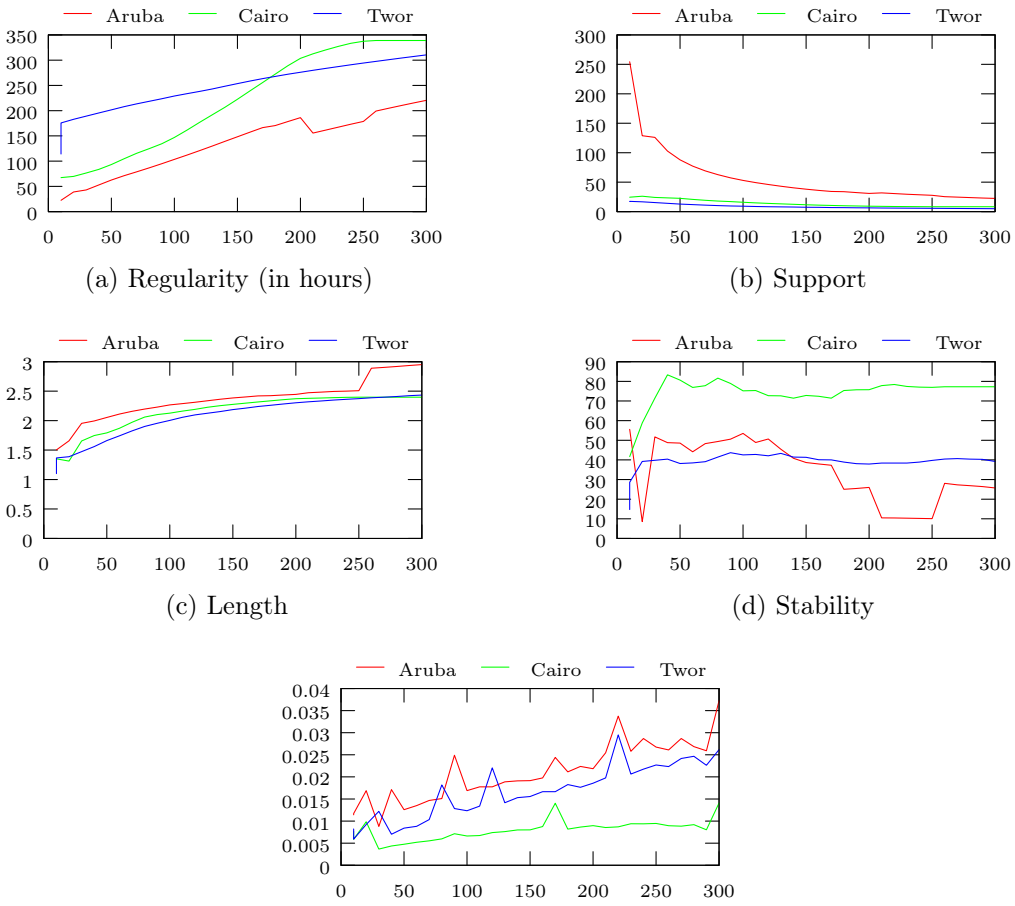
(b) Support

(c) Length

(d) Stability

(e) Average update time (in seconds)

Figure 5.6 – Influence of $k$ and the average regularity, support, length and stability of the top-$k$ episodes in the sensor datasets




(a) Regularity (in hours)

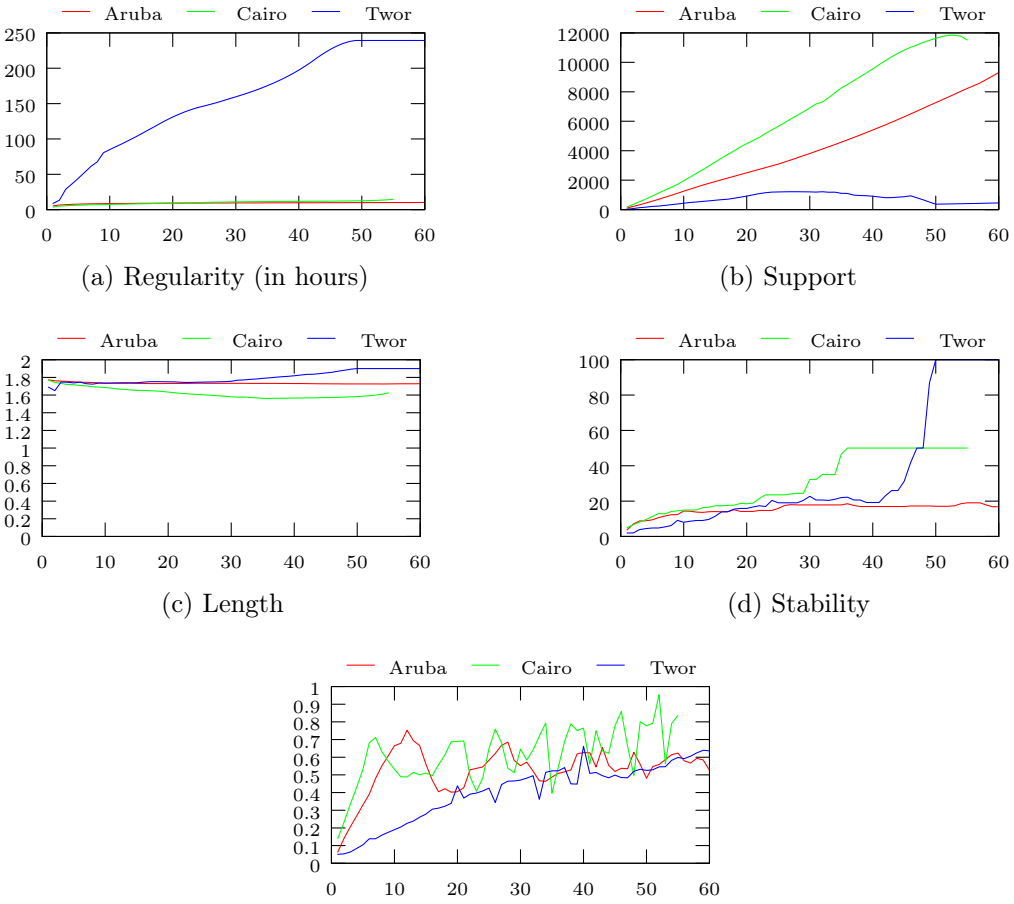
(b) Support

(c) Length

(d) Stability

(e) Average update time (in seconds)

Figure 5.7 – Influence of $k$ on the average regularity, support, length and stability of the top-$k$ episodes in the activity datasets




(a) Regularity (in hours)

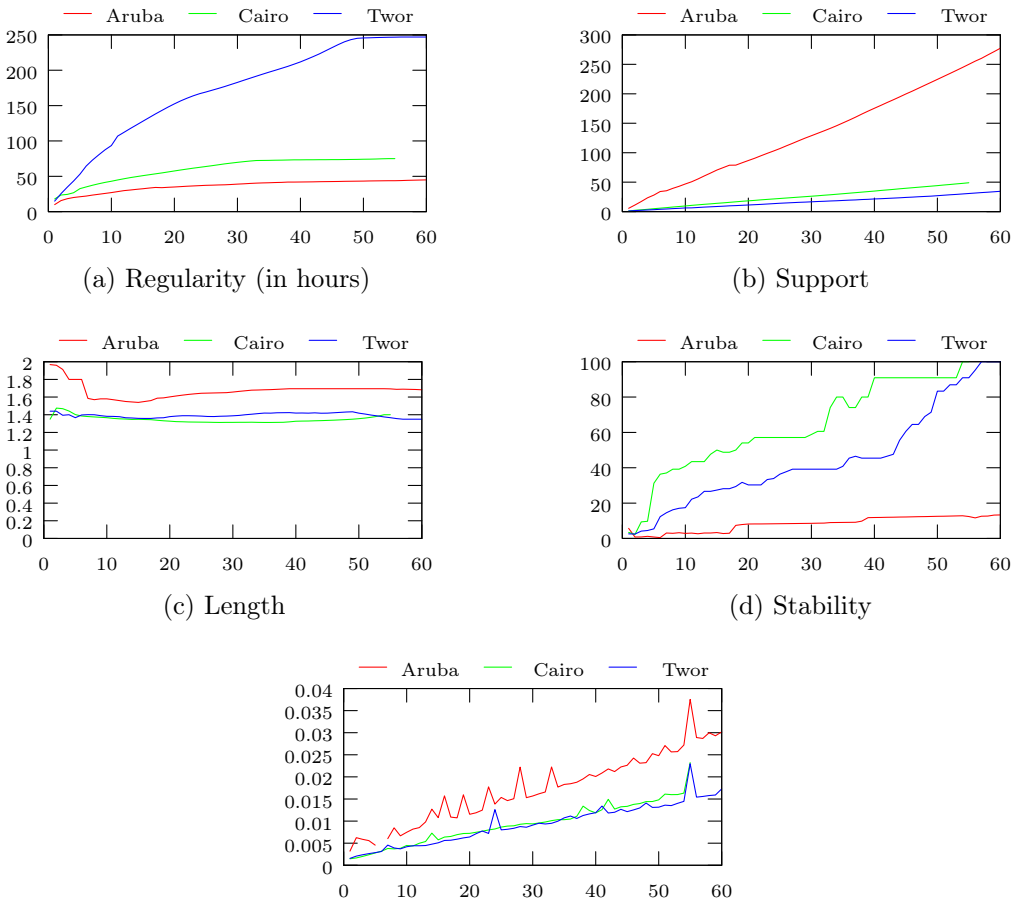
(b) Support

(c) Length

(d) Stability

(e) Average update time (in seconds)

Figure 5.8 – Influence of $m$ and the average regularity, support, length and stability of the top-$k$ episodes in the sensor datasets




(a) Regularity (in hours)

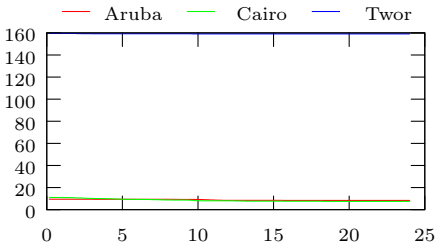
(b) Support

(c) Length

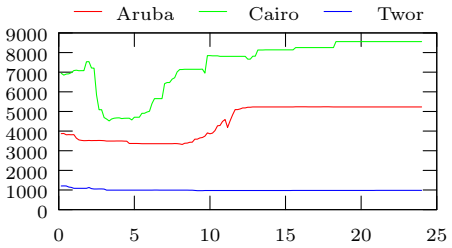
(d) Stability

(e) Average update time (in seconds)

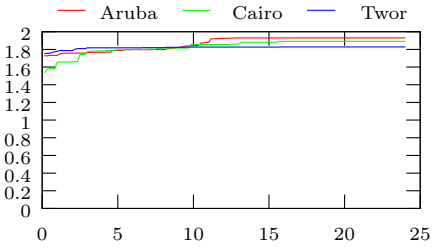
Figure 5.9 – Influence of $m$ on the average regularity, support, length and stability of the top-$k$ episodes in the activity datasets



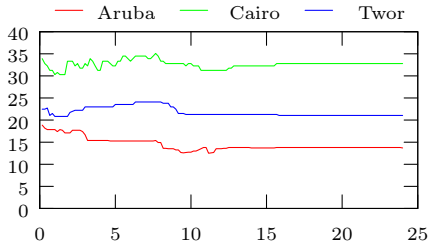

(a) Regularity (in hours)



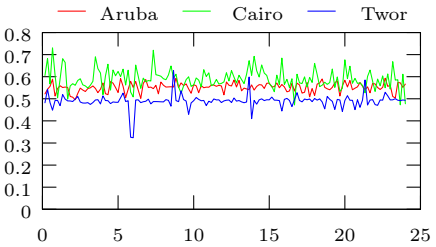

(b) Support




(c) Length

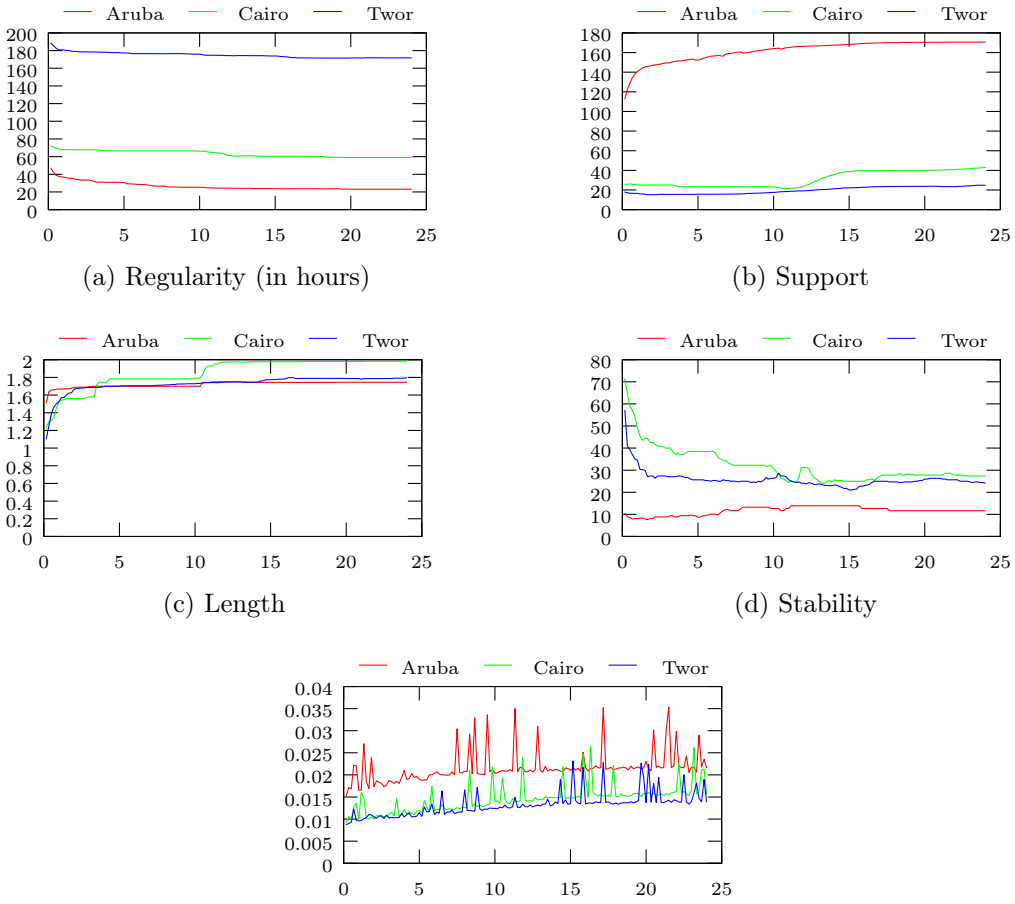



(d) Stability




(e) Average update time (in seconds)

Figure 5.10 – Influence of $T_{ep}$ (in hours) and the average regularity, support, length and stability of the top-$k$ episodes in the sensor datasets




(a) Regularity (in hours)




(b) Support




(c) Length




(d) Stability




(e) Average update time (in seconds)

Figure 5.11 – Influence of $T_{ep}$ (in hours) on the average regularity, support, length and stability of the top-$k$ episodes in the activity datasets

longer periods of time, and the chance to have longer intervals between the consecutive occurrences are higher.

- The support of the episodes also increases with $m$ (figures 5.8b and 5.9b), but not their frequency (the ratio between their support and the length of the window).

- The strongly increasing stability for the Cairo and Twor dataset when $m$ increases (figures 5.8d and 5.9d) is mostly linked to the very small number of updates performed: e.g., Twor contains 64 days of data. With $m = 60$, the $k$-tree is initialized when the $60^{\text{th}}$ batch of data is read and updated three times, once for each remaining batch. With so few updates moreover so close in time, it is normal to get very similar or even the same top-$k$ episodes and a high stability.

- The depth of the $k$-tree (the maximal length of the top-$k$ episodes) is obviously mostly influenced by the value of $k$ (figures 5.6–5.11c). Indeed, for a greater value of $k$, more episodes are requested, and the deeper levels of the $k$-tree need to be investigated.

- When $T_{ep}$ increases, longer occurrences are taken into account. This allows longer episodes to become more frequent (figures 5.10b and 5.11b). Some of these longer episodes eventually become top-$k$ episodes (figures 5.10c and 5.11c).

- The execution time is small and scalable. Figures 5.6e, 5.7e, 5.8e, 5.9e, 5.10e and 5.11e show the average time required for the update of the $k$-tree and $k$-list when a new batch of data arrives. The time requirements increase linearly with each of the three investigated parameters, the influence of $T_{ep}$ being the least noticeable.

### 5.5.2 Qualitative analysis

Tables 5.4–5.6 detail the top-10 most regular episodes within the latest window ($m = 30$) for the Aruba, Cairo and Twor activity datasets. The discovered episodes mainly concern expected daily habits: sleeping and eating patterns. The results in the Twor dataset need to handled carefully: indeed, there is a gap of almost 10 days in the recordings, which impacts much the regularity.

Table 5.4 – Top-10 regular episodes in the Aruba activity dataset

| **Episode** | **Regularity** | **Support** |
|---|---|---|
| Sleeping end | 1 day, 2:03:23 | 53 |
| Sleeping begin | 1 day, 2:05:51 | 53 |
| Relax end | 1 day, 6:20:26 | 350 |
| Relax begin | 1 day, 7:17:28 | 349 |
| Relax end, Relax begin | 1 day, 8:56:33 | 268 |
| Meal_Preparation end | 1 day, 11:32:12 | 207 |
| Meal_Preparation begin | 1 day, 11:57:49 | 207 |
| Meal_Preparation end, Meal_Preparation begin | 1 day, 12:08:29 | 200 |
| Relax end, Meal_Preparation end | 1 day, 18:10:49 | 101 |
| Relax begin, Sleeping end | 1 day, 22:40:27 | 29 |

Table 5.5 – Top-10 regular episodes in the Cairo activity dataset

| **Episode** | **Regularity** | **Support** |
|---|---|---|
| R1 sleep end | 1 day, 6:36:05 | 29 |
| R1 sleep begin | 1 day, 6:39:19 | 29 |
| R2 sleep end | 1 day, 7:00:25 | 29 |
| R2 sleep begin | 1 day, 7:06:45 | 29 |
| R1 wake end | 1 day, 21:05:31 | 29 |
| R1 wake begin | 1 day, 21:18:00 | 29 |
| R1 wake end, R1 wake begin | 1 day, 21:18:00 | 29 |
| R2 sleep end, R2 sleep begin | 1 day, 23:32:37 | 28 |
| Dinner end | 2 days, 0:09:53 | 22 |
| Dinner begin | 2 days, 0:19:30 | 22 |

Table 5.6 – Top-10 regular episodes in the Twor activity dataset

| **Episode** | **Regularity** | **Support** |
|---|---|---|
| R1_work_at_computer begin | 10 days, 1:34:40 | 21 |
| R1_work_at_computer end | 10 days, 1:42:27 | 21 |
| R1_work_at_computer end,<br>R1_work_at_computer begin | 10 days, 2:01:44 | 7 |
| R1_groom begin | 10 days, 2:11:27 | 20 |
| R1_work_at_computer end, R1_groom begin | 10 days, 2:11:27 | 5 |
| R1_groom end | 10 days, 2:13:39 | 20 |
| R1_work_at_computer end, R1_groom end | 10 days, 2:13:39 | 6 |
| R1_groom end, R1_groom begin | 10 days, 2:21:30 | 16 |
| R1_bed_to_toilet end | 10 days, 2:51:22 | 14 |
| R1_bed_to_toilet begin | 10 days, 3:07:05 | 14 |

The most regular episodes in the Aruba and Cairo sensor datasets are drawn on the maps of the smart homes figure 5.13 (the plans are provided with the data on the CASAS webpage). For better clarity, only the involved sensors and combinations of sensors involved in the top-50 episodes are displayed (the value ON or OFF of the sensor is not). For both datasets, an area of interest stands out (the bedroom for the Aruba datasets, an unlabeled room for the Cairo dataset). This area of interest is very interesting for a supervisor: it is very regularly used, and could thus be used for communication, or as a check-up point.

## 5.6 Conclusion

TKRES addresses the problem of mining top-$k$ regular episodes from a sensor stream. It extends the traditional regularity measure to the context of episode mining and designs a method for the discovery and update of such episodes. By controlling the number results in the output, TKRES prevents the supervisor from being overwhelmed with more results than he or she can handle. The top-$k$ approach also take a weight off the supervisor, who does not need to set a support or regularity threshold, which is often a difficult decision to make.

In order to discover the top-$k$ regular episodes, we introduce the $k$-tree data structure, which maintains the occurrence information of the episodes. We propose to make a trade-off between the memory usage and the computational time to update episode knowledge when data changes, thanks to the setting of a depth boundary $D_{k-tree}$ on the $k$-tree: the episodes shorter than $D_{k-tree}$ are fully investigated, stored and maintained, when the longer episodes are built only when they belong to the top-$k$ list. Experimental results on three smart home datasets show the efficiency of TKRES and its ability to detect patterns relevant to the human activity monitoring community.

This work could be further extended. In particular, using regularity only favors the short episodes. However, regularity is not the only interesting measure. The frequency, the length or the duration of the episodes could also help better target the interesting patterns, via user-defined combinations of these measures and regularity. It would also be interesting to investigate closed regular episodes.

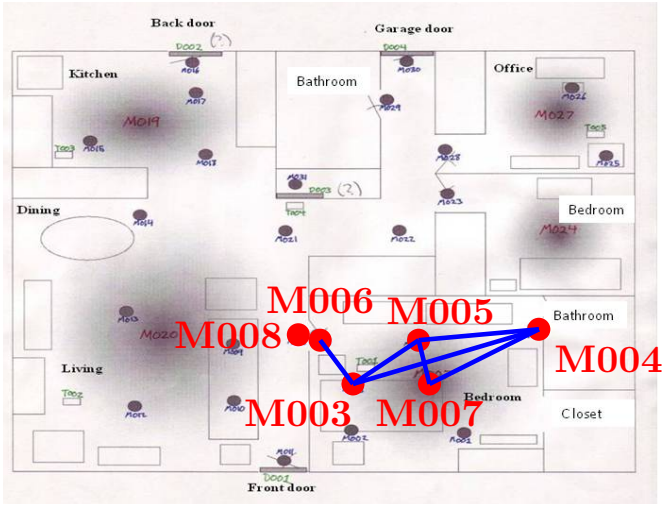



Figure 5.12 – Combinations of sensors involved in the top-50 regular episodes in the last window for the Aruba sensor datasets ($T_{ep} = 30$ minutes, $m = 30$)

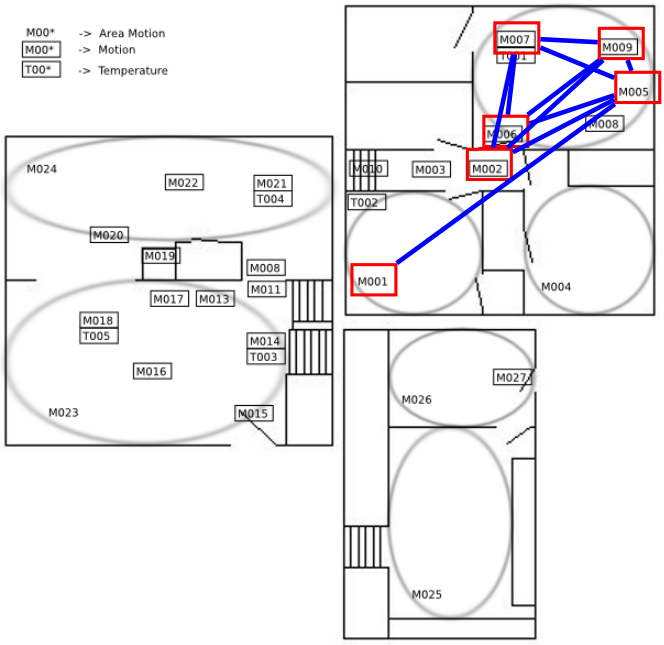



Figure 5.13 – Combinations of sensors involved in the top-50 regular episodes in the last window for the Cairo sensor datasets ($T_{ep} = 30$ minutes, $m = 30$)

# Chapter 6

# Conclusion

## 6.1 Summary and contributions

The work that is presented in this PhD thesis contributes to the activity monitoring community. In particular, it focuses on the discovery and description of the habits of an habitant in a smart home. The main points that have been covered by this work are:

- We have proposed a new periodicity description for reoccurring patterns, as well as an accuracy measure to evaluate the quality of the periodicity (Soulas et al., 2013). It is based on the description of its occurrence time distribution within a period of interest (e.g. one day or one week), via a mixture of Gaussian components. Each component, described with its average occurrence time and a standard deviation, means that an episode occurrence is expected to occur in the vicinity of its average occurrence time (more or less its standard deviation). Moreover, we have proposed a measure to evaluate the conformance between the mixture model and the observed occurrences of the episode: the accuracy. It measures the proportion of the expected occurrences that we indeed observed. This new formalism lead to the definition of a new data mining problem: the discovery of such patterns.
- We have proposed two algorithms for the discovery of such periodic patterns: extended Episode Discovery xED (Soulas et al., 2013, 2015) for static databases, and extended Episode Discovery on streams sxED (Soulas and Lenca, 2015) for evolving data.

- We have extended the definition of regularity to event sequence, and proposed an algorithm for the discovery of the top-k regular episodes in event streams TKRES (Amphawan et al., 2015).

- We moreover proposed in (Simonin et al., 2015) a model driven engineering approach to involve the supervisor in the activity monitoring. This contribution is detailed in appendix A.

All the algorithms are evaluated both qualitatively and quantitatively on real life data, coming from databases that are widely used in the activity monitoring community. xED and the model-driven engineering approach are designed for the discovery of habits on static data. sxED and TKRES can also handle non-stationary data: the discovered knowledge is updated with the arrival of new sensor readings, allowing us to maintain up-to-date knowledge on the current behavior of the user.

## 6.2 Future work

### 6.2.1 Periodicity characterization and evaluation

The definition used to assess the periodicity of the episodes allows a natural description of the habits, since it translates sentences like *"the user wakes up around 7 during week days"*. It also allows a characterization of the usual time variations (the standard deviations of the components, which is activity-dependent), and a quantitative assessment of how true the periodicity description is (*"in 80% of the week days, the user wakes up around 7"*). This definition, coupled with xED and sxED allows the discovery of relevant episodes, and provides the supervisor with an explanation on how the pattern occurs. This description also allows the discovery of out-of-the ordinary behaviors, like an occurrence of the pattern happening outside of an identified component, or an expected but missing occurrence.

However using accuracy as the sole measure for periodicity interest ranking puts the periodic components with a low standard deviations on an equal footing with components having a high standard deviation. A description with an arbitrary high standard deviation would thus perform better than a more targeted but less accurate description: accuracy can be "deceived"

into selecting the description with the high variability as the most suitable periodicity.

This is not an actual problem for xED and sxED. Indeed, the methods used for the tuning of components (based on Expectation-Maximization) do not generate periodicities with a very high standard deviation. A threshold on the maximal standard deviation can moreover further constrain the allowed variability for periodic patterns.

An interesting follow-up study would be to develop and analyze alternative quality measures. The *compression power*, used in xED, allows to favor long and highly accurate episodes. It could also favor frequent episodes if a factorization on frequent (yet not periodic) episodes is added as well, and periodicities with low variability if the gap between the expected and observed occurrence times are encoded too. Other interesting measures could include the coverage of the periodicity (the percentage of the occurrences that are explained by the periodicity description), and other compromises between accuracy, variability, length and frequency.

### 6.2.2 Handling uncertainty

Sensor data contain noise, linked to the measuring errors, message transmission failures, lack of battery, etc. Moreover, the sensors register activity that may have not been triggered by the monitored person (but by a family member, a caregiver, a pet, etc.). The sources for uncertainty are multiple, but have not been tackled yet. Kim et al. (2015) list the sources for uncertainty in activity recognition systems, and make a first attempt to assess their relative contributions on the activity recognition performance.

The Gaussian Mixture Models and accuracy measure used in xED and sxED tolerate and measure variability. However, they do not explain it, nor assess its meaning on the health or well-being of the elderly people. Additional analysis of the variability, its sources, and its influence on the pattern discovery algorithms would give new insights on the behaviors of the users.

Such analysis would also help refine habits, notably by assessing the statistical relevance of the discovered episodes and their periodic components.

### 6.2.3 Trend analysis and anomaly detection

A straightforward anomaly detection application of periodic and regular episode discovery is the detection of expected but missing occurrences, or the detection of unexpected occurrences. But it would be interesting to investigate more refined anomaly detection strategies.

Habits evolve with time. Some evolutions are perfectly normal, such as those linked with seasonal change. However, a habit change may sometimes also hint at an increasing difficulty to carry out daily activities, or at symptoms of a degenerative disease: growing lack of interest, increasing apathy, disorientation, etc. There are thus interesting follow up studies to be carried out in trend analysis and anomaly detection.

### 6.2.4 Episode and activity recognition

In xED, sxED and TKRES, episodes are defined as sets of event labels, and the recognition of an episode is based on string matching on the labels composing the episode. This approach restrains the algorithms to th use of simple event labels, where string matching on the labels is relevant. This does not adapt well to numerical data, or to more complex data types. This also reduces the class of sensors that can be used for such analysis.

Moreover, a given activity may generate different sensor footages. For example, depending on the type of food that is being cooked, the"prepare a meal" activity does not involve the same sensors. It would thus be interesting to investigate strategies for the clustering of the episodes into activities, as well as the potential advantages of a periodicity analysis on episode clusters compared to a periodicity analysis on raw episodes.

### 6.2.5 Feedback towards the users

In most current activity monitoring approaches, including the one developed in this thesis, sensors are installed in the home of an elderly person, and the generated data is analyzed, either on or off-line. The monitoring process is transparent to the user, who carries out his or her daily life as if the sensors were not active. Moreover the supervisors (physicians, caregivers, family members, or even the monitored person) is usually not an algorithm expert.

They need help parameterizing the algorithms and analyzing the results they produce.

In order to complete the monitoring process and help the end-users take the most of the opportunities offered by activity monitoring systems, it is necessary to develop an integrated framework. This framework would allow the tuning of the algorithms to each patient, and provide the supervisor with the results of the analyses as well as data visualization tools.

Initial work towards such a framework has started, resulting in (Simonin et al., 2015). This contribution is also included in appendix A, and details a process mining approach, based on the automatic generation and transformation of models describing the data relevant to the supervisor. In particular, the system hands back control to the supervisor, who, thanks to his or her domain knowledge, can select some patterns considered as critical. A statistical analysis then selects other patterns the supervisor should also deem interesting. Appendix B presents some of the visualizations that have been used and could help the supervisor gain a better understanding of the behaviors of the user. They are mostly standard representations (pie charts, heat maps, etc.), and lack depth: as of now, they are do not carry enough information, are not self-sufficient cannot replace the textual results: new ways to represent and communicate about activities and habits need to be devised.

# Appendix A

# Model driven engineering approach for ADL supervision

## A.1 Introduction

The behavior of the user is monitored by a supervisor, usually a physician, a caregiver, or the user him/herself. This supervisor is not a data mining expert, and could be overwhelmed by the amount of data, or the deployment and tuning of the algorithms.

In order to make event logs more accessible to the supervisor, we suggest a process mining approach (Simonin et al., 2015). Process mining is centered on the processing of event logs (van der Aalst et al., 2004). It is more accessible than a data mining approach for the supervisor, since it is closer to the processes. It helps deal with the lack of structure in the event log. This is why we propose and describe in the coming sections a process mining approach, assisting the supervisor for the monitoring. This approach completes our algorithmic contributions by including the user more actively in the mining process.

The *system* (the smart house) supports one or more *activities* of the user, where the system is specified in the system viewpoint and the activities in the business viewpoint (Zachman, 1987). The *support* relationship established between these viewpoints is considered as an *alignment* between a model of the system, and a model of the activities composing the process (Simonin et al., 2011).

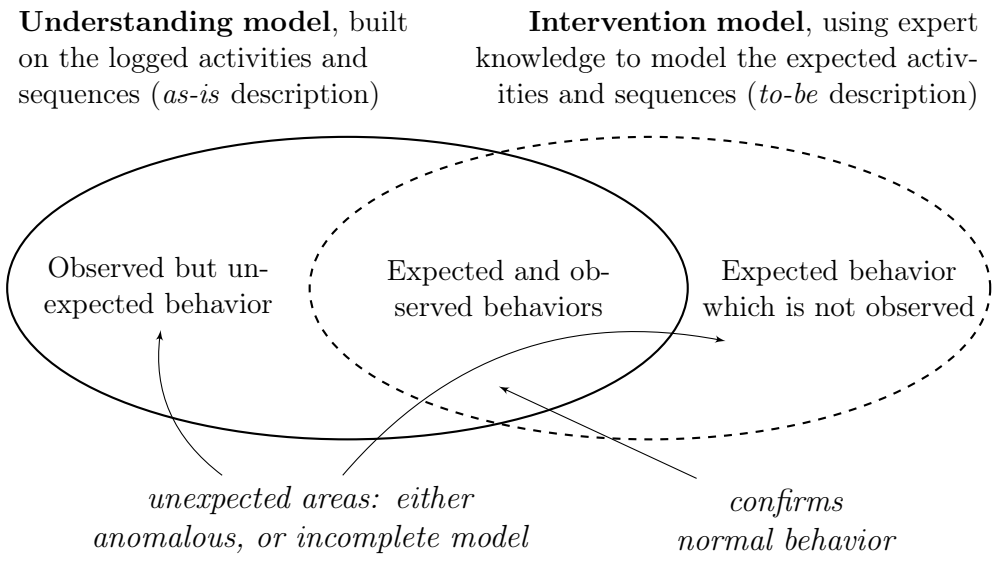



Figure A.1 – Understanding and intervention models

This contribution includes first the design of an understanding model of the activities of a system user (an *as-is* description: what activity is carried out, and when). An intervention model completes the understanding model to assist the supervisor (a *to-be* description). In particular, the intervention model enables the supervisor to act on the critical activities, and to detect anomalies. The models are automatically designed and built with a model driven engineering (MDE) approach.

Figure A.1 presents the understanding and intervention models: They are designed through rules, aiming at the design of relevant models for process mining (van der Aalst, 2013). In particular, these rules help with data understanding: they highlight data dependencies, anomalies, etc. These models contain some instantiations of concepts (the concept of *activity* is for example instantiated by the `go to bed` activity) that are represented in a meta-model, following a MDE approach. The supervising tool is built via the implementation of these rules.

This approach is complementary to classical data mining approach. Data mining produces also some patterns based on event logs. But most of the time, these patterns come only from machine learning algorithms without taking into account the knowledge and goals of the supervisor. We propose here to enhance the role of the system supervisor with the help of MDE: the supervisor can define some rules specifying the expected behaviors of the

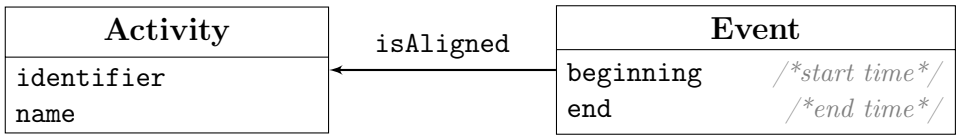



Figure A.2 – The concepts in the understanding model, and their relations

system user.

We present in section A.2 the automatic design of an understanding model for the monitored activities of a system user, based on event logs. The intervention model is presented in section A.3 and allows the action of the supervisor on the critical activities of the system user. Throughout sections A.2 and A.3 the understanding and intervention models are illustrated on a running example: the KA dataset (van Kasteren et al., 2010a) (this dataset is also used in chapters 3 and 4). Section A.4 contains the conclusions and perspectives.

## A.2 Understanding model

The understanding model is the transformation of an event log model into a process model. This contribution allows the supervisor to monitor sequences of activities, both serial (ordered) and parallel.

The event logs do not represent unambiguously the real behaviors of the system user. For example, some activities can be interleaved, which is not easily detected by a human eye among a big set of event log. This entails the transformation of an event logs model, constrained by the user activities, into a process. The transformation results in a process model, highlighting the activity sequences composing it.

### A.2.1 Event Logs and Activities

The model in input of a transformation model must conform to a meta-model representing the requirements for the model. The concepts used to design this understanding model, as well as their relationships are represented in figure A.2.

### A.2.2 Automated design of the understanding model

The understanding model results from a model transformation composed of rules, applied on the set of event logs. These rules allow the automated design of the understanding model activities, and the sequences between these activities. The terms of *mined activity* and *mined sequence* are used below to denote respectively an activity and a sequence in the understanding model.

#### A.2.2.1 Design of mined activities

**Alignment rule:** A *mined activity* is an activity that is aligned with at least one event log.

The basic alignment rule ensures that an activity is taken into account in the understanding model if at least one event entry logs it. This rule allows the supervisor to restrict the activities of the system user to those that are actually performed. The activities that are not transformed into mined activities can be linked to a technical problem affecting the event log production. They can also result from the user, who does not respect the procedure.

#### A.2.2.2 Mined sequence design

A mined sequence represents a temporal sequence from a mined activity (called *source*) to a mined activity (another one or itself, called *target*). For example, in the temporal sequence from the `prepare breakfast` activity to the `take shower` activity, the source activity is `prepare breakfast` and the target activity is `take shower`. The presence of a sequence suggests a relationship between two logged activities, thus helping the supervisor understand the recorded data. The meta-model of the understanding model defines a mined process, based on the concepts of mined activity and mined sequence, conforming to the activity and sequence design rules, as described in figure A.3. The `MinedActivity` and `MinedSequence` classes contain attributes for the assessment of their relevance (`input/outputWeight`, `timeSigma/Square/`, etc.).

The mined sequences are discovered from the event log set through four Mined Sequence Design Rules. The rules are also represented thanks to

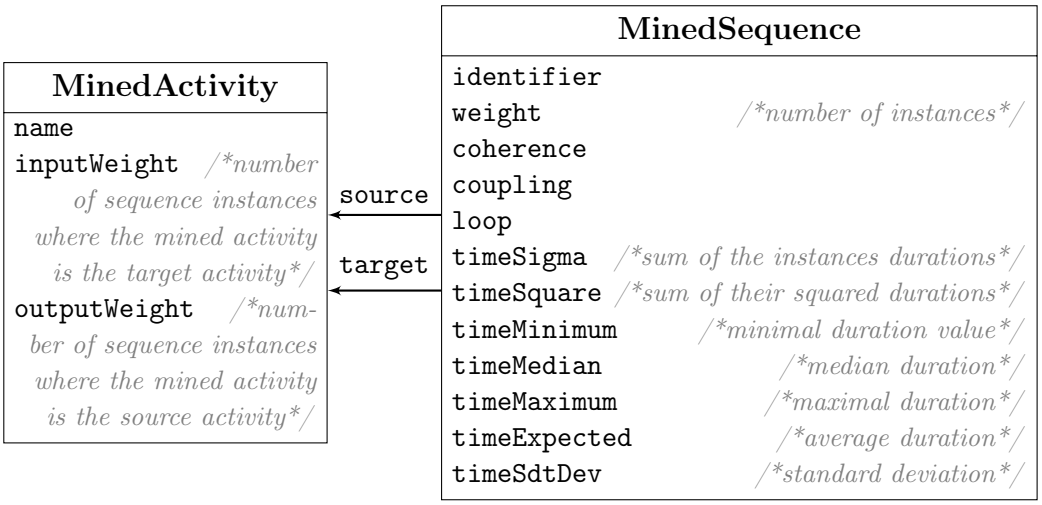



Figure A.3 – Definition of the `MinedActivity` and `MinedSequence` classes

graphs in figure A.4. In the following description of the mined sequence design rules, let $A_1$ and $A_2$ be two mined activities, respectively aligned with event logs $E_1$ and $E_2$. Let $S$ be the mined sequence from $A_1$ to $A_2$:

**Coherence** (figure A.4a)**:** $S$ verifies the coherence property if $A_2$ is coherent with $A_1$: ($E_1$.beginning $<$ $E_2$.beginning) AND ($E_2$.end $<$ $E_1$.end)

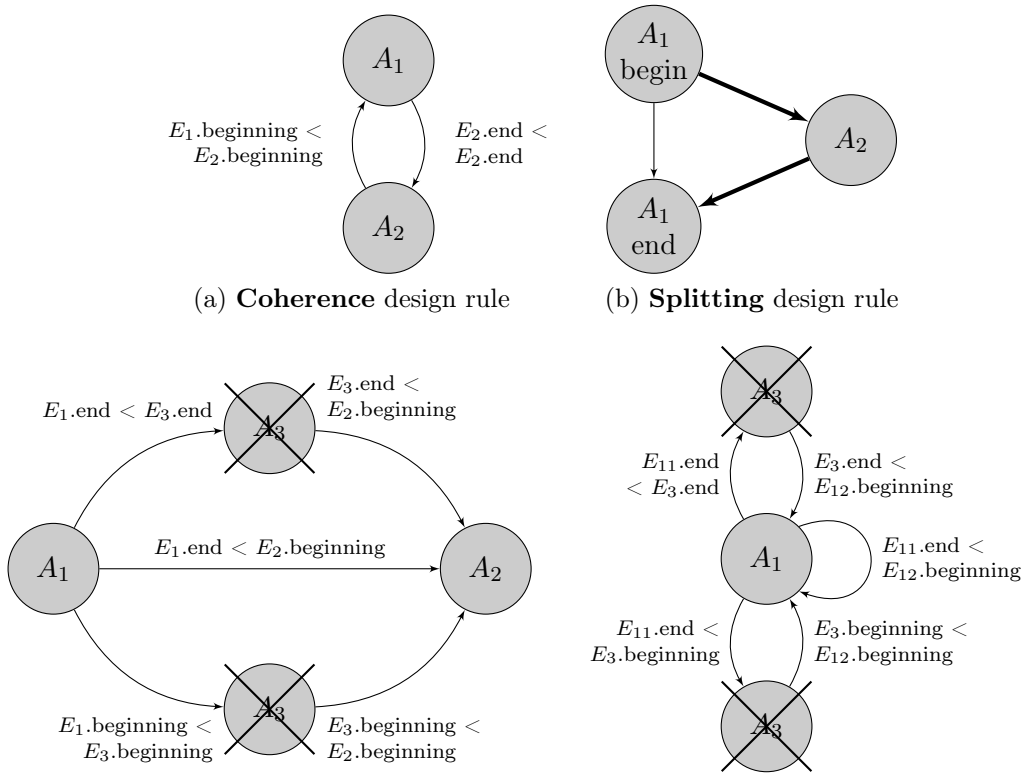
Coherence highlights a scenario where two activities are interleaved (parallel).

**Splitting** (figure A.4b)**:** If $S$ has a coherence property, then the $A_1$ activity is split into two activities $A_1\_begin$ and $A_1\_end$, respectively aligned with $E_1$.beginning and $E_1$.end.

When a mined sequence has a coherence property, the splitting rule proposes an appropriate sequence design: it creates three sequences:

- a sequence from the $A_1\_begin$ activity to the $A_1\_end$ activity,
- a sequence from $A_1\_begin$ to $A_2$,
- a sequence from $A_2$ to $A_1\_end$.

**Coupling** (figure A.4c)**:** $S$ has a coupling property if:
($E_1$.end $<$ $E_2$.beginning) AND there exists no event log $E_3$ such that: ($E_1$.end $<$ $E_3$.end $<$ $E_2$.beginning) OR ($E_1$.end $<$ $E_3$.beginning $<$ $E_2$.beginning)




(a) **Coherence** design rule (b) **Splitting** design rule

(c) **Coupling** design rule (d) **Looping** design rule

Figure A.4 – Representation of the design rules by a graph

This property specifies that there is a coupling between two activities of a system user when no activity beginning or end is logged between the end of the first activity and the beginning of the second one. The monitoring of the consecutive activities of the user is based on coupling.

**Looping:** Let $A_1$ be a mined sequence ,aligned with two event logs $E_{11}$ and $E_{12}$. The mined sequence $S$, designed from $A_1$ to $A_1$ has a loop property if:
($E_{11}$.end < $E_{12}$.beginning) AND there exists no event log $E_3$ such that: ($E_{11}$.end < $E_3$.end < $E_{12}$.beginning) OR ($E_{11}$.end < $E_3$.beginning < $E_{12}$.beginning)

A loop is a mined sequence from an activity to itself, satisfying the coupling property. A loop allows the supervisor to group the instances of one activity (see figure A.4d).

The understanding model can be represented by a graph where every vertex

is a mined activity. Every directed edge is a mined sequence, automatically generated by the model transformation implementing the previous four mined sequence design rules. An oriented edge from the mined activity $A_1$ towards the mined activity $A_2$ means that $A_1$ comes before $A_2$. An edge represents a mined sequence having either a coherence property, a coupling property or a loop property. The number of instances of each sequence (the *weight* attribute) is given in the label of each edge.

### A.2.3 Illustration on the KA dataset

A dataset containing 245 event logs of one person living in a smart house (van Kasteren et al., 2010a) is used throughout this contribution for illustration purposes (KA dataset, also used in chapter 3). The system is the house and the user is the person living there. The supervisor can be a doctor or anyone in charge of taking care of the user. The technical view of the system is based on a wireless sensor network. Every node of the network sends an event every time a change is detected by a sensor.

From a supervisor point of view, the size of the event log file is overwhelming, and prevents a good understanding of the system user activities. A first assistance is the representation of the activities and the activity sequences, aligned with the event logs. A sequence is an activity relationship specifying that an activity comes just after another activity.

The understanding model is automatically derived from the application of the mined sequence design rules, implemented by a model transformation. In the dataset (van Kasteren et al., 2010a), the *splitting* rule is applied twice:

- The `prepare Dinner` activity is split because of its coherence with the `use toilet` activity;
- The `go to bed activity` is split because of its coherence with the `use toilet` activity.

Figure A.5 presents the understanding model that is deduced from the 245 activity logs. This model is a graph as defined previously: e.g., the oriented edge from the `prepare breakfast` vertex to its successor vertex `take shower` means that a mined sequence from the `prepare breakfast` mined activity to the `take shower` mined activity is discovered in the set of event

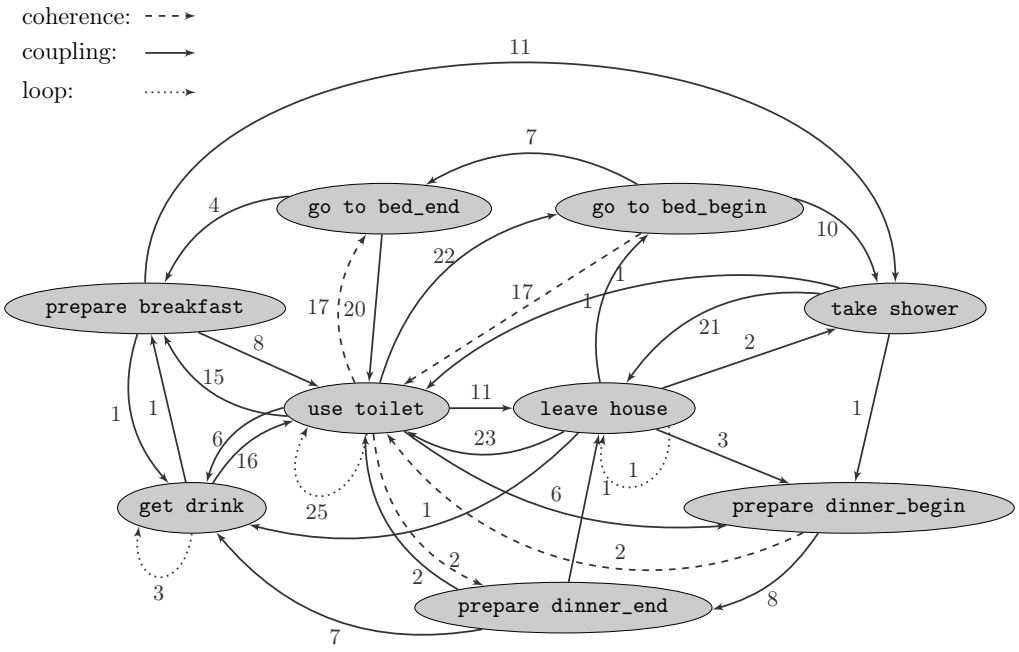



Figure A.5 – Understanding model of the KA dataset, represented by a graph

logs. This mined sequence is supported by 11 instances (edge weight of 11), and verifies the coupling property.

## A.3 Intervention model

The intervention model results from a transformation of the understanding model based on the monitoring needs of the supervisor. The supervisor of the activities of the system user chooses some sequences that he or she considers as critical. Critical activities (resp. sequences) are marked as such based on the supervisor's expert knowledge. The intervention model contains the critical sequences as well as all the mined sequences that have at least the same importance in the understanding model. The importance is defined by various statistical characteristics, detailed in the coming subsections.

### A.3.1 Sequence pruning based on critical sequences

In the intervention meta-model, a `critical` attribute and a `pruned` attribute are added to the understanding meta-model in the `MinedSequence` class (see figure A.3). The supervisor can choose statistical rules to specify useful thresholds for the pruning of irrelevant sequences. When a mined sequence is pruned in the intervention model, the supervisor can also analyse the reasons of this pruning. We select two statistical thresholds each defined by a mined sequence pruning rule. The thresholds are estimated in relation to the critical sequences: a critical sequence cannot be pruned.

**Instance weight:** A mined sequence is maintained in the intervention model if its number of instances (`weight` attribute of the `MinedSequence` class) divided by the number of instances of all the mined sequences having the same property is greater than a $T_{IW}$ threshold.

This rule allows the supervisor to ignore the mined sequences that are not frequent enough.

**Maximal duration:** A mined sequence is maintained in the intervention model if the maximum duration (`timeMax` attribute) is lower than a duration threshold $T_{MD}$, where the duration equals the time interval between the beginning of the target activity and the end of the source activity of the sequence.

This allows the supervisor to prune sequences that take too much time (the dependence between the source and the target might not be relevant). Depending on the context, a minimal duration threshold could be relevant too.

### A.3.2 System using anomaly detection

Anomalies are searched for critical mined sequences of the intervention model, and are conditioned by the understanding model. We suggest defining the detection of an anomaly through mined sequence anomaly rules:

**Source weight:** There is an anomaly for a critical mined sequence $S$ in the intervention model when the instance count of $S$ (`weight` attribute) divided by the sum of the weights of the mined sequences having the same source activity is lower than a threshold $T_{SW}$.

This rule assesses the relative weight of a critical mined sequence, compared with all the mined sequences sharing the same source mined activity. The supervisor can thus obtain information about the relevance of the critical mined sequence when the system user participates in the source activity of this sequence.

**Target weight:** There is an anomaly for a critical mined sequence $S$ in the the intervention model when the instance count of $S$ (`weight` attribute) divided by the sum of the weights of the mined sequences having the same target activity is lower than a threshold $T_{TW}$.

This rule is the counterpart of the previous rule, focusing this time on the target activity.

The next two rues compare the critical sequences to other indirect paths linking the source to the target activities in the critical sequence.

**Time factor:** There is an anomaly for a critical mined sequence $S$ in the intervention model if the average time span of the mined sequence paths (`expectedValue` attribute of the mined sequences used in the paths) from the source to the target of $S$ divided by the average time span of $S$ is greater than a threshold $T_{TF}$.

This rule informs the supervisor of situations when too much time is spent between the source and the target activity of a critical mined sequence, compared to the duration of the indirect paths.

**Weight factor:** There is an anomaly for a critical mined sequence $S$ in the intervention model if the average weight (`weight` attribute) of the indirect paths from the source of $S$ to the target of $S$ divided by the weight of $S$ is greater than a threshold $T_{WF}$.

This rule highlights situations when the critical mined sequence is less used by the user than some indirect paths, and could highlight for example a poor organization of the house, which does not allow a fast traversal.

### A.3.3 Experimental results

For the experiment, two mined sequences were chosen as critical: from `go to bed_end` to `use toilet` and from `prepare breakfast` to `take shower`.

The sequence pruning rules are implemented with minimum thresholds of 5.0 for the time factor and 0.15 for the weight factor, fixed based on the needs of the supervisor. Here are some remarks on the automatic detection of anomalies resulting from the implemented transformation:

1. The $T_{SW}$ ratio (*source weight* rule) shows that the `go to bed_end` to `use toilet` mined sequence is the main sequence having the `go to bed_end` mined activity as source: it represents 20 / (20 + 4) = 83% of the mined sequences starting with `go to bed_end` (the other sequence is from `go to bed_end` to `prepare breakfast` and has a weight of 4);

2. The $T_{SW}$ ratio shows that `prepare breakfast` to `take shower` is the most important mined sequence having the `prepare breakfast` mined activity as source: it represents 55% of the mined sequences starting with `prepare breakfast`;

3. The $T_{TW}$ ratio (*target weight* rule) shows that the `go to bed_end` to `use toilet` mined sequence is not a very significant activity sequence in relation to the eight sequences having `use toilet` mined activity as target: it represents only 18% of them;

4. The $T_{TW}$ ratio shows that the `prepare breakfast` to `take shower` mined sequence is significant among the two sequences having the `take shower` mined activity as target (48%);

5. The average time of the `go to bed_end` → `prepare breakfast` → `take shower` → `leave house` → `use toilet` sequence path has a 8.17 factor (*time factor* rule) in relation to the average time of the `go to bed_end` to `use toilet` critical mined sequence. The weight of the `go to bed_end` → `prepare breakfast` → `take shower` → `leave house` → `use toilet` sequence path has a 0.2 factor (*weight factor* rule) in relation to the average time of the `go to bed_end` to `use toilet` mined sequence. The direct path spans much less time, and is more frequent than the indirect path;

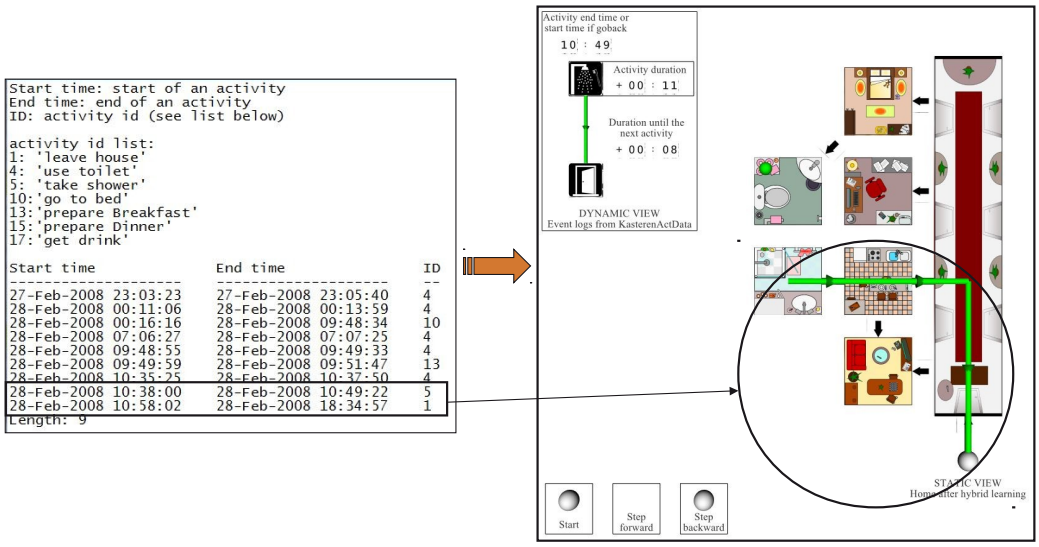
6. The average time of the `go to bed_end` → `prepare breakfast` → `use toilet` sequence path has a 5.37 factor (see the *time factor* rule) in relation to the average time of the `go to bed_end` to `use toilet` mined sequence. The weight of the `go to bed_end` → `prepare breakfast` → `use toilet` sequence path has a 0.2 factor (see the *weight factor* rule) in relation to the average time of the `go to bed_end` to `use toilet` mined sequence.

Figure A.6 represents a styled view of the house: the rooms are represented separately, and are linked to one another when there exists a door between them in the real house. The paths for two mined sequences are represented: the `take shower` to `leave house` sequence (A.6a) and the `prepare breakfast` to `use toilet` sequence (A.6b). The later is considered abnormal: the target weight of the sequence is $8/(8+20+1+23+2+16) = 0.1$, which is lower than the weight factor (0.15): this sequence is significantly less frequent than the others ending with `use toilet`, and is thus reported.

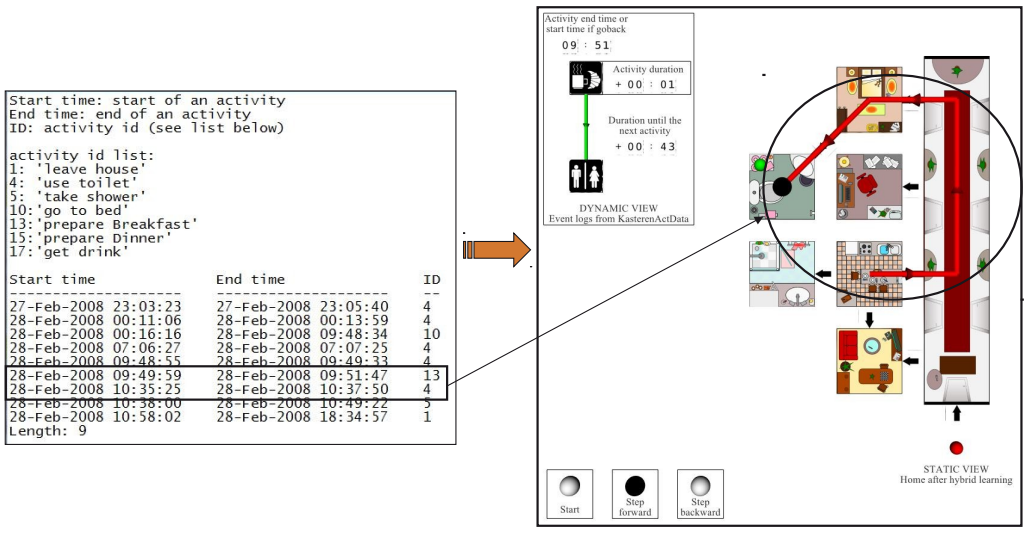
## A.4 Conclusion and Perspectives

In this contribution, we propose some rules to automatically design an understanding and intervention model of the activities of a user. This support tool targets here the sequential and parallel activities of the user. The rules are implemented using a MDE approach. This implementation provides first an understanding model representing the behavior of the system user. When the supervisor chooses critical sequences, the rules for the intervention model allow the pruning of irrelevant mined sequences from the understanding model, thus producing a custom and compact intervention model, adapted to the supervisor's needs and expertise. The anomalies in the critical mined sequences are detected with appropriate statistical measures (the set of measures could also be further enriched).

The perspective is the design of a relevant event log model with a statistical pruning of the dependencies between event logs. This pruning should be consistent with the activity sequence pruning suggested here. The second concept is related to the technical nodes used to generate the events (for example sensors in our experiment). A second perspective would be to align the existence of a communication link between technical nodes with a dependency between event logs. The objective of a tool enabling the supervisor to easily monitor the activities of a system user remains the same. Currently, the system is used offline and a perspective is to extend it for data streams.




(a) Expected occurrence of a critical sequence




(b) Anomaly detected for a critical sequence

Figure A.6 – Normal and abnormal critical sequences

# Appendix B

# Activity pattern visualization

In order to help the supervisor (physician, caregiver, family, user him/herself) get a better grasp on the activities and habits giving its tempo to the life in the smart home, it is necessary to provide tools for the visualization of the data and the extracted patterns. Initial attempts were for instance proposed as part of the CASAS project (Chen and Dawadi, 2011; Thomas and Crandall, 2011).

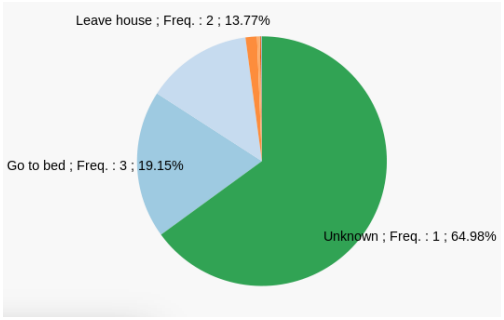
Over the course of this PhD, we have used graphical representations adapted from standard data visualization schemes. These visual tools were gathered in a web-platform, accessible from computers, tablets and cell phones. In particular, the service contains the following visual representations:

- The status table (figure B.1a) allows the quick verification of the last time an activity occurred. The color of the dots could for instance depend on the regularity of the corresponding episode. Such a figure could help reassure the family by showing that the ADLs are carried out regularly.
- Standard pie charts (figure B.1b) help assess the time spent on each ADL, or the time spent in different area of the house.
- The frequent succession of the activities can be viewed thanks to cord diagrams (figure B.1c) or tree maps (figure B.1d). In the cord diagram, the ADLs are represented around the wheel, their width being proportional to their occurrence count. The cords link each activity to the activities that are performed right after. The color of the cord

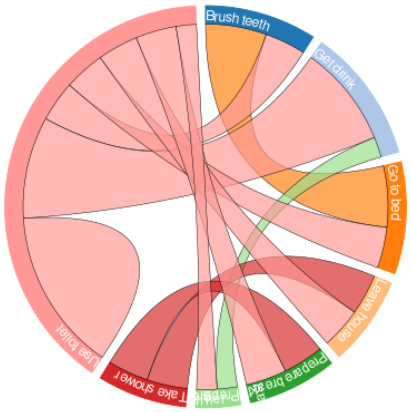
defines the reading order: it is the same color as the first activity in the sequence, and the width of the cord represents the frequency of the sequence. The tree map represent the same information: each box corresponds to a sequence, and the surface of the boxes is proportional to the frequency of the sequence.

- Heat maps are traditional representations for visualizing spatio-temporal data. Figure B.1e allows for instance to assess the solicitation of the different areas in the house for the ADLs (the presence of red in the blue circles indicates a greater occurrence frequency)
- The clock representation (figure B.1f) summarizes the periodic patterns: it shows, for each activity, the times at which the activity is likely to occur. The opacity of the color depends on the accuracy of the corresponding component.

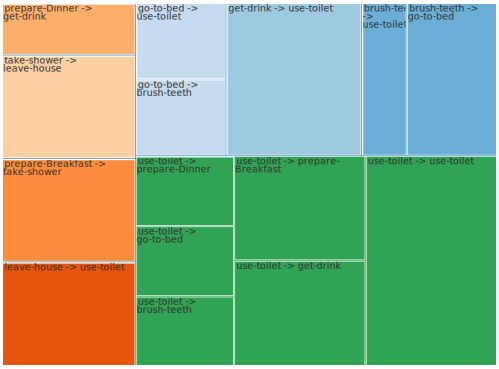

(a) Summary status table



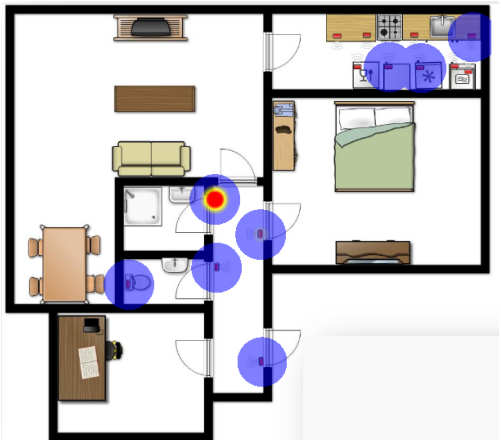

(b) Pie chart



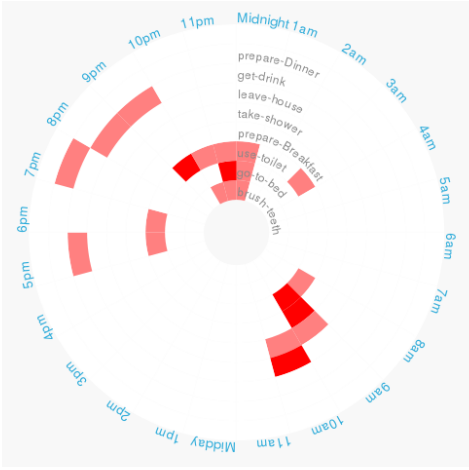

(c) Cord diagram




(d) Tree map



(e) Heat map




(f) Clock heat map

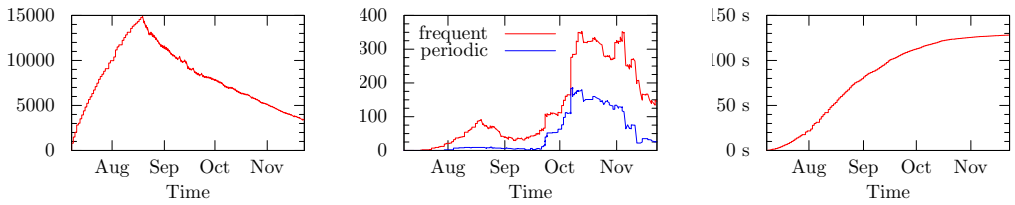
Figure B.1 – Data visualization proposal for a user-oriented platform

# Appendix C

# Experiment: sxED on the Travian game dataset

Travian is a web-browser game, where players, organized into alliances, fight for the fulfilling of objectives and the control of territories. The game company releases each day a snapshot of the server status: it contains information on the players (villages, alliance membership). These daily updates were collected for the 2014 fr5 game round, from July 8$^{\text{th}}$ to November 23$^{\text{rd}}$. We focus here on the players alliance shifts: the event labels look like "*Player P [joined|left] alliance A*". 27674 such events are recorded, but most labels are rare (25985 labels).

The dataset was processed with a period of one week, a window $T_W$ = six weeks, a minimal support $S_{min} = 5$ and a maximal episode duration $T_{ep} = 1$ day. Figure C.1 presents the evolution of the window size, episode counts, and execution time during the mining. The results are fairly different from those




(a) Event count in the window

(b) Interesting episode count

(c) Cumulative execution time

Figure C.1 – Execution log for the Travian fr5 alliance membership dataset

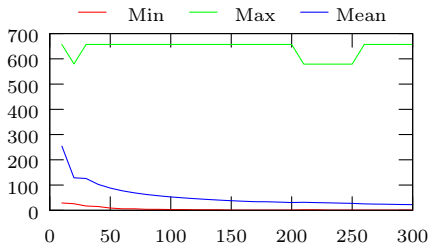
of the home automation dataset, but were explained by a player (picturing a domain expert). During the first 6 weeks, the window fills rapidly with events: new players register onto the game, and the diplomacy begins. The players join or switch alliances. After the 6 weeks, the event count in the window decreases with time. Several explanations: (i) the opening of a new game round (on August 22$^{\text{nd}}$) slowed down the number of new player registrations (players tend to join the most recent game round); (ii) most players have found an alliance they like: they stop changing alliances. Until October, little frequent and periodic patterns are detected, but their number increases rapidly after that. The periodic episodes discovered in the Sep, 18$^{\text{th}}$ – Oct, 30$^{\text{th}}$ (maximal count of periodic episodes) contain notably:

- *{1SixCentDix8 left Vétérans, 1SixCentDix8 joined iChiefs}*: 8 MO, 2 components, $\mu_1$ = Fri. 0:00, $\mu_2$ = Mon. 0:00, $\sigma_1 = \sigma_2 = 0$, $A = 80\%$
- *{1SixCentDix8 left iChiefs, 1SixCentDix8 joined Vétérans}*: 8 MO, 2 components, $\mu_1$ = Sat. 0:00, $\mu_2$ =Tue. 0:00, $\sigma_1 = \sigma_2 = 0$, $A = 80\%$
- *{Jill left Bakka, Jill joined LI}*: 10 MO, 2 components, $\mu_1$ = Mon. 18:00, $\sigma_1$ = 1 day, $\mu_2$ = Fri. 0:00, $\sigma_2 = 0$, $A = 75\%$

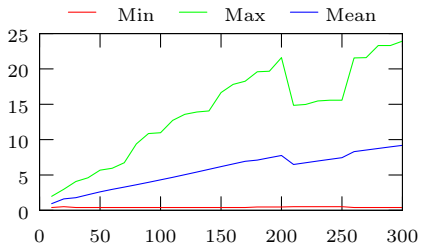
Some players periodically change of alliance: *1SixCentDix8* leaves *Vétérans* for *iChiefs* on Mondays and Fridays, and goes back to *Vétérans* one day later. *Jill* goes from *Bakka* to *LI* either on Mondays or Tuesdays, as well as on Fridays. This actually highlights a strategy allied alliances (*iChiefs* and *Vétérans* on one side, and *Bakka* and *LI* on the other side) have developed to share with one another the effects of artifacts owned by players *1SixCentDix8* and *Jill*, respectively.

# Appendix D

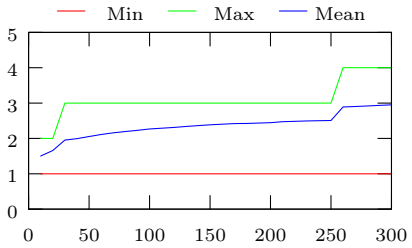
# Full results for TKRES, on the Aruba datasets (sensors and activities)



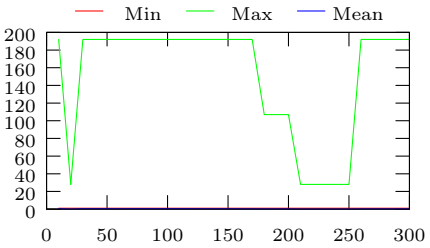

(a) Support




(b) Regularity

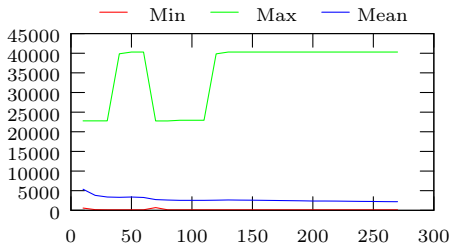



(c) Length

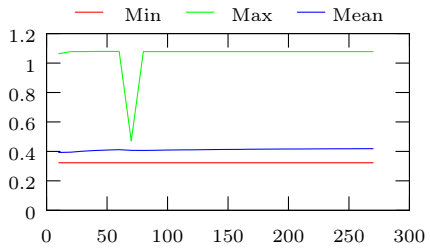



(d) Stability

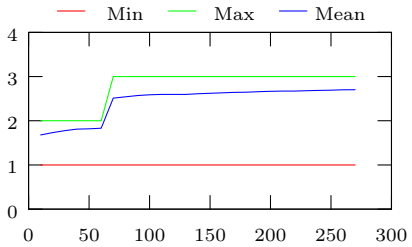
Figure D.1 – Influence of $k$ on Aruba activities



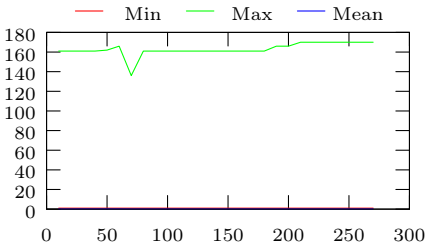

(a) Support




(b) Regularity

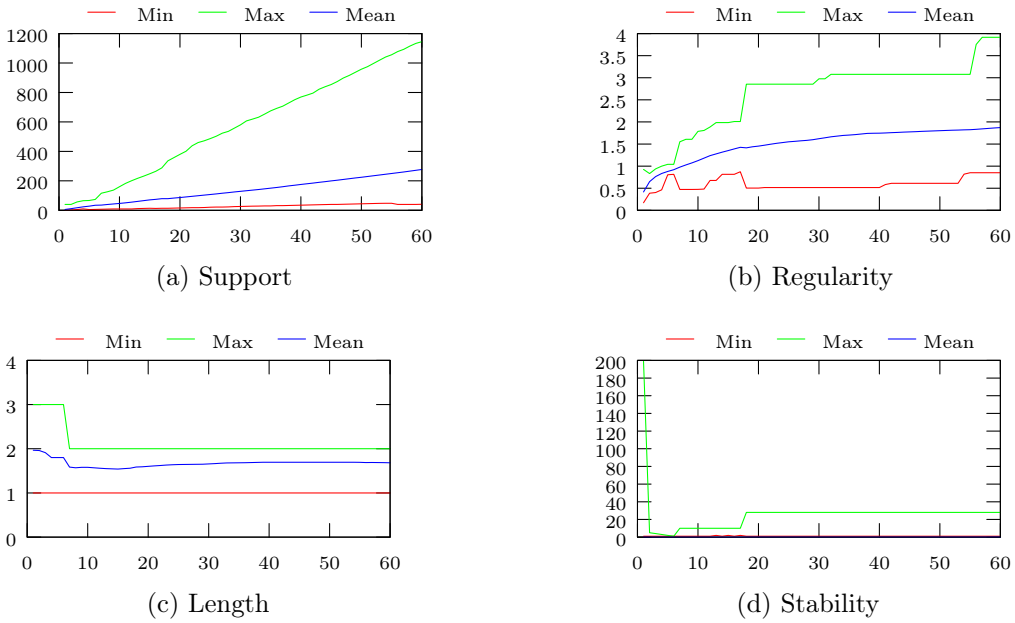



(c) Length




(d) Stability

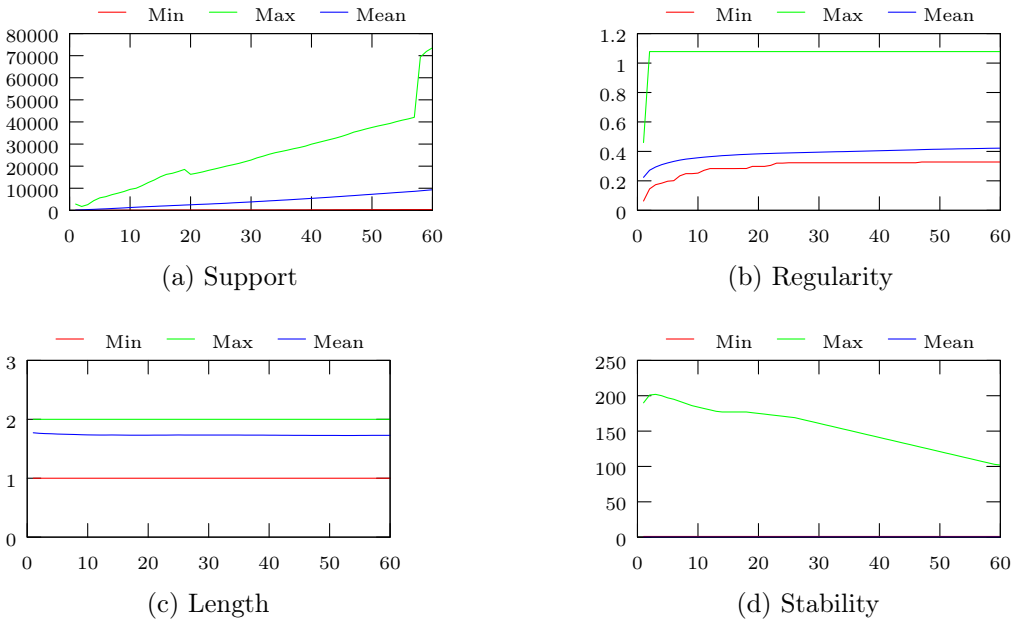
Figure D.2 – Influence of k on Aruba sensors




(a) Support (b) Regularity

(c) Length (d) Stability

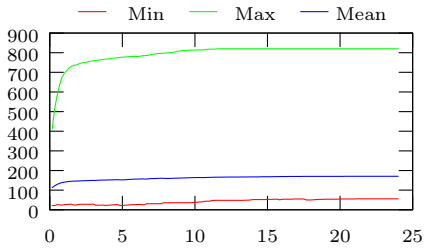
Figure D.3 – Influence of $m$ on Aruba activities



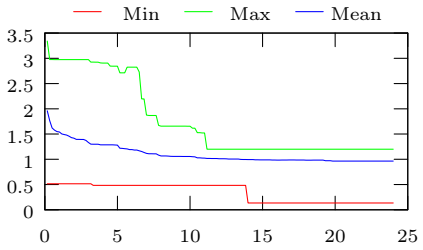

(a) Support (b) Regularity

(c) Length (d) Stability

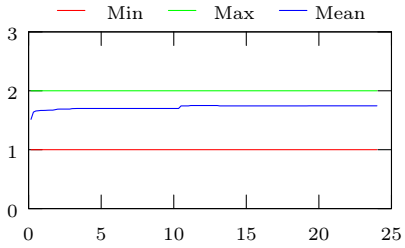
Figure D.4 – Influence of m on Aruba sensors



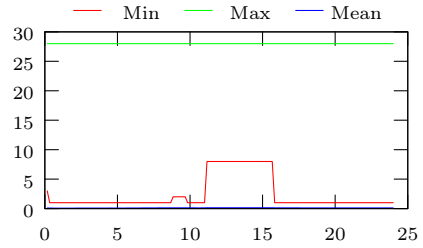

(a) Support




(b) Regularity

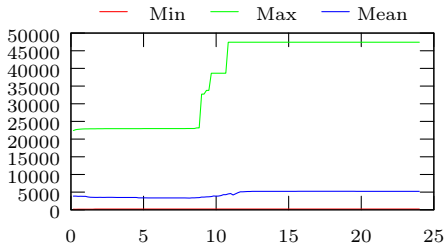



(c) Length

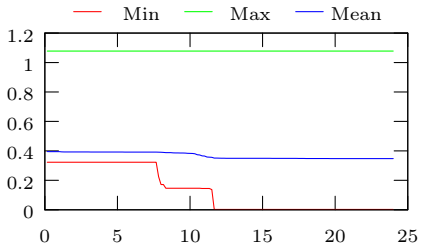



(d) Stability

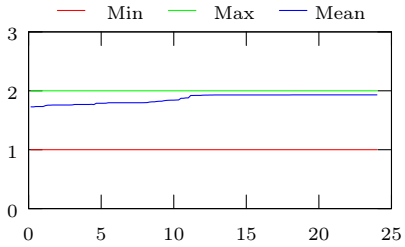
Figure D.5 – Influence of $T_{ep}$ on Aruba activities



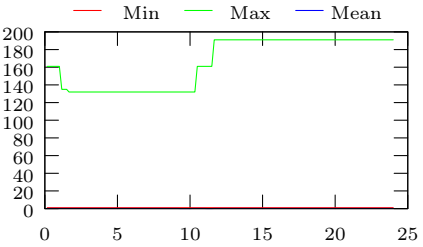

(a) Support




(b) Regularity




(c) Length




(d) Stability

Figure D.6 – Influence of $T_{ep}$ on Aruba sensors

# Bibliography

Acampora, G., Cook, D. J., Rashidi, P., and Vasilakos, A. V. (2013). A survey on ambient intelligence in healthcare. *Proceedings of the IEEE*, 101(12):2470–2494.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected Papers of Hirotugu Akaike*, pages 199–213. Springer.

Allègre, W., Burger, T., Berruet, P., and Antoine, J.-Y. (2012). A non-intrusive monitoring system for ambient assisted living service delivery. In Donnelly, M. P., Paggetti, C., Nugent, C. D., and Mokhtari, M., editors, *International Conference on Smart Homes and Health Telematics (ICOST)*, volume 7251 of *Lecture Notes in Computer Science*, pages 148–156. Springer.

Aloulou, H., Mokhtari, M., Tiberghien, T., Biswas, J., and Yap, P. L. K. (2014). An adaptable and flexible framework for assistive living of cognitively impaired people. *IEEE Journal of Biomedical and Health Informatics*, 18(1):353–360.

Amphawan, K. and Lenca, P. (2013). Mining top-k frequent/regular patterns based on user-given trade-off between frequency and regularity. In Papasratorn, B., Charoenkitkarn, N., Vanijja, V., and Chongsuphajaisiddhi, V., editors, *International Conference on Advances in Information Technology (IAIT)*, volume 409 of *Communications in Computer and Information Science*, pages 1–12. Springer.

Amphawan, K. and Lenca, P. (2015). Mining top-k frequent-regular closed patterns. *Expert Systems with Applications*, 42(21):7882–7894.

Amphawan, K., Lenca, P., and Surarerks, A. (2011). Efficient mining top-k regular-frequent itemset using compressed tidsets. In Cao et al. (2012), pages 124–135.

Amphawan, K., Soulas, J., and Lenca, P. (2015). Mining top-k regular episodes from sensor streams. In Science, P. C., editor, *International Conference on Advances in Information Technology (IAIT)*, number 69, pages 76–85.

Avci, U. and Passerini, A. (2013). A fully unsupervised approach to activity discovery. In Salah, A. A., Hung, H., Aran, O., and Gunes, H., editors, *International Workshop on Human Behavior Understanding (HBU)*, volume 8212 of *Lecture Notes in Computer Science*, pages 77–88. Springer.

Baratchi, M., Meratnia, N., and Havinga, P. J. M. (2013). Recognition of periodic behavioral patterns from streaming mobility data. In Stojmenovic, I., Cheng, Z., and Guo, S., editors, *Mobile and Ubiquitous Systems: Computing, Networking, and Services (MOBIQUITOUS)*, volume 131 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 102–115. Springer.

Benoit, M., Robert, P., Staccini, P., Brocker, P., Guérin, O., Lechowshi, L., and Vellas, B. (2005). One-year longitudinal evaluation of neuropsychiatric symptoms in alzheimer's disease. the real.fr study. *The Journal of Nutrition, Health & Aging*, 9(2):95–99.

Bergua, V., Bouisson, J., Dartigues, J.-F., Swendsen, J., Fabrigoule, C., Pérès, K., and Pascale, B.-G. (2013). Restriction in instrumental activities of daily living in older persons: Association with preferences for routines and psychological vulnerability. *International Journal of Aging and Human Development*, 4(77):309–329.

Cao, L., Huang, J. Z., Bailey, J., Koh, Y. S., and Luo, J., editors (2012). *New Frontiers in Applied Data Mining (PAKDD) Workshops*, volume 7104 of *Lecture Notes in Computer Science*. Springer.

Casas-Garriga, G. (2003). Discovering unbounded episodes in sequential data. In Lavrac, N., Gamberger, D., Blockeel, H., and Todorovski, L., editors, *European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, volume 2838 of *Lecture Notes in Computer Science*, pages 83–94. Springer.

Chen, C. and Dawadi, P. (2011). Casasviz: Web-based visualization of behavior patterns in smart environments. In *IEEE International Conference on Pervasive Computing and Communications (PerCom) Workshop Proceedings*, pages 301–303. IEEE.

Chen, J., Kam, A. H., Zhang, J., Liu, N., and Shue, L. (2005). Bathroom activity monitoring based on sound. In Gellersen et al. (2005), pages 47–61.

Chernbumroong, S., Cang, S., Atkins, A. S., and Yu, H. (2013). Elderly activities recognition and classification for applications in assisted living. *Expert Systems with Applications*, 40(5):1662–1674.

Cleland, I., Han, M., Nugent, C. D., Lee, H., Zhang, S., McClean, S. I., and Lee, S. (2013). Mobile based prompted labeling of large scale activity data. In Nugent, C. D., Coronato, A., and Bravo, J., editors, *International Work-Conference on Ambient Assisted Living and Active Aging (IWAAL)*, volume 8277 of *Lecture Notes in Computer Science*, pages 9–17. Springer.

Cook, D. J. (2012). Learning setting-generalized activity models for smart spaces. *IEEE Intelligent Systems*, 27(1):32–38.

Cook, D. J., Crandall, A. S., Thomas, B. L., and Krishnan, N. C. (2013a). Casas: A smart home in a box. *IEEE Computer*, 46(7):62–69.

Cook, D. J., Krishnan, N. C., and Rashidi, P. (2013b). Activity discovery and activity recognition: A new partnership. *IEEE Transactions on Cybernetics*, 43(3):820–828.

Cook, D. J. and Schmitter-Edgecombe, M. (2009). Assessing the quality of activities in a smart environment. *Methods of information in medicine*, 48(5):480–485.

Cook, D. J., Youngblood, G. M., III, E. O. H., Gopalratnam, K., Rao, S., Litvin, A., and Khawaja, F. (2003). Mavhome: An agent-based smart home. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 521–524. IEEE Computer Society.

Dawadi, P., Cook, D. J., and Schmitter-Edgecombe, M. (2015). Automated cognitive health assessment from smart home-based behavior data. *IEEE Journal of Biomedical and Health Informatics*, PP(99):1–1.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38.

Ester, M., Kriegel, H., Sander, J., and Xu, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In Simoudis et al. (1996), pages 226–231.

Fleury, A., Vacher, M., and Noury, N. (2010). SVM-based multimodal classification of activities of daily living in health smart homes: sensors, algorithms, and first experimental results. *IEEE Transactions on Information Technology in Biomedicine*, 14(2):274–283.

Fogarty, J., Au, C., and Hudson, S. E. (2006). Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition. In Wellner, P. and Hinckley, K., editors, *ACM Symposium on User Interface Software and Technology (UIST)*, pages 91–100. ACM.

Gama, J. (2012). A survey on learning from data streams: current and future trends. *Progress in Artificial Intelligence*, 1(1):45–55.

Gan, M. and Dai, H. (2014). Detecting and monitoring abrupt emergences and submergences of episodes over data streams. *Information Systems*, 39:277–289.

Gellersen, H., Want, R., and Schmidt, A., editors (2005). *International Conference on Pervasive Computing (PERVASIVE)*, volume 3468 of *Lecture Notes in Computer Science*. Springer.

Glynn, E. F., Chen, J., and Mushegian, A. R. (2006). Detecting periodic patterns in unevenly spaced gene expression time series using lomb-scargle periodograms. *Bioinformatics*, 22(3):310–316.

Gu, T., Wu, Z., Tao, X., Pung, H. K., and Lu, J. (2009). epSICAR: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pages 1–9. IEEE Computer Society.

Hamid, R., Maddi, S., Johnson, A. Y., Bobick, A. F., Essa, I. A., and Jr., C. L. I. (2009). A novel sequence representation for unsupervised analysis of human activities. *Artificial Intelligence*, 173(14):1221–1244.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In Chen, W., Naughton, J. F., and Bernstein, P. A., editors, *ACM SIGMOD International Conference on Management of Data*, pages 1–12. ACM.

Heierman, III, E. O., Youngblood, G. M., and Cook, D. J. (2004). Mining temporal sequences to discover interesting patterns. In *KDD Workshop on mining temporal and sequential data*.

Helal, S. and Chen, C. (2009). The gator tech smart house: Enabling technologies and lessons learned. In *International Convention on Rehabilitation Engineering & Assistive Technology (i-CREATE)*, pages 13:1–13:4. ACM.

Huynh, T., Fritz, M., and Schiele, B. (2008). Discovery of activity patterns using topic models. In Youn, H. Y. and Cho, W., editors, *International Conference on Ubiquitous Computing (UbiComp)*, volume 344 of *ACM International Conference Proceeding Series*, pages 10–19. ACM.

Jakkula, V. R., Crandall, A. S., and Cook, D. J. (2009). Enhancing anomaly detection using temporal pattern discovery. In Kameas, A. D., Callagan, V., Hagras, H., Weber, M., and Minker, W., editors, *Advanced Intelligent Environments*, pages 175–194. Springer US.

Katz, S. (1983). Assessing self-maintenance: Activities of daily living, mobility, and instrumental activities of daily living. *Journal of the American Geriatrics Society*, 31:721–727.

Kidd, C. D., Orr, R. J., Abowd, G. D., Atkeson, C. G., Essa, I. A., MacIntyre, B., Mynatt, E. D., Starner, T., and Newstetter, W. (1999). The aware home: A living laboratory for ubiquitous computing research. In Streitz, N. A., Siegel, J., Hartkopf, V., and Konomi, S., editors, *International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture (CoBuild)*, volume 1670 of *Lecture Notes in Computer Science*, pages 191–198. Springer.

Kim, E., Helal, S., and Cook, D. J. (2010). Human activity recognition and pattern discovery. *IEEE Pervasive Computing*, 9(1):48–53.

Kim, E., Helal, S., Nugent, C. D., and Beattie, M. (2015). Analyzing activity recognition uncertainties in smart home environments. *ACM Transactions on Intelligent Systems and Technology*, 6(4):52.

Kiran, R. U. and Reddy, P. K. (2010). Mining periodic-frequent patterns with maximum items' support constraints. In Shyamasundar, R. K. and Deshpande, P., editors, *Bangalore Annual Compute Conference*, pages 1:1–1:8. ACM.

Kotchera, A., Straight, A., and Guterbock, T. (2005). Beyond 50.05: A report to the nation on livable communities: Creating environments for successful aging. Technical report, AARP, Washington, D.C.

Lahiri, M. and Berger-Wolf, T. Y. (2008). Mining periodic behavior in dynamic social networks. In *IEEE International Conference on Data Mining (ICDM)*, pages 373–382. IEEE Computer Society.

Law, M. and Letts, L. (1989). A critical review of scales of activities of daily living. *The American journal of occupational therapy*, 43:522–528.

Li, F. and Dustdar, S. (2011). Incorporating unsupervised learning in activity recognition. In *Activity Context Representation: Techniques and Languages, AAAI Workshop*, volume WS-11-04 of *AAAI Workshops*. AAAI.

Li, Z., Han, J., Ding, B., and Kays, R. (2012). Mining periodic behaviors of object movements for animal and biological sustainability studies. *Data Mining and Knowledge Discovery*, 24(2):355–386.

Lin, S., Qiao, J., and Wang, Y. (2014). Frequent episode mining within the latest time windows over event streams. *Applied Intelligence*, 40(1):13–28.

Luštrek, M. and Kaluza, B. (2009). Fall detection and activity recognition with machine learning. *Informatica (Slovenia)*, 33(2):197–204.

Luštrek, M., Kaluza, B., Cvetkovic, B., Dovgan, E., Gjoreski, H., Mirchevska, V., and Gams, M. (2012). Confidence: Ubiquitous care system to support independent living. In Raedt, L. D., Bessière, C., Dubois, D., Doherty, P., Frasconi, P., Heintz, F., and Lucas, P. J. F., editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 1013–1014. IOS Press.

Mannila, H. and Toivonen, H. (1996). Discovering generalized episodes using minimal occurrences. In Simoudis et al. (1996), pages 146–151.

Mannila, H., Toivonen, H., and Inkeri Verkamo, A. (1997). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–289.

Mannila, H., Toivonen, H., and Verkamo, A. I. (1995). Discovering frequent episodes in sequences. In Fayyad, U. M. and Uthurusamy, R., editors, *International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 210–215. AAAI Press.

Nazerfard, E., Das, B., Holder, L. B., and Cook, D. J. (2010a). Conditional random fields for activity recognition in smart environments. In Veinot, T. C., Çatalyürek, Ü. V., Luo, G., Andrade, H., and Smalheiser, N. R., editors, *ACM International Health Informatics Symposium*, pages 282–286. ACM.

Nazerfard, E., Rashidi, P., and Cook, D. J. (2010b). Discovering temporal features and relations of activity patterns. In Fan, W., Hsu, W., Webb, G. I., Liu, B., Zhang, C., Gunopulos, D., and Wu, X., editors, *ICDM Workshops*, pages 1069–1075. IEEE Computer Society.

Patel, S. N., Robertson, T., Kientz, J. A., Reynolds, M. S., and Abowd, G. D. (2007). At the flick of a switch: Detecting and classifying unique electrical events on the residential power line. In Krumm, J., Abowd, G. D., Seneviratne, A., and Strang, T., editors, *International Conference on Ubiquitous Computing (UbiComp)*, volume 4717 of *Lecture Notes in Computer Science*, pages 271–288. Springer.

Patnaik, D., Laxman, S., Chandramouli, B., and Ramakrishnan, N. (2012). Efficient episode mining of dynamic event streams. In Zaki, M. J., Siebes, A., Yu, J. X., Goethals, B., Webb, G. I., and Wu, X., editors, *IEEE International Conference on Data Mining (ICDM)*, pages 605–614. IEEE Computer Society.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Philipose, M., Fishkin, K. P., Perkowitz, M., Patterson, D. J., Fox, D., Kautz, H. A., and Hähnel, D. (2004). Inferring activities from interactions with objects. *IEEE Pervasive Computing*, 3(4):50–57.

Pigot, H. (2010). When cognitive assistance brings autonomy in daily living: the DOMUS experience. *Gerontechnology*, 9(2):71.

Pusiol, G., Brémond, F., and Thonnat, M. (2011). Unsupervised discovery, modeling, and analysis of long term activities. In Crowley, J. L., Draper, B. A., and Thonnat, M., editors, *Inernational Conference on Computer Vision Systems (ICVS)*, volume 6962 of *Lecture Notes in Computer Science*, pages 101–111. Springer.

Rashidi, P. and Cook, D. J. (2010). Mining sensor streams for discovering human activity patterns over time. In Webb et al. (2010), pages 431–440.

Rashidi, P., Cook, D. J., Holder, L. B., and Schmitter-Edgecombe, M. (2011). Discovering activities to recognize and track in a smart environment. *IEEE Transactions on Knowledge and Data Engeneering*, 23(4):527–539.

Rashidi, P. and Mihailidis, A. (2013). A survey on ambient-assisted living tools for older adults. *IEEE Journal of Biomedical and Health Informatics*, 17(3):579–590.

Rissanen, J. (1989). *Stochastic complexity in statistical inquiry theory*. World Scientific Publishing Co., Inc.

Rodner, T. and Litz, L. (2013). Data-driven generation of rule-based behavior models for an ambient assisted living system. In *IEEE International Conference on Consumer Electronics*, pages 35–38.

Rodríguez-Verjan, C., Augusto, V., Xie, X., and Buthion, V. (2013). Economic comparison between hospital at home and traditional hospitalization using a simulation-based approach. *Journal of Enterprise Information Management*, 26(1):135–153.

Roy, P., Bouzouane, A., Giroux, S., and Bouchard, B. (2011). Possibilistic activity recognition in smart homes for cognitively impaired people. *Applied Artificial Intelligence*, 25(10):883–926.

Schwarz, G. et al. (1978). Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464.

Silva, L. C. D., Morikawa, C., and Petra, M. I. (2012). State of the art of smart homes. *Engineering Applications of Artificial Intelligence*, 25(7):1313–1321.

Simonin, J., Bertin, E., Le Traon, Y., Jézéquel, J.-M., and Crespi, N. (2011). Analysis and improvement of the alignment between business and information system for telecom services. *International Journal On Advances in Software*, 4(1&2):117–128.

Simonin, J., Soulas, J., and Lenca, P. (2015). Activity monitoring process based on model-driven engineering - Application to ambient assisted living. *Journal of Intelligent Systems*, 24(3):371–382.

Simoudis, E., Han, J., and Fayyad, U. M., editors (1996). *International Conference on Knowledge Discovery and Data Mining (KDD)*. AAAI Press.

Soulas, J. and Lenca, P. (2015). Periodic episode discovery over event streams. In Pereira, F. C., Machado, P., Costa, E., and Cardoso, A., editors, *Progress in Artificial Intelligence - Portuguese Conference on Artificial Intelligence (EPIA)*, volume 9273 of *Lecture Notes in Computer Science*, pages 547–559. Springer.

Soulas, J., Lenca, P., and Thépaut, A. (2013). Monitoring the habits of elderly people through data mining from home automation devices data. In Correia, L., Reis, L. P., and Cascalho, J., editors, *Progress in Artificial Intelligence - Portuguese Conference on Artificial Intelligence (EPIA)*, volume 8154 of *Lecture Notes in Computer Science*, pages 343–354. Springer.

Soulas, J., Lenca, P., and Thépaut, A. (2015). Unsupervised discovery of activities of daily living characterized by their periodicity and variability. *Engineering Applications of Artificial Intelligence*, 45:90–102.

Stikic, M., Huynh, T., Laerhoven, K. V., and Schiele, B. (2008). ADL recognition based on the combination of RFID and accelerometer sensing. In Saranummi, N., Korhonen, I., and Wactlar, H. D., editors, *International Conference on Pervasive Computing Technologies for Healthcare (ICST)*, pages 258–263. IEEE.

Surana, A., Kiran, R. U., and Reddy, P. K. (2011). An efficient approach to mine periodic-frequent patterns in transactional databases. In Cao et al. (2012), pages 254–266.

Sölvesdotter, M., ten Berg, H., and Berleen, G. (2007). Healthy aging – A challenge for Europe – Chapter 2. Statistics. Technical report, European Commision.

Tanbeer, S. K., Ahmed, C. F., and Jeong, B. (2010a). Mining regular patterns in data streams. In Kitagawa, H., Ishikawa, Y., Li, Q., and Watanabe, C., editors, *Database Systems for Advanced Applications, DASFAA, Part I*, volume 5981 of *Lecture Notes in Computer Science*, pages 399–413. Springer.

Tanbeer, S. K., Ahmed, C. F., and Jeong, B. (2010b). Mining regular patterns in incremental transactional databases. In Han, W., Srivastava, D., Yu, G., Yu, H., and Huang, Z. H., editors, *Advances in Web Technologies and Applications APWeb*, pages 375–377. IEEE Computer Society.

Tanbeer, S. K., Ahmed, C. F., Jeong, B., and Lee, Y. (2008). Mining regular patterns in transactional databases. *IEICE Transactions on Information and Systems*, 91-D(11):2568–2577.

Tanbeer, S. K., Ahmed, C. F., Jeong, B., and Lee, Y. (2009). Discovering periodic-frequent patterns in transactional databases. In Theeramunkong, T., Kijsirikul, B., Cercone, N., and Ho, T. B., editors, *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*

*(PAKDD)*, volume 5476 of *Lecture Notes in Computer Science*, pages 242–253. Springer.

Tapia, E. M., Intille, S. S., and Larson, K. (2004). Activity recognition in the home using simple and ubiquitous sensors. In Ferscha, A. and Mattern, F., editors, *Pervasive Computing (PERVASIVE)*, volume 3001 of *Lecture Notes in Computer Science*, pages 158–175. Springer.

Tatti, N. and Cule, B. (2011). Mining closed episodes with simultaneous events. In Apté, C., Ghosh, J., and Smyth, P., editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1172–1180. ACM.

Thomas, B. and Crandall, A. (2011). A demonstration of pyviz, a flexible smart home visualization tool. In *Pervasive Computing and Communications Workshops (PERCOM Workshops), 2011 IEEE International Conference on*, pages 304–306.

Truong, T. B. T., de Lamotte, F. F., Diguet, J., and Saïd-Hocine, F. (2009). Assisted living service identification based on activity patterns. In *Joint Conferences on Pervasive Computing (JCPC)*, pages 417–422.

van der Aalst, W. M. P. (2013). Business process management: A comprehensive survey. *International Scholarly Research Notices*, 2013:1–37.

van der Aalst, W. M. P., Weijters, T., and Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engeneering*, 16(9):1128–1142.

van Kasteren, T., Englebienne, G., and Kröse, B. J. A. (2010a). An activity monitoring system for elderly care using generative and discriminative models. *Personal and Ubiquitous Computing*, 14(6):489–498.

van Kasteren, T., Englebienne, G., and Kröse, B. J. A. (2010b). Transferring knowledge of activity recognition across sensor networks. In Floréen, P., Krüger, A., and Spasojevic, M., editors, *Pervasive Computing*, volume 6030 of *Lecture Notes in Computer Science*, pages 283–300. Springer.

Webb, G. I., Liu, B., Zhang, C., Gunopulos, D., and Wu, X., editors (2010). *IEEE International Conference on Data Mining*. IEEE Computer Society.

Wilson, D. H. and Atkeson, C. G. (2005). Simultaneous tracking and activity recognition (STAR) using many anonymous, binary sensors. In Gellersen et al. (2005), pages 62–79.

Zachman, J. A. (1987). A framework for information systems architecture. *IBM Systems Journal*, 26(3):276–292.

Zhang, T. and Chen, Y. (2006). Meeting the needs of elderly people in China. *British Medical Journal*, 333(7564):363–364.

Zheng, H., Wang, H., and Black, N. D. (2008). Human activity detection in smart home environment with self-adaptive neural networks. In *IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 1505–1510. IEEE.

Zhou, W., Liu, H., and Cheng, H. (2010). Mining closed episodes from event sequences efficiently. In Zaki, M. J., Yu, J. X., Ravindran, B., and Pudi, V., editors, *Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD)*, volume 6118 of *Lecture Notes in Computer Science*, pages 310–318. Springer.

Zhu, H., Wang, P., He, X., Li, Y., Wang, W., and Shi, B. (2010). Efficient episode mining with minimal and non-overlapping occurrences. In Webb et al. (2010), pages 1211–1216.

# Résumé en français : Monitoring de l'activité via la fouille de capteurs domotiques

## Introduction générale et objectifs

Les personnes âgées vivent aujourd'hui plus longtemps, et souhaitent pour la plupart rester dans leur logement aussi longtemps que possible. Cette population est cependant plus fragile que la population générale : il leur est plus dangereux de vivre seuls. L'émergence et la diffusion de technologies de mesure et de communication permet désormais le développement de nouveaux services, et en particulier la domotique et l'hospitalisation à domicile. Ces technologies promettent de belles avancées pour le vieillissement à domicile : l'automatisation des équipements facilite les interactions de la personne âgée avec son environnement, les systèmes de communication aident à maintenir un lien avec la famille géographiquement éloignée et améliorent l'insertion sociale dans la communauté locale.

Les capteurs disséminés dans le logement enregistrent en continu une trace de l'activité. Ces traces forment des motifs porteurs d'information sur la santé et le bien-être de l'habitant. Des besoins ont donc émergé pour le monitoring à domicile, via l'exploitation de données issues de capteurs. En particulier, les objectifs en terme de monitoring incluent la détection de chutes ou de dangers, la reconnaissance des activités de la vie quotidienne, l'évaluation de l'autonomie, l'analyse des habitudes, etc.

**Objectifs** Cette thèse contribue au domaine de la fouille de données pour le monitoring de l'activité. Nous nous intéressons en particulier aux routines et

habitudes, qui sont des comportements fondamentaux de la vie quotidienne des individus, mais qui n'ont que peu été étudiés par le passé. Cette thèse se concentre sur la découverte et la description des habitudes d'individus, via la fouille des données issues de réseaux de capteurs domotiques. Nous y proposons un formalisme de description pour la caractérisation des habitudes, ainsi que plusieurs algorithmes et outils pour leur découverte automatique. L'exposé des résultats de la thèse se déroule sur six chapitres, ici résumés.

# Assistance à domicile

**Résumé** Le premier chapitre introduit le contexte socio-démographique et détaille les approches actuelles pour le monitoring des comportements dans le domicile à partir de capteurs disséminés dans la maison. Ce chapitre se concentre sur le monitoring de l'activité et les challenges liés à ce domaine.

**Contexte démographique** Le vieillissement de la population, notamment dans les pays industrialisés, change actuellement le profil démographique mondial. Les populations âgées, plus fragiles, présentent une plus forte incidence de situations de handicap et de maladies chroniques. Elles sont en général également plus isolées que les jeunes générations. Leur prise en charge est actuellement problématique. Le vieillissement à domicile offre une double réponse au problème actuel : il permet de contrôler les besoins en institution spécialisée, et apporte aux personnes âgées la satisfaction de rester chez elles.

**Activités de la vie quotidienne** Vivre dans son domicile présuppose d'être autonome, c'est à dire d'être en mesure de réaliser les *activités de la vie quotidienne* (AVQ). Ce sont les activités que l'on réalise pour assouvir nos besoins fondamentaux (se nourrir, se déplacer, etc). Monitorer l'activité permet donc de s'assurer de l'autonomie du patient suivi. Les capteurs domotiques permettent un tel suivi : l'activité réalisée au sein du logement influe sur les valeurs prises par les nombreux capteurs qui s'y trouvent (détecteur de mouvement, consommation électrique, capteur d'ouverture de porte, etc).

**Systèmes d'aide pour le vieillissement au domicile** Au cours des vingt dernières années, les initiatives exploitant des données domotiques pour l'assistance à la personnes se sont multipliées, voir notamment les revues de

la littératures de Rashidi and Mihailidis (2013) ou Acampora et al. (2013). Les objectifs des différents projets sont très variés : détection d'anomalies, reconnaissance de l'activité, évaluation de l'état de santé, tracking, etc. En terme de fouille de données, les différentes approches peuvent néanmoins être regroupées en deux grandes familles : la reconnaissance d'AVQ (classification supervisée), et la découverte de motifs d'activité (méthodes non-supervisées).

L'utilisation de la classification supervisée a déjà été largement étudiée dans le cadre de l'assistance à domicile. Elle nécessite cependant d'avoir accès à des données annotées pour la phase d'apprentissage. Mais de telles annotations sont en pratique très rarement disponibles. Dans cette thèse, nous nous concentrons donc sur des techniques non supervisées d'extraction de motifs pertinents. Les méthodes non supervisées concurrentes incluent entre autres des transpositions au domaine de techniques issues du traitement automatique du langage (Hamid et al., 2009; Huynh et al., 2008), l'utilisation de cartes auto adaptatives (Zheng et al., 2008), la découverte de règles d'association (Rodner and Litz, 2013; Jakkula et al., 2009) et la recherche de motifs fréquents (Rashidi et al., 2011).

# Monitoring des habitudes

**Résumé** Le chapitre 2 caractérise une classe de comportements humains : les habitudes. Nous y décrivons et comparons différentes plates-formes de monitoring des habitudes, et constatons notamment que la majorités des approches de la littérature présentent un manque majeur : elles ne prennent pas suffisamment en compte de la variabilité inhérente à la vie humaine. Nous proposons donc un nouveau modèle de périodicité pour la description des habitudes. Ce modèle est non seulement robuste vis-à-vis de la variabilité, mais la caractérise également. Il s'agit donc d'un outil de description puissant, par la suite utilisé dans les contributions décrites dans les chapitres 3 et 4.

**Caractérisation des habitudes** Les habitudes peuvent en général être décrites via des phrases du type "Si `<condition>`, alors la personne `<réalise une action>`". Dans cette thèse, nous nous intéressons aux habitudes liées à un contexte temporel. De telles habitudes peuvent être décrites ainsi : "Chaque `<période>`, aux environs de `<position temporelle au sein de la période>`, l'utilisateur `<réalise une action>`".

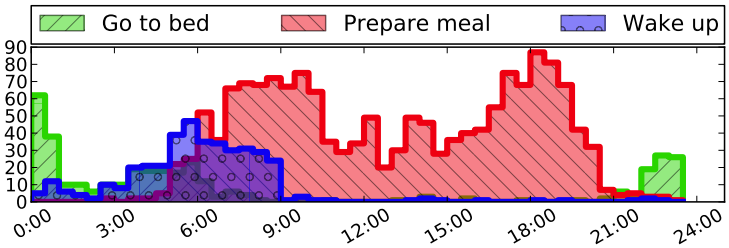



FIGURE a – Histogramme présentant la répartition temporelle de trois AVQ classiques : manger, se lever, et aller se coucher. Ces activités sont extraites du jeu de données CASAS Aruba, utilisé par la suite avec tous les algorithmes.

Le rythme auquel les habitudes ont lieu (la *période*) dépend de l'activité : les repas ont lieu quotidiennement, les courses sont faites chaque semaine, les visites chez le médecins une à deux fois par mois par exemple. Pour caractériser la *périodicité* de ces habitudes, il est intéressant d'observer la répartition temporelle des occurrences. La figure a présente un histogramme des occurrences de trois AVQ classiques : se lever, manger, et se coucher. Il apparait en particulier que :

- Il peut y avoir plusieurs *clusters* d'occurrences au sein de la période analysée, que l'on appelle par la suite *composantes* de la périodicité ;
- Chaque composante a ses propres moyenne et écart type ;
- Les composantes n'expliquent pas l'intégralité des données : certaines occurrences ont lieu en dehors des zones denses

**Mesures de périodicité** Plusieurs tentatives de description de la périodicité ont été proposées dans la littérature. On peut les classer dans trois catégories :

- La *régularité* (Tanbeer et al., 2009; Amphawan et al., 2011) mesure l'intervalle maximal entre deux occurrences successives d'une activité. C'est une mesure simple et d'interprétation aisée. Elle est en général peu impactée par la variabilité temporelle des heures d'occurrences, mais reste sensible aux occurrences manquantes. Nous utilisons cette mesure dans TKRES, la proposition décrite au chapitre 5.

- Les *cycles d'intervalles* (Heierman et al., 2004; Lahiri and Berger-Wolf, 2008) décrivent les intervalles entre les occurrences successives. Cette classe de descriptions est très sensible à la variabilité dans les données, et donc peu adaptée à la description de comportements humains.
- *Distributions de probabilité* (Nazerfard et al., 2010b; Li et al., 2012; Baratchi et al., 2013) : Ces descriptions associent à chaque instant une probabilité d'occurrence d'une activité. Ces descriptions sont en général moins impactées par la présence d'outliers dans les données sur lesquelles elles sont construites. Les propositions décrites dans les chapitres 2, 3 et 4 sont basées sur ce type de périodicité.

**Description comme loi mélange de Gaussiennes** Nous estimons qu'une description à la fois simple et expressive de la périodicité tient dans la description des composantes d'occurrences qui émergent dans les histogrammes de répartition temporelle. Nous proposons d'assimiler chaque cluster à une composante Gaussienne, décrite par sa moyenne (*quand* l'habitude a lieu) et son écart type (comment l'heure habituelle d'occurrence *varie*).

$$M_E^T = \{(\mu_1, \sigma_1), \cdots, (\mu_m, \sigma_m)\}$$

Une telle description a des propriétés à la fois descriptives (une observation correspond-elle à la périodicité attendue ?) et prédictives (quand s'attend-on à observer la prochaine occurrence ?). Une occurrence est attendue, si elle a lieu dans un intervalle de $\pm a \cdot \sigma_i$ autour de la date $\mu_i$ où $(\mu_i, \sigma_i)$ est l'une des composantes du modèle, et $a$ est un paramètre choisi par l'utilisateur (typiquement, $a = 2$). On appelle par la suite *précision* (*accuracy* dans la version originale de la dissertation) la proportion des occurrences attendues qui sont effectivement observées. On distingue ainsi les habitudes (bonne précision) des autres comportements, dont la précision est plus faible.

Les chapitres suivants proposent des stratégies pour le calcul de la périodicité des comportements et l'extraction des habitudes.

# Découverte de motifs périodiques dans des données statiques

**Résumé** Le chapitre 3 décrit l'algorithme *extended Episode Discovery* (xED) (Soulas et al., 2013, 2015). Cet algorithme, non-supervisé, découvre les mo-

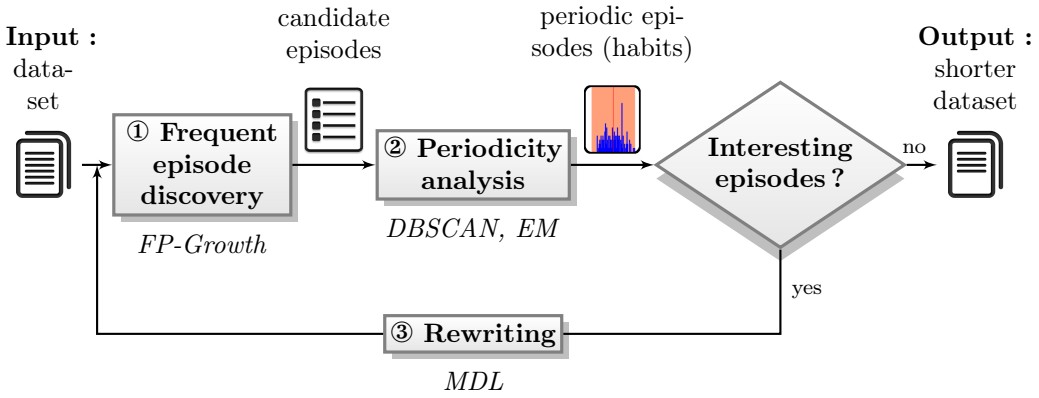



FIGURE b – xED : fonctionnement général

tifs périodiques (les habitudes) dans une base statique d'événements. xED utilise le formalisme proposé dans le chapitre 2 pour la caractérisation de la périodicité des épisodes. xED est évalué sur des jeux de données réels issus de la littérature et est comparé à un algorithme concurrent.

**Formalismes** Les données sont une séquence d'*événements*, où chaque événement est décrit par un *timestamp* (une date) et un *label* (une description, en général textuelle, de ce qui a lieu). Les habitudes sont recherchées sous la forme d'*épisodes*, c'est à dire de collections de labels qui apparaissent périodiquement ensemble. Une occurrence d'un épisode $E = \{e_1, \cdots e_n\}$ correspond à l'observation des labels qui composent $E$ au cours d'une période $T_{ep}$. La périodicité de $E$ est calculée à partir des dates d'occurrence de $E$.

**Fonctionnement général** L'algorithme proposé dans le chapitre 3, xED, est inspiré du fonctionnement de l'algorithme Episode Discovery (ED, Heierman et al. (2004)). ED vise également la découverte de motifs périodiques dans une séquence d'événements. La structure globale pour la recherche des épisodes est préservée entre les deux algorithmes : fonctionnement itératif en trois étapes. En revanche, le fonctionnement intrinsèque de chaque étape diffère fortement : le modèle de périodicité est différent. ED recherche des cycles d'intervalles qui se répètent, xED recherche un mélange de Gaussiennes avec une forte précision.

La figure b présente le fonctionnement global de xED. Dans un premier

temps, la séquence d'événements est parcourue avec une fenêtre glissante, et le contenu de la fenêtre glissante à chaque étape forme une transaction (un ensemble de labels qui ont lieu dans un même intervalle de temps). Un algorithme de recherche d'itemsets fréquents (ici FP-Growth, Han et al. (2000)) peut alors trouver les épisodes (itemsets) candidats. La périodicité de chaque candidat est alors analysée : un modèle mélange est appris pour chaque épisode. Les périodicités les plus convaincantes sont alors sélectionnées. Les deux étapes-clés pour la découverte des habitudes sont l'analyse de périodicité, et la sélection des épisodes périodiques.

**Caractérisation de la périodicité** La première étape est la détermination du nombre de composantes, via le clustering des dates d'occurrence avec DBSCAN (Ester et al., 1996) : chaque cluster dans DBSCAN correspond à une composante périodique.

Les caractéristiques des composantes sont ensuite apprises grâce à un algorithme d'Espérance-Maximisation (EM, Dempster et al. (1977)). Ce processus itératif assigne chaque date observée à la composante qui la décrit le mieux. Les caractéristiques des composantes sont alors mises à jour pour maximiser la vraisemblance de l'assignation. Le processus est répété jusqu'à convergence. Il est alors aisé de calculer la précision de la description.

**Sélection des épisodes intéressants** Les épisodes sont sélectionnés en se basant sur le principe de longueur de description minimale (Rissanen, 1989). Ce principe, issu de la théorie de l'information, précise que le meilleur encodage est le plus court : une telle description capture et exploite les caractéristiques saillantes du message. Les épisodes périodiques sont donc ici utilisés pour réécrire les données sous une forme plus compacte : un en-tête est inséré, décrivant l'épisode et sa périodicité. Toutes les occurrences de l'épisode qui correspondent à la périodicité attendue sont retirées des données, et un événement spécial marquant l'absence d'une occurrence est inséré aux dates où une occurrence attendue n'a pas été observée. On obtient ainsi un jeu de données équivalent aux données initiales, mais mettant en avant la périodicité des épisodes concernés. Les épisodes permettant la plus grande compression sont donc les plus intéressants.

**Évaluation** L'évaluation de xED sur différents jeux de données issus de benchmarks classiques de l'assistance à domicile montre les capacités de xED

à découvrir des motifs périodiques, que l'on peut directement relier aux AVQ recherchés : en particulier, les routines du matin et du soir sont particulièrement marquées. La comparaison avec ED est également probante : xED explore plus efficacement l'espace des solutions, et génère des descriptions d'habitudes plus informatives pour l'utilisateur final.

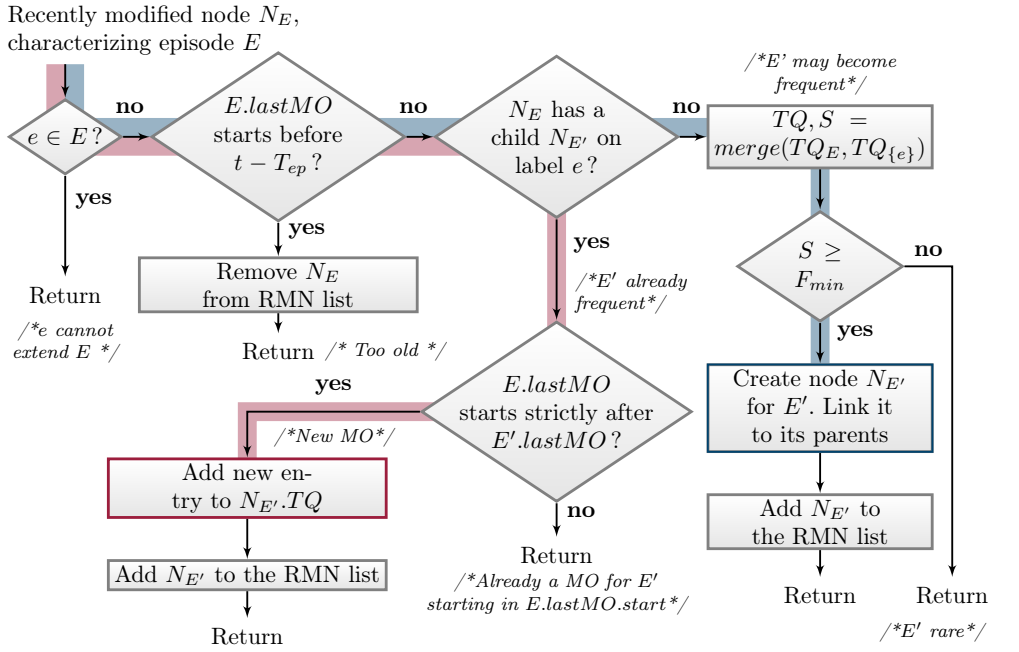
## Découverte de motifs périodiques fréquents dans des flots d'événements

**Résumé** Le chapitre 4 poursuit des développements initiés avec xED et propose des adaptations pour la gestion de flots de données. L'algorithme proposé, sxED (*streaming xED*, Soulas and Lenca (2015)) permet la découverte d'habitudes, mais surtout la mise à jour des habitudes au fur et à mesure que de nouvelles mesures arrivent et la détection de nouveaux motifs. sxED est évalué sur des benchmarks réels et comparé à xED.

**Évolution des habitudes** xED considère les habitudes comme des comportements périodiques stables sur la durée de l'expérimentation. Dans la pratique, les habitudes évoluent. Les changements d'habitudes peuvent aussi être des facteurs d'inquiétude car souvent observés quand les maladies neurodégénératives évoluent. Il est donc nécessaire de prendre en compte, voire de détecter une évolution dans les habitudes. En terme de fouille de données, cela se traduit par la nécessité de gérer des flots d'événements, ce qui pose des contraintes sur les algorithmes utilisés. La contrainte la plus marquante est qu'ils ne peuvent pas avoir accès à l'historique des observations. Ils ne peuvent de plus pas supposer que la distribution des données est stable.

**Fonctionnement général de sxED** Afin de détecter et monitorer les habitudes dans un flot d'événements, nous proposons sxED, qui maintient à jour les motifs périodiques dans le passé récent. sxED utilise une fenêtre glissante pour repérer la période d'intérêt, et exploite des structures de données dédiées pour parcourir et mettre à jour efficacement les épisodes périodiques à chaque nouvelle observation. sxED est composé de deux étapes principales : (i) la recherche des épisodes fréquents (les candidats), et (ii) l'analyse de périodicité. Pour chacune de ces étapes, une contribution est proposée.

**Algorithm a** Mise à jour d'un noeud récemment modifié avec un nouvel événement $(e, t)$. Le chemin rouge correspond à l'observation d'une nouvelle occurrence minimale, le bleu à la découverte d'un nouvel épisode fréquent.




**Recherche d'épisodes fréquents** Nous définissons le support comme le nombre d'occurrences minimales (Mannila and Toivonen, 1996) disjointes. Cette mesure est monotone : le support d'un épisode est inférieur à celui de ses sous-episodes. Afin de faciliter la recherche et la mise à jour des épisodes fréquents, nous proposons une structure de treillis, dont chaque nœud décrit un épisode et sa file d'occurrence (*time queue*, en version originale). La time queue précise les intervalles couverts par les occurrences minimales. Elle permet le calcul du support, et se met à jour incrémentalement.

Une autre information permet l'intégration rapide de nouvelles données : la connaissance des nœuds mis à jour récemment. En effet, seuls les descendants de ces nœuds sont susceptibles d'être mis à jour, ce qui réduit très considérablement l'espace de recherche. La mise à jour du treillis quand un nouvel événement arrive se fait alors en partant des nœuds récemment modifiés, en suivant l'algorithme a

**Mise à jour de la périodicité des épisodes** Quand la time queue d'un épisode change, sa périodicité est susceptible de changer également. Il suffit

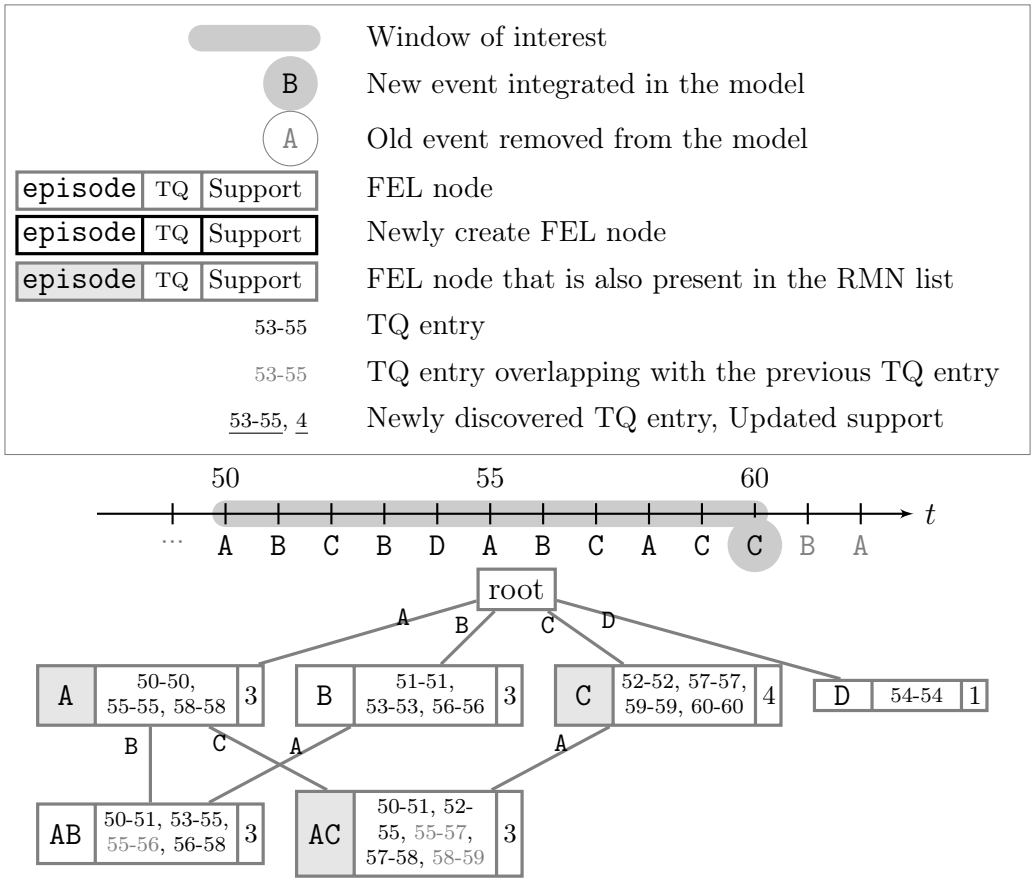
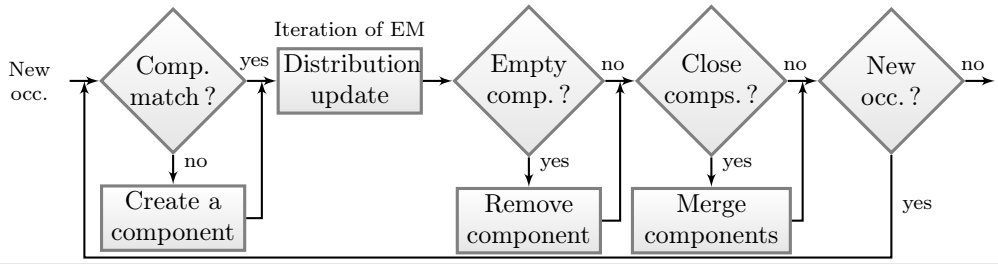



FIGURE c – Treillis des épisodes fréquents pour un jeu de données fictif (donné en milieu de figure), quand (C, 60) est le dernier événement en date. La boite du haut décrit les conventions utilisées dans le treillis (partie basse de la figure).

**Algorithm b** Stratégie générale pour la mise à jour de la périodicité




dans le cas général simplement de réaliser une itération supplémentaire de Espérance-Maximisation, en utilisant la nouvelle time queue : les habitudes changent en général peu. Mais il peut parfois être également nécessaire de faire évoluer le nombre de composantes dans la loi mélange décrivant la périodicité. Nous proposons à cet effet des heuristiques permettant l'ajout de composantes quand les observations ne correspondent pas aux attentes, et la fusion de composantes existantes similaires (figure b).

**Évaluation de sxED** L'évaluation est réalisée sur différents jeux de données réels, issus de la littérature. On y étudie en particulier l'évolution des habitudes le long de l'expérimentation, ainsi que les capacités de passage à l'échelle de sxED. Une comparaison qualitative entre xED et sxED est également réalisée. Elle montre que sxED trouve des épisodes très similaires à ceux trouvés par xED, avec un temps d'exécution significativement plus rapide.

## Monitoring des $k$ motifs les plus réguliers

**Résumé** Le chapitre 5 étudie un autre type de description de périodicité : la *régularité*. Ce modèle de périodicité est moins robuste vis-à-vis de la variabilité temporelle des habitudes, mais demeure intéressant par sa simplicité et ses capacités de passage à l'échelle. Dans ce chapitre, nous proposons TKRES (Amphawan et al., 2015), un algorithme permettant la découverte et la mise à jour des épisodes les plus réguliers dans un flot de données domotiques. L'approche *top-k* permet de contrôler la taille de l'espace des résultats, ce qui facilite l'analyse des résultats par l'utilisateur final. TKRES est également évalué sur des jeux de données réels validés dans la littérature.

**Formalismes** La description de la périodicité via une distribution de probabilité sur une période d'intérêt est un outil puissant et extressif, mais dont la maintenance est gourmande en ressources. Nous étudions donc une mesure alternative de périodicité : la *régularité*, qui étudie les intervalles entre les occurrences successives d'un comportement. Initialement définie pour la recherche d'itemsets dans des données transactionnelles (Tanbeer et al., 2008), la régularité est ici adaptée aux flots d'événements. Nous définissons la régularité d'un épisode $E$ comme l'intervalle de temps maximal entre le début d'une occurrence minimale de $E$ et la fin de l'occurrence minimale disjointe suivante : la régularité est donc monotone.

Nous proposons TKRES pour la découverte et la mise à jour des $k$ épisodes les plus réguliers dans un flot d'événements parcouru par une fenêtre glissante composée de $m$ batchs. L'approche top-$k$ a deux avantages majeurs : l'utilisateur final défini la taille des résultats produits en fonction de ses capacités de traitement, et il n'a pas besoin de fixer de seuil de fréquence ou de régularité, une tâche réputée difficile.

**Structures de données** La recherche des $k$ épisodes les plus réguliers est réalisée à partir de deux structures dédiées :

- Un arbre $k$-tree (figure d) des préfixes contenant les time queues des épisodes. La navigation au sein de l'arbre est accélérée par l'utilisation de pointeurs permettant également un parcours transversal de l'arbre ;
- Une liste chainée ($k$-list) à $k$ éléments : les épisodes recherchés.

Puisqu'on ne cherche que $k$ épisodes, il n'est pas nécessaire de construire l'intégralité du $k$-tree : il suffit en général qu'il soit suffisamment profond pour contenir $k$ nœuds : la régularité étant monotone, les nœuds les moins profonds sont plus réguliers que leurs descendance. Certains nœuds peuvent néanmoins manquer, ils sont alors construits au besoin, mais ne sont pas maintenus. Cette stratégie permet de trouver un compromis entre espace de stockage nécessaire et vitesse de mise à jour.

**Évaluation** TKRES a été évalué sur des jeux de données issus de la littérature du monitoring d'activité. Nous y montrons la capacité de TKRES à maintenir à jour la $k$-list en temps (quasi-)réel quand les paramètres varient.

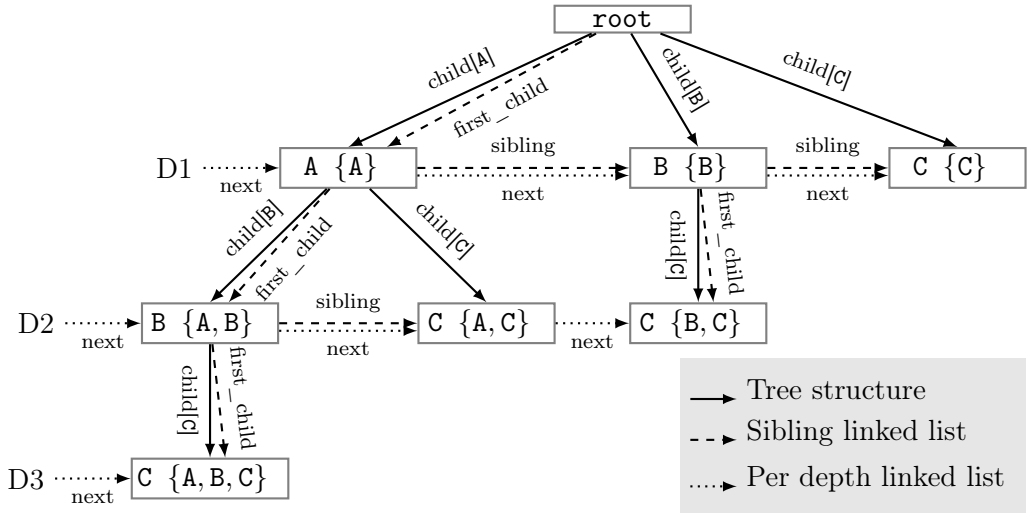



FIGURE d – Exemple de $k$-tree, si l'alphabet des labels d'événements est $\{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$. On met ici l'accent sur les mécanismes de navigation au sein de l'arbre

L'analyse qualitative des épisodes extraits permet de mettre à jour certaines habitudes quotidiennes (rythmes de sommeil, repas), et met en avant certaines zones géographiques du logement.

# Conclusions et perspectives

**Résumé** Le chapitre 6 conclue cette thèse. Il résume les contributions principales et met en avant des pistes à explorer par la suite.

**Contributions** Le travail présenté dans cette thèse se situe dans le domaine du monitoring de l'activité. Il se concentre en particulier sur la découverte et la description des habitudes des habitants d'environnements intelligents. Les contributions principales sont :

- Nous proposons un nouveau formalisme de description de la périodicité de motifs récurrents, basé sur la description de la distribution temporelle des événements au sein de la période considérée. Nous proposons

également une mesure, la précision, qui évalue l'adéquation entre le modèle de périodicité proposé et les observations des capteurs.

- Nous proposons deux algorithmes pour la découverte de motifs périodiques suivant ce formalisme : xED (Soulas et al., 2013, 2015) et sxED (Soulas and Lenca, 2015). xED est conçu pour l'analyse de bases de données statiques, sxED permet la gestion de données évolutives.
- Nous étendons le concept de régularité aux séquences d'événements, et proposons un algorithme, TKRES (Amphawan et al., 2015) pour la découverte des $k$ épisodes les plus réguliers dans des flots d'événements.
- Nous évaluons les algorithmes qualitativement (pertinence des motifs trouvés pour le monitoring de l'activité) et quantitativement (utilisation des ressources et passage à l'échelle des solutions proposées) sur des jeux de données réels.

Les contributions de ce travail sont principalement algorithmiques. Néanmoins, nous nous sommes aussi intéressés à l'inclusion du personnel médical et des aidants dans le processus de monitoring. Ces contributions sont décrites dans les annexes de la thèse. L'annexe A présente une approche pour la fouille de processus qui place l'expertise du superviseur au centre de la recherche, ce qui une fouille adaptée à ses attentes et ses requêtes, et utilisant ses connaissances. L'annexe B propose des outils de visualisation de données.

### Pistes pour des travaux futurs

- Caractérisation de la périodicité : nous évaluons la périodicité sur sa précision seulement. D'autres mesures de qualité pourraient certainement enrichir l'approche.
- Gestion de l'incertitude : les données provenant de capteurs, elles sont par nature incertaines, ce qui a des conséquences sur les résultats.
- Analyse de tendance : plus que les habitudes elles-mêmes, c'est souvent leur évolution dans le temps qui nous renseignent sur l'état de santé.
- Reconnaissance d'activités : les activités sont ici assimilées à des épisodes, ce qui ne prend pas en compte la variabilité possible dans la réalisation des activités.

## Résumé

Le vieillissement de la population mondiale au cours des décennies à venir pose de nouveaux défis socitaux. L'un de ces défis est de permettre aux personnes âgées, souvent isolées et fragilisées de vivre dans leur domicile personnel le plus longtemps possible, et dans les meilleures conditions. L'émergence de nouvelles technologies domotiques et des réseaux de capteurs offrent de nouvelles opportunités pour faciliter les interactions des personnes âgées avec leur environnement et assurer un suivi médical au jour le jour. En particulier, les capteurs domotiques collectent des informations sur l'activité dans le logement, et permettent d'évaluer autonomie état de santé.

Dans cette thèse, nous nous concentrons sur l'exploitation des données enregistrées par de tels réseaux de capteurs. Nous nous intéressons en particulier à la découverte des habitudes, puisqu'elles jouent un rôle important dans le maintien de l'autonomie chez les personnes âgées. Nous proposons donc ainsi plusieurs méthodes pour la caractérisation et la découverte non supervisée des habitudes. Nous nous attachons en particulier à la prise en compte et la description de la variabilité inhérente à la vie humaine. Nous proposons également des stratégies pour la mise à jour des connaissances en temps réel. Les différentes contributions sont évaluées qualitativement et quantitativement sur des jeux de données réelles, issues de la littérature du domaine.

Ces contributions ouvrent la voie pour la mise en place de nouvelles méthodes pour le monitoring d'individus isolés via l'adaptation personnalisée du logement, la détection d'anomalies, ou l'analyse de l'évolution des habitudes et de l'état de santé.



**Mots-clés :** Exploration de données, Apprentissage automatique, Réseaux de capteurs, Assistance aux personnes âgées, Monitorage médical, Habitude, Systèmes experts, Classification automatique



## Abstract

The aging of the population in the coming decades raises new challenges in order to help elderly people live longer at home, independently and safely. The emergence of assistive technologies, and in particular home automation devices and sensor networks open up new opportunities to ease the interactions between the elderly and their environment, and to monitor their health status remotely. In particular, the home automation sensors record information on the activity in the home and make it possible to assess autonomy and well-being.

In this thesis, we focus on the mining of the data recorded by such sensor networks. We focus more particularly on the discovery of habits. Indeed, the daily routines help maintain autonomy. We thus propose unsupervised data mining algorithms for the discovery and description of periodic behaviors. A particular attention is paid to the inclusion and characterization of the variability inherent in human life. We also propose strategies for the real-time knowledge discovery and update. The interactions with the human supervisor (physician, caregiver, family member) have also been studied, which lead to the design of a supervision model, combining sequence mining with expert knowledge allows a custom monitoring, tailored to the exact expectations of the supervisor. All of these contributions are evaluated on real-life datasets from the literature.

These contributions opens the way towards innovative methods for the monitoring of isolated individuals, in particular thanks to the personalized adaptation of the home, anomaly detection, or the analysis of the evolution of habits and heath status.



**Keywords** : Data mining, Machine learning, Sensor networks, Old age assistance, Patient monitoring, Habit, Pattern perception, Expert systems, Automatic classification



TELECOM Bretagne

**n° d'ordre : 2016telb0377**

**Télécom Bretagne**

**Technopôle Brest-Iroise - CS 83818 - 29238 Brest Cedex 3**

**Tél : + 33(0) 29 00 11 11 - Fax : + 33(0) 29 00 10 00**