

Internet of Things

Michele Ianni · Antonella Guzzo ·
Raffaele Gravina ·
Hassan Ghasemzadeh ·
Zhelong Wang *Editors*

Activity Recognition and Prediction for Smart IoT Environments

 Springer

Internet of Things

Technology, Communications and Computing

Series Editors

Giancarlo Fortino, Rende (CS), Italy

Antonio Liotta, School of Computing, Edinburgh Napier University, Edinburgh, UK

The series Internet of Things - Technologies, Communications and Computing publishes new developments and advances in the various areas of the different facets of the Internet of Things. The intent is to cover technology (smart devices, wireless sensors, systems), communications (networks and protocols) and computing (theory, middleware and applications) of the Internet of Things, as embedded in the fields of engineering, computer science, life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in the Internet of Things research and development area, spanning the areas of wireless sensor networks, autonomic networking, network protocol, agent-based computing, artificial intelligence, self organizing systems, multi-sensor data fusion, smart objects, and hybrid intelligent systems.

Indexing: *Internet of Things* is covered by Scopus and Ei-Compendex **

Michele Ianni • Antonella Guzzo •
Raffaele Gravina • Hassan Ghasemzadeh •
Zhelong Wang
Editors

Activity Recognition and Prediction for Smart IoT Environments


 Springer

Editors

Michele Ianni 
University of Calabria
Rende, Italy

Antonella Guzzo
University of Calabria
Rende, Italy

Raffaele Gravina
University of Calabria
Rende, Italy

Hassan Ghasemzadeh 
Arizona State University
Phoenix, AZ, USA

Zhelong Wang
Dalian University of Technology
Dalian, Liaoning, China

ISSN 2199-1073

ISSN 2199-1081 (electronic)

Internet of Things

ISBN 978-3-031-60026-5

ISBN 978-3-031-60027-2 (eBook)

<https://doi.org/10.1007/978-3-031-60027-2>

© The Editor(s) (if applicable) and The Author(s), under exclusive license to Springer Nature Switzerland AG 2024

This work is subject to copyright. All rights are solely and exclusively licensed by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

If disposing of this product, please recycle the paper.

Preface

In recent years, the rapid evolution of technologies such as the Internet of Things (IoT), artificial intelligence (AI), and machine learning has significantly transformed various aspects of daily life, from healthcare and industrial processes to home automation and personal wellness. The convergence of these technologies has given rise to innovative solutions for activity recognition and prediction, which are pivotal in creating smarter, more responsive environments. This book, “Activity Recognition and Prediction for Smart IoT Environments,” provides a comprehensive exploration of the latest advancements in this dynamic field.

Chapter 1 introduces the concept of pervasive computing and its application in smart spaces, specifically smart homes and offices. It details a methodology that leverages process mining techniques to automatically segment and model human habits based on sensor data, offering insights into how these models can be used to anticipate and enhance user actions in smart environments.

Chapter 2 focuses on the role of wearable sensors in human activity recognition. It examines how these sensors, due to their compact size and computational power, have become essential tools for real-time activity and emotion detection. The chapter reviews various methods and applications of wearable sensors in domains such as healthcare and sports performance analysis.

Chapter 3 addresses the challenges faced by wheelchair users due to a sedentary lifestyle. It presents a postural monitoring system designed to recognize and analyze various postures to prevent health complications. This chapter emphasizes the importance of selecting appropriate sensors and machine learning techniques to optimize postural recognition.

Chapter 4 compares video-based and sensor-based Human Activity Recognition (HAR) systems. It highlights the advantages of sensor-based methods in terms of privacy and cost, and discusses the impact of deep learning on feature representation, balancing the benefits with computational overhead.

Chapter 5 explores the recognition of multi-user activities using wireless signals. It introduces a novel two-level data fusion method applied to Wi-Fi Channel State Information (CSI) signals for identifying individual and group actions. This chapter

also addresses the security implications of collecting and processing sensitive user data.

Chapter 6 presents a probabilistic cascading binary classifier designed for resource-efficient activity classification in wearable devices. It discusses how this approach manages sensor sampling frequency and feature extraction to balance energy consumption with classification accuracy.

Chapter 7 delves into the application of machine learning in Industry 4.0, focusing on Human Activity Recognition (HAR) within smart factories. The chapter reviews the use of advanced AI and machine learning methodologies to enhance automation and efficiency in industrial settings, providing insights for practitioners and researchers alike.

Chapter 8 examines the challenges and advancements in real-time HAR using smart devices for elder care and medical rehabilitation. It highlights the complexities of dealing with heterogeneous data sources and sensor limitations, and discusses the growing prominence of deep learning in overcoming these challenges.

This book is intended for researchers, academics, professionals, and students who are interested in the intersection of activity recognition, prediction, and IoT technologies. It brings together a diverse range of topics and methodologies, offering a valuable resource for understanding and advancing the state of the art in smart environments.

We hope that the insights and innovations presented in this volume will inspire further research and contribute to the development of smarter, more intuitive systems that enhance our interactions with technology and improve our quality of life.

Rende, Italy
Rende, Italy
Rende, Italy
Phoenix, AZ, USA
Dalian, Liaoning, China

Michele Ianni
Antonella Guzzo
Raffaele Gravina
Hassan Ghasemzadeh
Zhelong Wang

Contents

Discovering Human Habits Through Process Mining: State of the Art and Research Challenges	1
Francesco Leotta, Massimo Mecella, and Silvestro Veneruso	
Methodology for Human Activity Recognition Based on Wearable Sensor Networks	19
Zhelong Wang	
A Sitting Posture Monitoring System in Wheelchair Users	47
Patrick Vermander, Nerea Perez, Aitziber Mancisidor, Itziar Cabanes, and Jon Torres-Unda	
A Comprehensive Review of Deep Learning for Activity Recognition	67
Dipanwita Thakur and Giancarlo Fortino	
Multi-User Activity Monitoring Based on Contactless Sensing	97
Qimeng Li, Bharat Lal, and Raffaele Gravina	
Efficient Sensing and Classification for Extended Battery Life	111
Mahdi Pedram, Ramesh Kumar Sah, and Hassan Ghasemzadeh	
Unveiling the Potential of Machine Learning in Activity Recognition for Industry 4.0	141
Diletta Chiaro, Pian Qi, Mariapia De Rosa, Salvatore Cuomo, and Francesco Piccialli	
Human Activity Recognition: Trends and Challenges	161
Dipanwita Thakur and Arindam Pal	
Index	183

Discovering Human Habits Through Process Mining: State of the Art and Research Challenges



Francesco Leotta, Massimo Mecella, and Silvestro Veneruso

1 Introduction

Recent years have shown a growing interest in the market for embedding sensors and actuators in physical environments to facilitate the execution of numerous tasks. The idea is to use sensors to collect real-time information about the environment and to use it to trigger actions through actuators in order to automate physical tasks, aiding humans in their daily lives. The term *pervasive* (or ubiquitous) *computing* is usually employed to indicate the set of techniques apt to this aim [27].

Smart spaces represent, in particular, an emerging class among pervasive computing application areas. Smart homes and offices are, in particular, representative examples where pervasive computing could take advantage of *Ambient Intelligence* (AmI) [24]. AmI lies at the crossroad between many different areas, including Artificial Intelligence (AI) and Human–Computer Interaction (HCI).

AmI techniques are based on models specifically trained for a specific home and/or inhabitant. In particular, models represent [21]:

- *Actions*, i.e., atomic interactions with the environment (e.g., using a household)
- *Preferences*, i.e., a set of rules explicitly representing decisions to make in specific contextual conditions
- *Activities*, i.e., groups of actions (one in the extreme case) or sensor measurements/events with a final goal (e.g., cleaning the house). Activities can be collaborative, including actions by multiple users, and can interleave with each other.

F. Leotta (✉) · M. Mecella · S. Veneruso
Dipartimento di Ingegneria Informatica Automatica e Gestionale “A. Ruberti”, Sapienza
Università di Roma, Rome, Italy
e-mail: Leotta@diag.uniroma1.it; Mecella@diag.uniroma1.it; mecella@dis.uniroma1.it;
Veneruso@diag.uniroma1.it

- *Habits*, i.e., groups of activities that happen in specific contextual conditions. Habits represent human routines (e.g., what the user does every morning between 08:00 and 10:00).

It has been argued that business process formalisms can be used as models for activities and habits [19]. A *business process* is a set of interrelated tasks performed in a company in order to conduct specific functions (e.g., ordering goods). In order to acquire such models, process mining (PM) techniques can be employed. PM [25] is a fairly recent research discipline that combines Data Mining techniques with techniques used in Business Process Management (BPM) [7]. Its main objective is to extract meaningful information from event logs. When PM techniques such as *process discovery* are applied to data gathered from smart spaces, it is possible to model and visualize human activities and habits as business processes.

In this chapter, we first introduce a state-of-the-art methodology [8] allowing, given a sensor log, to automatically segment *human habits* by applying process mining techniques. Such methodology relies on a bottom-up discretization strategy for the timestamp attribute. Such discretization algorithms find the best division of a continuous attribute by iteratively merging contiguous subranges (also called “bins”) following a quality evaluation heuristic. In this case, the heuristic is based on quality measures computed on the process models automatically mined, through *process discovery*, from the intermediate bins. In particular, we drive the discretization targeting process models with high *simplicity* and low *structuredness*. Each obtained bin then represents a time range in which the human is supposed to perform activities following a clearly identifiable human process.

Habits mined via this approach are then further analyzed with several process discovery algorithms to obtain related process models. Such models of habits can be used, in conjunction with condition mining techniques, to anticipate user actions and enact them whenever possible.

2 Background and Related Works

2.1 Sensor Data in Smart Spaces

Sensors available in a smart space produce a *sensor log*.

Definition 1 (Sensor Log) Given a set $S = \{s_1, \dots, s_n\}$ of available sensors, a sensor log is an ordered sequence of measurements coming from S , each represented by a tuple $\langle ts, s, v \rangle$, where $s \in S$ identifies the source sensor producing a measured value v at the timestamp ts . The measured value v can be either categorical or numeric.

Measurements can be produced by a sensor on a periodic basis (e.g., temperature measurements in a room) or whenever a particular event happens (e.g., door

openings). As many of the algorithms proposed in the literature borrow the terminology of data mining, the sensor log could be translated as a sequence of events instead of a sequence of raw measurements. For instance, the set of raw sensor measurements representing the activation of motion sensors upon entering the bathroom could be translated as “the inhabitant entered the bathroom.”

Definition 2 (Event Log) Given a set $E = \{e_1, \dots, e_n\}$ of event types, an event log is a sequence of pairs $\langle e, t \rangle$, where $e \in E$ and t is a timestamp.

Translating a sensor log into an event log is not a trivial task, and it could cause a loss of information, especially if a discretization of periodic sensor measurements is required [19].

An example of such translations is provided in [20]. The approach is applicable to logs only consisting of Presence InfraRed (PIR) sensors, i.e., motion sensors triggering whenever a human intercepts their detection zone. The triggering sequence of PIR sensors describes the movements of a human in the environment. Such movements are extracted in [20] by applying the seminal trajectory clustering algorithm called TRACCLUS [18]. TRACCLUS consists of two phases: a trajectory partitioning technique based on the minimum description length (MDL) principle, followed by a density-based line segment clustering algorithm. The authors in [20] exploit the partitions found by TRACCLUS to segment the log into subtrajectories with a homogeneous velocity. Each of these trajectories is then classified into one of these three categories with labels MOVEMENT, AREA, and STAY by considering information features such as duration, velocity, and heterogeneity.

The classification allows to replace sequences of raw sensor measurements with human actions consisting of a category and a location, the latter inferred by the position of the corresponding sensors in the house. For example, the $\langle \text{AREA Bathroom} \rangle$ action indicates that the human inhabitant was wandering around the bathroom, whereas $\langle \text{STAY Kitchen_table} \rangle$ represents the fact that the inhabitant remained for a while sitting at the kitchen table. At this point, the sensor log has been converted into an event log or, more specifically, an action log.

The authors in [14] propose a different clustering technique in order to turn sensor measurements into actions.

2.2 Process Mining

Process mining [25] is a fairly recent research discipline that combines Data Mining techniques with methodologies used in BPM [7], such as process modeling and process analysis. Its main objective is to extract meaningful information from event logs. Three types of process mining techniques are usually identified:

1. *Process discovery*, i.e., a technique for discovering the process model describing the behavior shown in the event log. Thus, it takes as input an event log and automatically generates the correspondent process model.

2. *Conformance checking*, i.e., a technique for comparing an existing process model with an event log of the same process. It can be used to check if reality, as recorded in the log, conforms to the model and vice versa.
3. *Enhancement*, i.e., a technique used to extend or improve an existing process model using information about the actual process recorded in some event log

In Sect. 3, we show an approach that employs *process discovery*, which produces as output a process model represented using the *Petri net* formalism, i.e., a directed graph composed of nodes and arcs (respectively, called *places* and *transitions*). Such models find the right balance between the following properties:

- *Soundness*, i.e., the fact that the resulting Petri net will always terminate and does not contain deadlocks
- *Fitness*, i.e., how much the input log can be replayed on the resulting Petri net.
- *Precision*, i.e., the resulting Petri net should not allow for a behavior that is not related to the one in the event log.
- *Rediscoverability*, i.e., the algorithm discovers a model language-equivalent to the system shown in the event log.

The structural behavior of a Petri net mined through process discovery can be analyzed using several different quality measures. In the approach shown in Sect. 3, the authors are mainly interested in *structuredness* and *simplicity* metrics since they want to find human process models that are easy to read and understand.

Structuredness [16] is a measure obtained by disassembling the observed model into small submodels, assigning a score to each of them, and combining these scores. In particular, the score will be lower for patterns perceived as simple (e.g., sequences, while, and choice patterns) and higher for the complex ones.

Simplicity is a metric depending only on the size and structure of the model without considering its behavior. Given a Petri net, we can define as $|F|$ the number of arcs, $|P|$ the number of places, and $|T|$ the number of transitions inside the model. Therefore, we can define this property as $\frac{|P|+|T|}{|F|}$.

A higher value of *simplicity* means that the Petri net is simpler to understand. Conversely, if the equation returns a low value, we expect that the Petri net has a complex structure. Lower values of simplicity may happen instead when the number of arcs is much bigger than the number of nodes in the net, so, in general, this leads to Petri nets that are not easy to read.

2.3 Unsupervised Approaches to Ambient Intelligence

The vast majority of approaches in ambient intelligence are inherently manual, thus requiring a segmented and labeled dataset at least at training time, whereas windowing techniques can be used to roughly segment the sensor stream at runtime [15]. Anyway, few works exist that can be defined as fully automatic.

The authors in [1] propose the APUBS algorithm to automatically extract Event–Condition–Action (ECA) rules by considering the typology of the sensors involved in the measurements and the time relations between their activations. An ECA rule has the form “ON event IF condition THEN action,” thus automating the execution of an action as soon as a specific event is detected and if certain contextual conditions are met. APUBS is based on a clear distinction between (i) type O sensors installed in objects, thus providing direct information about the actions of the users (e.g., a sensor capturing the opening of the fridge door), (ii) type C sensors providing information about the environment, and (iii) type M sensors providing information about the position of the user inside the house.

Events in the event part of the ECA rule always come from type O and type M sensors. Conditions are usually expressed in terms of the values provided by type C sensors. Finally, the action part contains only type O sensors, which can also serve as actuators. The set of Type O sensors is called *mainSeT*. As a first step, the APUBS method discovers, for each sensor in the *mainSeT*, the set *associatedSeT* of type O and type M sensors that can be potentially related to it as triggering events, by using the seminal APriori method. As a second step, the technique discovers the temporal relationships between the events in *associatedSeT* and those in *mainSeT*. During this step, nonsignificant relationships are pruned. As a final step, the conditions for the ECA rules are mined with a JRip classifier. The authors in [5] instead extract ECA rules by using a variation of the seminal APriori algorithm.

The approach described in Sect. 3 differs from ECA rule extraction as it does not directly discover enactment rules. Instead, it discovers the process models behind each human habit. This process can then be employed to extract enactment rules, which are strongly related to the choice made by the human inhabitant, while he/she is following his/her habit.

The authors in [3] proposed instead an approach based on the minimum description length (MDL) principle to automatically extract activity patterns. The algorithm takes as input a dataset consisting of a sequence of sensor events witnessing human interactions with the environment. At each step, the algorithm looks for the patterns that best compress the dataset. A pattern consists of a sequence of sensor events and their occurrences in the dataset. Starting from a single pattern for each different sensor event, the algorithm at each step tries to extend patterns, aiming for the best possible compression. In particular, every instance of the pattern is replaced by a symbol associated with the pattern. The compression of a dataset D given a pattern P is given by the formula $\frac{DL(D)}{DL(D|P)+DL(P)}$, where $DL(D)$ represents the description length of the dataset with the current patterns (measured, e.g., in bits), $DL(D|P)$ represents the description length of D if all of the occurrences of P are replaced with a symbol, and $DL(P)$ represents the description length of the pattern, which must be considered in order to take into account the pattern length in compression evaluation. The algorithm stops as soon as no further compression is possible, returning all the patterns found. In the original algorithm, a clustering step is applied in order to recognize variants of human routines.

The extraction of meaningful information from activities presents a daunting task that has been addressed through various methodologies and within different

application contexts, encompassing daily living [11, 12], as well as security-related domains [9, 10, 13], to name just a few examples.

Differently from these last approaches, in Sect. 3, we focus on habits instead of activities. Additionally, in [3] patterns are extracted with the sole goal of recognizing them at run-time without providing a structured description of human routines.

3 Unsupervised Human Habit Discovery

The main idea behind the application of PM in smart spaces is to use data mining techniques that target classical business processes, also defined as cyber-physical processes [17]. Applying PM to this scenario introduces several challenges [2]:

- Smart spaces typically produce sensor logs, while process mining techniques require event logs. Events in event logs are executions of tasks, while sensor logs contain fine-grained sensor measurements.
- Process mining requires the log to be segmented into traces, where each trace represents an “execution” of a specific process (an activity or a habit).
- The choice of the right modeling formalism to employ is not trivial [6].

The goal of this section is to show how process mining, and in particular process discovery, can be applied to smart homes for log segmentation purposes. In the literature, a large body of research can be found on log segmentation. Such methodologies can be grouped into two families: manual and automatic. Manual techniques often require manual labeling of training instances, usually involving final users in annoying and imprecise training sessions. Furthermore, they require a segmented and labeled dataset at least at training time, whereas windowing techniques can be used to roughly segment the sensor stream at run-time [15]. On the other hand, automatic methods that aim at partially or completely eliminating human effort in labeling show limitations related to the absence of domain knowledge.

As pointed out above, a major requirement for applying process mining to smart spaces is to convert the sensor log into an event log. To achieve this aim, we can employ the technique proposed in [20] and already described in Sect. 2.1.

The obtained event log \mathcal{A} is a timestamped sequence of tuples $\langle d, s, e, a \rangle$, where

- d is the day in the log.
- s and e are, respectively, the timestamps at which the action starts and ends. Tuples follow a chronological order according to s .
- a is the result of the sensor aggregation. It is labeled as STOP, AREA, or MOVEMENT, and it is followed by the identifier of the position.

The event log is further preprocessed: Consecutive repetitions of the very same action are merged together. In particular, given two consecutive tuples (i.e., consecutive events in the log) $A = \langle d_A, s_A, e_A, a_A \rangle$ and $B = \langle d_B, s_B, e_B, a_B \rangle$,

if $d_A = d_B$, $a_A = a_B$, and $e_A = s_B$, then A and B are merged into a single event defined as $\langle d_A, s_A, e_B, a_A \rangle$. This operation can be applied iteratively to a chain of multiple subsequent tuples, as shown in Table 1.

Once this preprocessing of the event log is completed, we can then proceed to the segmentation phase. For this purpose, we apply a classical bottom-up discretization technique based on the attribute referring to the time of day. Just to capture the underlying principle behind our approach, we can refer to the well-known *Chimerge* [23] strategy: This algorithm starts by dividing the entire range of an attribute at the finest level of granularity possible. Then, adjacent bins/intervals that met a condition based on χ^2 (i.e., the statistical measure *Chi*) are iteratively merged. If this condition is not met, the algorithm stops, and those bins/intervals remain separated.

We start our segmentation by dividing the entire range of the *time-of-the-day* attribute (i.e., 00:00–24:00) into bins of constant width. For instance, if 15 minutes is chosen as the minimum bin width, the *time-of-the-day* attribute will be divided into $24 * (60/15) = 96$ bins. Each bin is associated with a correspondent sublog (of the unsegmented event log) where we have a case for each day in the dataset only containing actions with `start_timestamp` included in the bin (e.g., all the actions in a specific day happening since 00:00–00:15).

Definition 3 Given an action log \mathcal{A} , we define $eventLog(\mathcal{A}, [a, b])$ as the event log having the day as a case identifier and containing, for each case, the actions performed between time a and b during the day associated with the case.

Once this initial segmentation of the log is provided, Algorithm 1 is executed. The algorithm takes as input (i) a finite set `intervals` of chronologically ordered intervals/bins (96 in the previous example), (ii) a parameter `minN` denoting the minimum number of intervals to be returned by the algorithm, and (iii) a parameter `minScore` representing the stop criterion.

The algorithm finds the best possible segmentation of the event log in habits, producing no less than `minN` intervals/bins (see row 1). At each iteration (see rows 2–16), the algorithm iterates on all the intervals/bins, and for each pair of adjacent intervals/bins (see row 5), it applies the *inductive miner* (see row 6) to the event log obtained by merging those intervals/bins. In order to also consider possible habit intervals that cover two consecutive days (e.g., a time interval that lasts from 23:00 to 01:00 of the following day), the merging of two adjacent intervals/bins is circular, so that the last interval/bin in the array is merged with the first.

For each couple of adjacent intervals/bins, the inductive miner produces a Petri net pn . For each of these Petri nets, we compute a score obtained starting from the simplicity and structuredness measures introduced in Sect. 2. This equation allows us to find the pair of adjacent intervals/bins whose correspondent Petri net is the most readable one. In fact, in general, the Petri nets that are more simple and easy to understand have a high value for simplicity and a low value for structuredness. Thus, in each iteration, the couple of adjacent intervals/bins that may actually be merged are the ones that maximize the value obtained from this equation. Simplicity in particular is multiplied by a factor of 100, which has been empirically chosen in

Table 1 The portion of the event log shown in Table (b) represents the output of the merging of events from Table (a). Three consecutive STAY events are merged into a single one

(a)			(b)		
Day	Start ts	End ts	Start ts	End ts	Action
...					
0	07:55:00	08:00:00			AREA Bathroom
0	08:00:00	08:01:49	07:55:00	08:00:00	AREA Bathroom
0	08:01:49	08:07:09	08:00:00	08:07:22	STAY Kitchen_table
0	08:07:09	08:07:22	08:07:22	08:09:17	STAY Kitchen_table
0	08:07:22	08:09:17	08:07:22	08:09:17	STAY Bedroom
...					


```

Data: A, intervals, minN, minScore
1 while  $len(intervals) > minN$  do
2   max = 0;
3   index = null;
4   for  $i$  in  $[0, len(intervals) - 2]$  do
5     pair = concat(intervals[i], intervals[i+1]);
6     pn = inductiveMiner(eventLog(A, pair));
7     score =  $100 \times pn.simplicity - pn.struct$ ;
8     if  $score > maxScore$  then
9       max = score;
10      index = i;
11    end
12  end
13  if  $maxScore < minScore$  then
14    return intervals;
15  end
16  intervals = merge(intervals, index, index+1);
17 end
18 return intervals;

```

Algorithm 1: The segmentation algorithm

order to uniformize the two quality measures since simplicity values are always less than 1. We keep track of the maximum score computed and of the corresponding couple of adjacent bins (see rows 8–11).

Once all adjacent bins have been considered, if the maximum score computed is above the `minScore` threshold, the `intervals` array is updated by merging the adjacent bins corresponding to the maximum score. Otherwise, the algorithm ends, as any additional merging would not be convenient.

Noticeably, the algorithm always terminates. The merging phase stops either if it is not convenient to keep merging bins or if a minimum number of bins are reached (note that iteration by iteration the number of bins is always decreased by one). After the algorithm terminates, the `intervals` variable contains a set of intervals/bins each corresponding to a habit. Here the rationale is that bins will be merged only if the resulting Petri net results are simpler and less structured, meaning that the process model of the underlying habit is easy to read.

4 Experiments

The approach described in Sect. 3 is validated against the *aruba* dataset from the CASAS project (<http://casas.wsu.edu/datasets/>). The reason for this choice relies on the major impact this project has had on the community and the availability as source code of algorithms that are often used as benchmarks [4, 14, 19, 22, 26].

The *aruba* dataset is a partially labeled sensor log that contains data collected in a real-life scenario for 220 days. It contains 6477 labeled activities, with the

Table 2 Most frequent temporal intervals for each activity

Activity	Most frequent intervals
Bed to toilet	00:30–00:45 03:30–06:15
Eating	08:00–08:15 09:15–10:45 12:00–12:15 13:45–14:15 18:00–19:00 19:15–19:45
Enter home	11:15–11:30 13:15–14:15 14:30–15:15 15:45–16:00
Housekeeping	10:30–11:45 14:30–15:15
Leave home	11:30–11:45 14:45–15:00
Meal preparation	06:00–06:30 06:45–10:45 12:00–12:30 13:15–14:45 15:15–20:15
Relax	08:45–09:30 13:30–00:00
Sleeping	23:30–07:30
Wash dishes	09:45–10:30 11:00–11:15 17:45–18:00 19:30–20:00 20:45–21:00
Work	12:00–12:30 14:15–14:30 15:45–16:45

start and end markers of activities performed by the resident, meaning that for a subset of measurements, we know the activity correspondent to those activations of sensors. The activities available in the dataset, with the correspondent number of their occurrences, are the following ones: *meal preparation* (1606), *relax* (2910), *eating* (257), *work* (171), *sleeping* (401), *wash dishes* (65), *bed to toilet* (157), *enter home* (431), *leave home* (431), *housekeeping* (33), and *resperate* (6). The inhabitant interacts, among others, with 31 *Presence InfraRed (PIR)* sensors that trigger as soon as a person enters their detection area, producing discrete measures that can be easily associated with actions.

Once obtained the action log \mathcal{A} from the raw log, Algorithm 1 is executed, passing as arguments (i) the preprocessed action log, (ii) an array of 96 initial bins obtained by segmenting the *time-of-the-day* attribute in equal-width bins of 15 minutes, (iii) a minimum number of bins equal to 2, and (iv) a minimum score chosen empirically.

The six best intervals from our discretization algorithm are the following ones: *05:15–07:00*, *07:00–13:45*, *13:45–18:15*, *18:15–21:45*, *21:45–23:00*, and *23:00–05:15*. Each of these temporal intervals is considered a human *habit*.

In order to evaluate the quality of the result obtained, a simple statistical analysis of the dataset has been performed. In particular, each habit was mapped to a set of activities to show its plausibility. As already stated, these activities have been manually labeled in the original sensor log \mathcal{S} . According to the activity label available in the raw log, we computed the temporal intervals in which each activity occurs more frequently (see Table 2).

Table 3 introduces another kind of association obtained by comparing the most frequent intervals for each activity with respect to the habits found by our discretization algorithm. This association shows in which habit a certain activity occurs more frequently. For instance, the activity *sleeping* occurs more frequently during the time interval “23:30–07:30” and so can be associated with both the habit intervals “23:00–05:15” and “05:15–07:00.”