

# Discovering Routine Events and their Periods in Sensor Time Series Data

Jia Fang, Athanasios Bamis and Andreas Savvides  
ENALAB, Electrical Engineering Department, Yale University  
51 Prospect St. Room 000, New Haven, CT 06520, USA  
{firstname.lastname}@yale.edu

**Abstract**—This paper describes an algorithm for determining if an event occurs on a routine basis within a fixed time interval and also determines the period of that interval. The algorithm is geared towards the observation of routine actions taken by humans that are consistently recurring but not exactly periodic. The goal of the algorithm is to determine the minimum period of the interval in which the event occurs that also encloses the entire duration of the event. The proposed algorithm, that focuses on a single event type, is tested with artificially generated data and data from a sensor network testbed that verify its correct operation.

## I. INTRODUCTION

A number of sensor network systems have been deployed for monitoring a person’s whereabouts at varying levels of granularity ranging from home level monitoring to world-wide GPS monitoring. Such systems have demonstrated a lot of success towards the collection of high quality data, creating a set of new challenges for data processing especially in the area of understanding the structure of behaviors. This is a promising area of research that could lead to new ways of analyzing behavior, precisely quantifying changes and observing habits to a level that could closely monitor certain medical conditions while also providing assistive services.

In this paper, we present a method geared towards studying the habits of a person to the extent they can be captured in time and location information collected by a sensor network. We study the notion of “spatio-temporal routines” and propose a new algorithm for determining whether an event occurs consistently within a fixed time interval, without itself being periodic. Drawing on the experiences and lessons learned from our sensor network testbeds in elder monitoring [1], in this work we place our emphasis in the treatment of a single event. With this, the key property of human routines we wish to capture is that although an event, such as a kitchen visit for two hours, may occur on a regular basis, it will most likely not occur exactly, or even approximately, periodically over any significant portion of the interval of observation. Similarly, while a person may visit the garage for an hour every afternoon, he may do so anytime between 1pm and 4pm, and therefore, the actual visit itself will (most likely) not occur periodically. What does occur periodically however, and this will be the point we will exploit in our ultimate definition of spatio-temporal routines, is the time interval in which the garage visit takes place, i.e. the time

period 1pm to 5pm. The proposed algorithm also considers gaps in spatio-temporal routines. These account for the cases where a person may visit the garage for an hour every weekday but not on weekends, and also the case where a person may visit the kitchen for two hours in the evening everyday except on those rare days when he has to work late.

Discovering spatio-temporal routines poses four main difficulties that we try to address in our algorithm design. First, the events making up a routine are not themselves periodic or even approximately periodic, and we must discover those periodic time intervals containing those events which expose the fact that the events occur in regular intervals. Second, the period of these periodically occurring time intervals is not known beforehand and must be discovered. Third, the periodically occurring envelopes need not extend across the entire interval of observations, and there may be “gaps” between collections of periodically occurring envelopes which make up a routine. Fourth, events occurring in periodically occurring envelopes can be interleaved with “noise” events in the form of extra events in an envelope, and events between envelopes.

In addition to its immediate usability in understanding event properties, the solution to the problem described here has promising implications in routine analysis and sensing. In the case of the former, there is a lack of a mathematical representation of a human routine with respect to its components. The proposed algorithm exposes the properties of individual components by treating them as single events. This will help group together all the components of a human routine, allowing us to study their structure in order to develop more abstract mathematical interpretations of routines. For the latter, understanding the periodicity of events will most likely enable a new form of *top-down sensing* that could compensate for the lack of sensors for directly sensing human behaviors and routines. Instead, of requiring custom sensors, the proposed algorithm could be used to recognize routines from their indirect timing properties, allowing the sensing of complex behaviors using simple off-the-shelf sensors.

The rest of the paper describes the components of the proposed algorithm. Section II surveys the related work in the sensor networks and data mining literature. Section III provides the problem statement followed by an overview

of a simplified version of a sliding window algorithm used to determine if a candidate period of an event is indeed the period in Section IV. Section V provides an algorithm for discovering a set of candidate periods by efficiently pruning out impossible candidates. In section VI we provide the algorithm for selecting the period out of the previously generated set of periods. Section VII provides suggestions on how to use the proposed algorithm without knowledge of its parameters and section VIII presents our evaluation results using real and synthetic data. Section IX concludes the paper.

## II. RELATED WORK

In [2], the authors consider a sequence of data sets, and the goal is to look for sets of elements which occur periodically in the sets. For example, if the sequence of data sets are given as  $\mathcal{D}_1, \mathcal{D}_2, \dots, \mathcal{D}_n$ , then an element  $x$  occurs periodically if there are  $l, o < n$ , so that beginning with  $\mathcal{D}_o$ ,  $x$  occurs in every  $l$ th data set. Hence, only perfect periodicity is considered, and the original data is assumed to be pre-segmented into buckets. In [3], the authors consider a problem with the same set-up as that of [2]. However, instead of discovering elements which occur with perfect periodicity, the authors considers the problem of finding subsets of elements which occurs frequently among all the data sets. Also, the elements in each  $\mathcal{D}_i$  are considered as a sequence of symbols where order matters. In [4], the authors consider a set up where a sequence of events with time stamps and a specified window length is given. An event set is defined to be frequent if there are at least some specified number of time windows with the specified length in which the events in the set occur. The periodicity of time windows are not considered. In [5], a sequence of symbols is considered, and the time line is viewed as being composed of segments where a segment is considered “valid” if a sequence of symbols is perfectly periodic in the segment, i.e. the symbol sequence is repeated a number of times exceeding some user specified threshold. The aim in that work is to discover the combination of disjoint valid segments, with respect to a particular symbol sequence, that has the desirable maximality properties in terms of repetition and length. There is some resemblance here with our work in that we also consider imperfect periodicity where the periodicity does not hold over the entire time interval but only over a portion of it within certain limits. However, the fundamental difference here from our work, is that a symbol sequence must repeat itself without the interleaving of other noisy symbols in a time segment in order for that segment to be considered one where the symbol sequence is periodic. In [6], the time line is viewed as contiguous on/off-segments, where on-segments denote portions of the time line where the event occurs periodically or approximately periodically within some user specified “jitter factor.” In our work, we also consider an on/off-segment model, however the

type of periodicity we consider is fundamentally different. Given an event type of interest, we do not require events of that type to occur (approximately) periodically in the on-segments. Rather, we only require that time windows containing events of that type occurs periodically in the on-segments, and the event itself can occur in any random point in that time window. This is a necessary requirement because unlike previously considered applications, such as inventory replenishment, individual events comprising human routines almost never occur perfectly periodically even within a small portion of the time interval.

In [7] the authors consider a slightly different problem, where the events are modeled as lines in a 2-dimensional space representing the start time and the duration of the events. Consequently, events are clustered together, so as to maximize the overlapping of the line segments that belong to the same cluster, minimizing at the same time the number of clusters. Although this approach can provide indications of periodicity by finding events that occur at approximately the same time and having the same envelope, it does so using pre-specified temporal windows and neglecting periodicity within these windows. Moreover, it is largely dependent on the duration of the events, and attempts to cluster the events in a way that minimizes the envelopes around them. On the contrary, the algorithm presented in this paper, attempts given a class of events to determine the optimal envelope around them. In fact, our method can be combined with the algorithm presented in [7] to achieve optimal selection of windows and, hence, clusters. Concerning some simpler forms of finding periodicity. Circadian rhythms, namely discretizing time in bins of fixed size and measuring the percentage of events in these bins [8], [1], can be used to provide a rough estimation of when these events occur. However, the amount of detail provided by these methods is not adequate enough to characterize the actual periodicity of the events.

## III. PROBLEM STATEMENT

We are given a sequence of  $n$  events where all the events in the sequence are of the same type, and an occurrence time is given for each event. We assume that events are classified at a lower layer using approaches such as clustering or grammars [9], [7]. Let  $e$  denote the event type. We will consider events of the same type as equivalent, so for simplicity of notation, we will use *event*  $e$  to mean an event of *type*  $e$ . Let  $p_i$  denote the occurrence time of the  $i$ th event in the given sequence, and let  $\mathcal{P}$  denote the set of all occurrence times. So  $p_1 < p_2 < \dots < p_n$  and  $\mathcal{P} = \{p_i \mid i \in \{1, \dots, n\}\}$ . Let  $F$  and  $S$  be two user specified parameters where  $F$  determines whether events of type  $e$  occur frequently enough, and  $S$  is a real between 0 and 1 which determines if events of type  $e$  occurs over a significant enough fraction of the entire interval of observation. We call  $F$  and  $S$  the *frequency* and *support* respectively. Let  $min_{rep}$  be a user specified positive

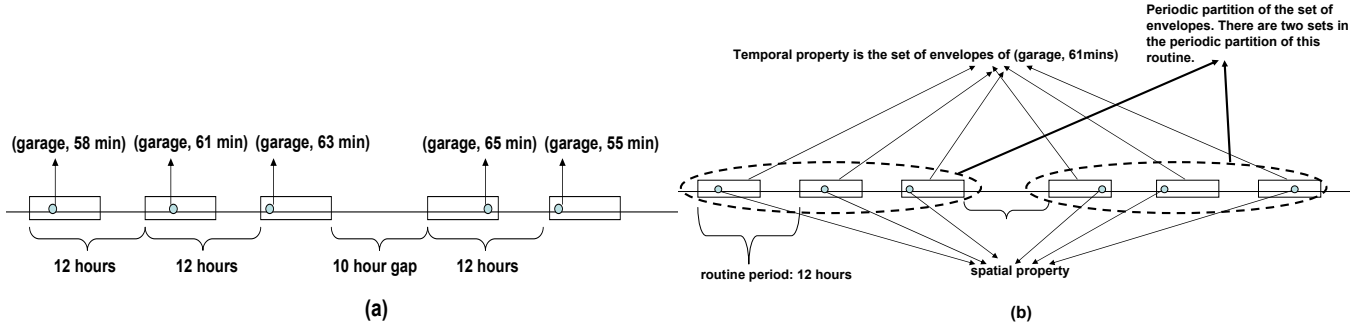


Figure 1. (a) A routine with parameters:  $S=0.7$ ,  $F = 4$ ,  $min_{rep}=2$ . (b) The temporal property, periodic partition and spatial property of the routine.

integer greater than or equal to two which determines when events of type  $e$  repeat in enough periodically occurring time intervals to qualify as a regularly reoccurring behavior. Let  $L$  denote the length of the interval of observation. We assume the interval of observation has been discretized in the chosen time unit. So if the time unit is a minute, and the interval of observation is thirty days, then the total length of the interval of observation is  $L = 24 * 60 * 30$ , and the set of all occurrence times is a subset of  $\{1, 2, 3, \dots, L\}$ .

We say that a time interval is an *envelope* of event  $e$  if an occurrence of event  $e$  begins and ends in the interval. Formally, we say that event  $e$  is a *spatio-temporal routine* (or just a *routine*), and we symbolize it with  $\{e\}$ , if there exists a set  $\mathcal{T}$  of envelopes, where all the envelopes have the same lengths, and the following hold:

- 1) The number of envelopes in  $\mathcal{T}$  is at least  $F$ :  $|\mathcal{T}| \geq F$ .  
Condition 1 says that an event must occur frequently (as deemed by user specified parameter  $F$ ) in order for it to be considered a routine.
- 2) There is a partition  $\mathcal{P}$  of  $\mathcal{T}$  such that each set in  $\mathcal{P}$  contains at least  $min_{rep}$  envelopes, and the envelopes in each set of  $\mathcal{P}$  occur periodically and with the same period. The period is required to be at least the length of each envelope in  $\mathcal{T}$ , and given any pair of sets  $\mathcal{P}_1$  and  $\mathcal{P}_2$  in  $\mathcal{P}$ , the distance between the left endpoint of any envelope in  $\mathcal{P}_1$  and the left endpoint of any envelope in  $\mathcal{P}_2$  is not equal to  $l$ . Let  $I(\mathcal{P}')$  denote the smallest time interval containing all the envelopes in  $\mathcal{P}'$  for each  $\mathcal{P}' \in \mathcal{P}$ .
- 3)  $I(\mathcal{P}')$  and  $I(\mathcal{P}'')$  do not overlap for any  $\mathcal{P}', \mathcal{P}'' \in \mathcal{P}$ , and the sum of the lengths of  $I(\mathcal{P}')$  for all  $\mathcal{P}' \in \mathcal{P}$  is at least  $SL$ .

Conditions 2 and 3 ensure that although a behavior need not reoccur regularly over the entire interval of observation to be considered a routine, it should reoccur regularly over most of the interval of observation.

When the above hold, we call  $\mathcal{T}$  a *temporal property* of the routine, and we call  $\mathcal{P}$  the *periodic partition* of  $\mathcal{T}$ . We call the period of the envelopes in each set of  $\mathcal{P}$  a *period* of the routine. The *spatial property* of the routine are the events contained in the envelopes of  $\mathcal{T}$ , where each event is

indexed by its position in the given event sequence. A routine may have more than one temporal and spatial property pair associated with it. Note that in order for event  $e$  to be a routine, we only require that there exist (a sufficient number) of envelopes which are periodic. There may be a complete lack of periodicity or even approximate periodicity among the event occurrence times themselves.

See Figure 1(a) for an illustration of a routine  $\{e\}$  with period 12 hours, where  $e$  is the event of a garage visit with duration of an hour which we denote by  $\{garage, 60minutes\}$ . The envelopes are denoted by the rectangles, and each envelope contains an event of the same type as event  $e$ .  $\mathcal{T}$  is the set of all the envelopes, and is a temporal property of the routine. The spatial property are the events contained in the envelopes. The first three envelopes occur periodically and the last three envelopes occur periodically, and there is a break in the periodicity of the envelopes between the third and fourth envelopes. A periodic partition of  $\mathcal{T}$  contains two sets, the first of which consists of the first three envelopes, and the second of which consists of the last three envelopes. See Figure 1(b) for an illustration of the routine  $\{e\}$  with its temporal properties, periodic partition, period and spatial property labeled.

An *aligned routine* with period  $l$  is a routine of period  $l$  with at least one temporal property  $\mathcal{T}$  where the distance between the left endpoints of any two intervals in  $\mathcal{T}$  is a multiple of the period  $l$ . See Figure 2(a) for a temporal property of an aligned routine. For example, suppose a person visits the kitchen for an hour on most days in the time interval between 8am and 10am for breakfast. That the person does not visit the kitchen between 8am and 10am on Monday and Tuesday correspond to gaps or interruptions in the routine; however, when the person does eat breakfast on Wednesday, one may reasonably assume he will still do so between 8am and 10am. In an aligned routine, once the routine is resumed after an interruption, it would be as if there had been no interruption.

Given a sequence of occurrence times and user specified parameters  $F, S$  and  $min_{rep}$ , our aim is to determine if  $e$  is an aligned routine, and if so, find the smallest  $l$  such that  $e$  is a routine with period  $l$ . We call this the routine discovery

problem. Towards this end, we present an algorithm for identifying a subset of those  $l'$  which *cannot* be the periods of routine  $e$ , and thus generate a set of candidate periods which can be guaranteed to contain all  $l$  such that  $e$  is a routine with period  $l$ , assuming  $e$  is a routine. Secondly, given a candidate period  $l$ , we present an algorithm with complexity polynomial in  $|\mathcal{P}|$  for determining if  $e$  is an aligned routine with period  $l$ , and if so, return a temporal property of the routine. The key challenge in this problem is that in a routine, there need not be periodicity, or even approximate periodicity, between the event occurrences. The periodicity is only required to occur between envelopes (of unknown length) containing some significant enough portion of the event occurrences, and furthermore, the period of the periodically occurring envelopes are also unknown.

#### IV. ROUTINE DISCOVERY ALGORITHM: ALIGNED ROUTINES

Given parameters  $F, S$  and  $min_{rep}$ , the routine discovery algorithm determines if there is a routine with those parameters, and the smallest  $l$  such that the event is a routine with period  $l$ . The outline of the algorithm is as follows:

- i Generate a set of candidate periods for event  $e$  which is guaranteed to contain all  $l$  where  $e$  is a routine with period  $l$ .
- ii Beginning with the smallest candidate period, determine if  $\{e\}$  is a routine with a specified candidate period. If  $\{e\}$  is a routine with the specified period, say  $l$ , then determine a spatial and temporal property pair for the routine, and return that  $\{e\}$  is a routine with period  $l$ .

We carry out step i by identifying a subset of those  $l'$  which can be guaranteed to *not* be the period of routine  $e$  assuming  $e$  is a routine. To do so, we extend the distance based pruning approaches considered in [5], [6] which are based on the observation that if an event is periodic with period  $l$  while satisfying a user specified support, then there must be a sufficient number of event pairs for which the inter-event time is  $l$ . We extend this approach to the case where the periodicity is not between event occurrences but between envelopes of unknown length.

The challenges in step ii are that the occurrence times of event  $e$  need not be periodic or even approximately periodic in order for  $\{e\}$  to be a routine, and events occurring in periodically occurring envelopes can be interleaved with “noise” events in the form of extra events in an envelope. Furthermore, there may be one or more “gaps” between sequences of periodically occurring envelopes with noise events possibly occurring in the gaps. The algorithm for determining if  $\{e\}$  is a routine with a candidate period  $l$  is based on a sliding window sequence approach. In the next section, we will illustrate the intuition behind the basic sliding window algorithm.

#### A. Intuition: Sliding window algorithm

We will first illustrate the intuition behind the sliding window algorithm for discovering aligned routines. Consider the event of a kitchen visit with duration of an hour, which we denote by  $(kitchen, 60minutes)$ , and kitchen visits which last approximately an hour are also considered to be the same type of event. Suppose the event is an aligned routine with period  $l = 24hours$ . See Figure 2(a) for a temporal property of the routine. Note that there is a sequence of contiguous time intervals, each of length  $l$ , such that each envelope in the routine belongs to one of the time intervals, and no two envelopes are in the same interval.

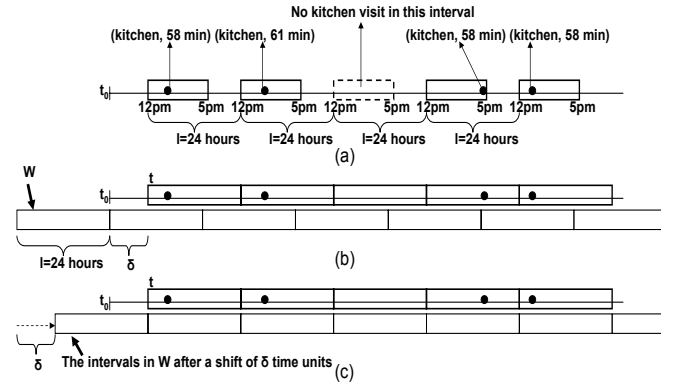


Figure 2. Spatio-temporal routine  $\{(kitchen, 60mins)\}$  occurring daily between 12pm and 5pm, with  $F = 4$ ,  $min_{rep} = 2$  and  $S = 0.6$ , with no phase shift.

Let  $L$  denote the length of the entire interval of observation, and let  $t_0$  denote the first time point on the interval. Let  $\mathcal{W}$  denote a sequence of contiguous  $\lceil \frac{L}{l} \rceil + 1$  time intervals each of length  $l$ , and where  $t_0 - l$  is the left endpoint of the first interval in the sequence:  $\mathcal{W} = [t_0 - l, t_0], [t_0, t_1], [t_1, t_2], \dots, [t_{\lceil \frac{L}{l} \rceil - 1}, t_{\lceil \frac{L}{l} \rceil}]$ , and  $t_i - t_{i-1} = l$  for each  $i$ . See Figure 2(b). Let  $t$  be the left endpoint of the earliest occurring envelope in the routine, and let  $\delta$  denote the distance between  $t$  and the left endpoint of the time interval in  $\mathcal{W}$  containing  $t$ . See Figure 2b. If we “slide” the entire sequence of time intervals in  $\mathcal{W}$  to the right by  $\delta$ , then we will discover a set of envelopes (of events with the same type as  $(kitchen, 60mins)$ ) which make up a temporal property of a routine  $\{(kitchen, 60mins)\}$  with period  $l$ . Since  $\delta$  is at most  $l$ , it follows that by sliding the sequence of time intervals in  $\mathcal{W}$  to the right one time unit at a time, we will, after at most  $l$  time units, discover that  $(kitchen, 60mins)$  is a routine of period  $l$  with parameters  $F = 4$ ,  $min_{rep} = 2$  and  $S = 0.6$ . See Figure 2(c).

#### V. GENERATING CANDIDATE PERIODS

In this section, we present an algorithm that identifies in  $O(|P| \log(L))$  steps all the candidate periods for the set of occurrence times  $P$ . Our algorithm is based on a set of constraints that eliminate values that cannot be valid periods.

It is easy to observe that the maximum possible period is obtained when our routine  $\{e\}$  reoccurs only  $F$  times over the entire interval of observation, in which case the maximum possible period will be  $l_{max} = \frac{L}{F}$ . Similarly, the minimum possible period  $l_{min}$  is obtained when all given occurrences of  $e$  are included in the routine, and the routine reoccurs over an interval of length  $SL$ :  $l_{min} = \frac{SL}{|\mathcal{P}|}$ . In order  $e$  to be a routine with period  $l$ , where  $l_{min} \leq l \leq l_{max}$ , there have to be at least  $F_l = \max(F, \frac{SL}{l})$  envelopes in the temporal property of  $\{e\}$ , since we need to have more envelopes than our frequency  $F$  and we need to have at least one envelope in every set of the periodic partition of the temporal property. A pair of envelopes in the temporal property are *adjacent* if the distance between their left endpoints is  $l$ . The number of adjacent pairs of envelopes in a temporal property of the routine is minimized when there are only  $F_l$  envelopes in the temporal property, and the number of envelopes in each set of the periodic partition of the temporal property is  $min_{rep}$ . Hence, there are at least  $Q(l) = \max(min_{rep} - 1, \lfloor \frac{F_l}{min_{rep}} \rfloor (min_{rep} - 1))$  pairs of adjacent envelopes in any temporal property of routine  $\{e\}$  with period  $l$ , where  $\lfloor \frac{F_l}{min_{rep}} \rfloor$  is the maximum number of sets in  $P$  that allows each of the sets to contain at least  $min_{rep}$  envelopes. The difference between the occurrence times of any two non-identical events is called *inter-occurrence* time. Clearly, all inter-occurrence times of the events in two adjacent envelopes can range between 0 and  $2l$ . These observations lead us to the following necessary condition for  $l$  to be a period of routine  $\{e\}$ :

*Claim 1:* If  $\{e\}$  is a routine with period  $l$ , then  $l_{min} \leq l \leq l_{max}$ , and there are at least  $Q(l)$  inter-occurrence times  $t$  where  $0 < t < 2l$ .

Any  $l$  that satisfies Claim 1 is called a *candidate period* of spatio-temporal routine  $\{e\}$ . The cardinality of the candidate period set will depend on the given set of occurrence times and the parameters  $F$  and  $S$  and will include all actual periods. Following, we introduce a procedure that uses Claim 1 to determine if  $l$ ,  $l_{min} \leq l \leq l_{max}$ . The procedure uses dynamic programming principles to check if there are more than  $Q(l)$  inter-occurrence times that are in the range  $(0, 2l)$ . The basic premise is that if the time between  $p_j$  and  $p_k$ , where  $j < k$ , is greater than  $2l$ , then the inter-start time between  $p_i$  and  $p_k$  must be greater than  $2l$  for all  $i < j$ . The procedure examines all times in  $\mathcal{P}$  one by one beginning with  $p_1$ . Let  $C(l)$  be the number of inter-start times we have found at each step of the procedure which are in the interval  $(0, 2l)$ . To begin with, let  $i = 2$  and let  $C(l) = 0$ :

#### Step 0:

- If  $C(l) \geq Q(l)$ , then stop, and return that  $l$  is a candidate period.
- If  $i = |\mathcal{P}| + 1$  and  $C(l) < Q(l)$ , then stop, and return that  $l$  is not a candidate period.

- If none of the above holds, goto Step 1.

**Step 1:** If  $|p_i - p_{i-1}| \geq 2l$ , then let  $i = i + 1$  and go to step 0; else, let  $j = i - 1$  and go to Step 2.

**Step 2:** If  $|p_i - p_j| < 2l$ , then let  $C(l) = C(l) + 1$ , and if  $C(l) \geq Q(l)$ , then stop, and return that  $l$  is a candidate period, else let  $j = j - 1$  and repeat Step 2. If  $|p_i - p_j| \geq 2l$ , then let  $i = i + 1$  and go to Step 0.

The complexity of the procedure will be worst case linear in  $|\mathcal{P}| + Q(l)$ . However, the procedure will have to be run for every  $l$  in  $(0, 2l)$ . An additional observation that can help us drastically eliminate the times we need to run the above procedure is that if  $l$  is not a candidate period and, consequently, there are less than  $Q(l)$  inter-occurrence times that lie between 0 and  $2l$ , any number less than  $l$  cannot be a period since it will also have less than  $Q(l)$  inter-occurrence times inside  $(0, 2l)$ . Using similar arguments, when  $l$  is a candidate period every number greater than  $l$  will also be a candidate period. These observations are summarized in the following lemma:

*Lemma 1:* Suppose  $\lfloor \frac{F}{min_{rep}} \rfloor \geq \frac{min_{rep}}{(min_{rep}-1)}$ . If  $l$ , where  $l_{min} \leq l \leq l_{max}$ , is not a candidate period of routine  $\{e\}$ , then no  $l'$  where  $l' < l$  is a candidate period of routine  $\{e\}$ . Also, if  $l$  is a candidate period, then all  $l'$  where  $l_{max} \geq l' > l$ , are also candidate periods.

Lemma 1, suggests a divide and conquer approach for determining efficiently the smallest  $l \in [l_{min}, l_{max}]$  that is a candidate period. This will allow us to consider as the set of candidate periods the interval  $[l, l_{max}]$ . Our algorithm proceeds similarly to binary search and uses the procedure introduced before to check if the middle point of our current subinterval is a candidate period. If it is a candidate period then the subinterval on its right is checked, otherwise the smallest candidate period is contained in the left subinterval. Based on this, we next give a procedure of complexity in the order of  $\log(l_{max} - l_{min})|\mathcal{P}|$  for finding  $l$ . In the following, let  $U$  denote initially the interval  $[l_{min}, l_{max}]$ :

**Step 0:** Let  $a$  be the left endpoint of  $U$  and let  $b$  be the right endpoint of  $U$ , and let  $l = \lfloor \frac{b-a}{2} \rfloor + a$ . If  $l$  is a candidate period then goto Step 1; else goto Step 2.

**Step 1:** If  $l - 1 \geq a$ , then let  $U = [a, l - 1]$ , and goto Step 0; else return  $l$ .

**Step 2:** If  $l + 1 \leq b$ , then let  $U = [l + 1, b]$ , and goto Step 0; else return  $l + 1$ .

The number of candidate periods which are generated is dependent on the routine parameters  $F$ ,  $S$ ,  $min_{rep}$  and the given occurrence times. To prune further the set of candidate periods, the following is a necessary condition which candidate periods must satisfy in order to be an actual period of a routine  $e$ . If  $\{e\}$  is a routine with period  $l$ , then

for  $i > 1$ , there are at least  $\lfloor \frac{F}{\min_{rep}} \rfloor (\min_{rep} - i)$  inter-occurrence times in the interval  $((i-1)l, (i+1)l)$ . Modifying the procedure given above to take into account this condition is straightforward.

## VI. DISCOVERING ALIGNED ROUTINES WITH GIVEN CANDIDATE PERIOD

In the previous section we provided an algorithm for extracting a set of candidate periods given our input sequence. In this section, we will continue by providing an algorithm that given each candidate period determines whether the candidate period is an actual period of the routine  $\{e\}$ . The algorithm uses a sliding window approach to discover in time  $O(|P|^2)$  one of the possible temporal properties of the routine  $\{e\}$  given a period  $l$ , provided such a property exists.

Following, we say that a time interval (equivalently window, as we define later on) *contains* an occurrence of event  $e$ , or just event  $e$ , if the event is contained inside the given time interval, namely there is a  $p \in \mathcal{P}$  such that the start time and end time of  $p$  are both contained in the interval. We say that a time interval  $(a, b]$  is a *valid segment* if all the following conditions hold:

- the length of the interval is a multiple of  $l$ ,
- $\lfloor \frac{b-a}{l} \rfloor \geq \min_{rep}$ , and
- each interval (window)  $(a+il, a+(i+1)l]$  contains an occurrence of event  $e$  for  $i \in \{0, 1, 2, \dots, \lfloor \frac{b-a}{l} \rfloor - 1\}$ .

Hence,  $e$  is a routine with period  $l$  if and only if there are non-overlapping valid segments the sum of whose lengths is  $\theta$  where  $\theta \geq SL$ , and  $\frac{\theta}{l} \geq F$ . We remind that  $SL$  is the interval over which the routine reoccurs and  $F$  is the support.

Based on these requirements, our algorithm proceeds by splitting  $L$ , the entire interval of observation, to a proper number of intervals or windows of length  $l$  and counting the number of adjacent windows that contain the event  $e$ . More specifically, given a candidate period  $l$ , let  $N = \lceil \frac{L-l}{l} \rceil$  be the smallest number of windows of length  $l$  whose total length is greater than  $L$ . Let  $\mathcal{W}_0$  denote a sequence of  $N$  intervals each of length  $l$ , where  $-l$  is the left endpoint of the first interval in the sequence, namely  $\mathcal{W}_0 = (-l, 0], (0, l], (l, 2l], \dots, ((N-2)l, (N-1)l]$ . Note that as time 0 is considered the occurrence time of the first event  $p_0 \in \mathcal{P}$ . However, the endpoints of the envelope including the events of routine  $\{e\}$  might not be both included in the same window of  $\mathcal{W}_0$ . Consequently, we will need to consider all possible shifts of  $\mathcal{W}_0$ . Hence, for any  $\lambda$  where  $0 \leq \lambda \leq l$ , let  $\mathcal{W}_\lambda$  denote the sequence of time intervals obtained by shifting  $\mathcal{W}_0$  to the right by  $\lambda$  time units:  $\mathcal{W}_\lambda = (-l + \lambda, \lambda], (\lambda, l + \lambda], (l + \lambda, 2l + \lambda], \dots, ((N-2)l + \lambda, (N-1)l + \lambda]$ .

Also, let  $\text{win}_\lambda(p) = \lceil \frac{p - \lambda + l}{l} \rceil$  indicate the window where  $p \in \mathcal{P}$  is contained. Clearly,  $\text{win}_\lambda(p) = i$  if  $p$  is contained in the  $i$ th window of  $\mathcal{W}_\lambda$  unless  $p$  is the left endpoint of the  $i$ th interval of  $\mathcal{W}_\lambda$  (or equivalently, if  $p$  is the right endpoint of the  $(i-1)$ th interval of  $\mathcal{W}_\lambda$ ), in which case

$\text{win}_\lambda(p) = i - 1$ . To check for a routine our algorithm will go through all events in the sequence combining adjacent windows to form valid segments. A significant speed up of our algorithm can be achieved by not considering values of  $\lambda$  that cannot lead to valid segments. If  $\delta_\lambda(p)$  denotes the distance between  $p$  and the left endpoint of the interval of  $\mathcal{W}_\lambda$  where  $p$  is contained, namely  $\text{win}_\lambda(p)$ , then it is easy to observe that there is no reason to increase  $\lambda$  by values less than  $\min_{p \in \mathcal{P}} \delta_\lambda(p)$ , since no window that didn't already contain an event will start containing one.

Formalizing the above discussion, we now introduce a detailed algorithm. For ease of exposition, we will in the following assume there is an ordering  $q_1, q_2, \dots, q_n$  of the given occurrence times in  $\mathcal{P}$  where  $0 < \delta_0(q_1) < \delta_0(q_2) < \dots < \delta_0(q_n)$ . The algorithm to be outlined below can be easily generalized to the case where the previous condition does not hold, and the implementation of the algorithm is of the generalized version. Let  $\mathcal{V}$  be any set of valid segments, and let  $\mathcal{T}(\mathcal{V})$  denote the set of all windows contained in some valid segment. The algorithm for finding a temporal property of  $\{e\}$  given the set of candidate periods follows. As in the rest of this section, the algorithm assumes that the interval of observation begins at time 0:

For  $k = 0$  to  $n$ :

- **Step 0**  
If  $k = 0$ , then let  $\lambda = 0$ ; else, let  $\lambda = \delta_0(q_k)$ . Let  $\mathcal{S}_\lambda = \emptyset$ ,  $\text{left} = \text{win}_\lambda(p_1)$ ,  $\alpha = \text{left}$ ,  $\beta = \alpha$ ,  $i = 2$ , and  $\text{reps} = 0$ .
- **Step 1**  
Let  $\beta = \text{win}_\lambda(p_i)$ , and goto Step 2.
- **Step 2**  
If  $\beta \leq \alpha + 1$ :  
Let  $\alpha = \beta$ .  
Let  $i = i + 1$ . If  $i \leq n$  then goto Step 1,  
else if  $i > n$  then goto Step 3.  
If  $\beta > \alpha + 1$  then goto Step 3.
- **Step 3**  
Let  $\text{right} = \alpha$  and let  $\alpha = \beta$ .  
If  $(\text{right} - \text{left} + 1) \geq \min_{rep}$ , then let  $\mathcal{S}_\lambda = \mathcal{S}_\lambda \cup \{[(\text{left} - 1)l - l + \lambda, (\text{right})l - l + \lambda]\}$ , and let  $\text{reps} = \text{reps} + (\text{right} - \text{left} + 1)$ .  
If  $(N - \alpha + 1) < \min_{rep}$  or  $(n - i + 1) < \min_{rep}$ , then goto Step 4.  
Let  $i = i + 1$ , set  $\text{left} = \alpha$  and goto Step 1.
- **Step 4**  
If  $\text{reps} \geq F$  and  $(\text{reps})l \geq SL$ , then stop the algorithm and return  $\mathcal{T} = \mathcal{T}(\mathcal{S}_\lambda)$ ; otherwise, if  $k = n$ , then return  $\mathcal{T} = \emptyset$ .

*Lemma 2:*  $\{e\}$  is an aligned routine with period  $l$  if and only if the output of the above algorithm  $\mathcal{T}$  is not the empty set. Furthermore, if  $\mathcal{T}$  is not empty, then  $\mathcal{T}$  is a temporal

property of routine  $e$  with period  $l$ .

The complexity of each iteration of the above procedure is linear in  $n$ , where we remind that  $n = |\mathcal{P}|$ , and since there are at most  $n$  iterations, we have that the complexity of the above algorithm is polynomial in  $n$ , the number of occurrence times.

So far, we have silently assumed that our routines are aligned. In our technical report [10], we give a polynomial time algorithm with respect to  $|\mathcal{P}|$  for general routines which may not be aligned. The algorithm for the general case is based on the algorithm for aligned routines given above, and the output of the algorithm is a set  $\bar{\mathcal{V}}$  of non-overlapping valid segments. Our main result for the general case is the following. For any interval  $v$ , let  $len(v)$  denote the length of  $v$ . If  $e$  is a routine with a temporal property  $\bar{T}$  with periodic partition  $\mathcal{P}$  and period  $l$  such that: (i) For each  $\mathcal{P}' \in \mathcal{P}$ :  $I(\mathcal{P}')$  is a subinterval of a valid segment with length at least  $3min_{rep}l$ , (ii)  $\sum_{\mathcal{P}' \in \mathcal{P}} \frac{min_{rep}-1}{min_{rep}} \frac{len(I(\mathcal{P}'))}{l} \geq F$ , and (iii)  $\sum_{\mathcal{P}' \in \mathcal{P}} \frac{min_{rep}-1}{min_{rep}} len(I(\mathcal{P}')) \geq SL$ , then the set  $\mathcal{T}(\bar{\mathcal{V}})$  is a temporal property of routine  $\{e\}$  with period  $l$ .

## VII. DISCOVERING PERIODS OF ROUTINES WITH UNKNOWN PARAMETERS

Given a sequence of event occurrence times, if the user already knows the parameters of the kind of routine he is interested in, then the routine discovery algorithm can be used to determine if the event is a routine with those parameters, and a temporal property of the routine if one exists. For example, given a sequence of  $n$  occurrence times for bathroom visits, the user may like to know if there is a routine with parameters  $F = 0.90n$ ,  $min_{rep} = 3$  and  $S = 0.7$ . More often than not however, it may be that a user suspects or knows a particular sequence of event occurrence times is the result of some routine being performed, and the user would like to confirm that there is a routine, and discover the period of the routine. In this case, the routine discovery algorithm cannot be used directly to discover the period of routines because the parameters of the routine, i.e.,  $F$ ,  $S$  and  $min_{rep}$ , are not known, and different parameters can result in routines with different periods being discovered. Hence, we will in the following give an algorithm, which we call *period discovery*, that builds on the routine discovery algorithm to find periods of routines with unknown parameters. In Section VIII we will demonstrate the effectiveness of the period discovery procedure using artificially generated and actual routines.

Let  $F = \frac{n}{2}$ ,  $min_{rep} = 2$  and  $S = 0.5$  be the “minimal” settings for the routine parameters. These settings are minimal in the sense that a routine, if one exists, should cover at least 50% of the interval of observation and contain at least half of the occurrence times. Let  $F^{max}$  be the largest frequency  $F$  such that the given event is a routine with parameters  $F = F^{max}$ ,  $min_{rep} = 2$  and  $S = 0.5$ . Let

$min_{rep}^{max}$  be the largest  $min_{rep}$  such that the given event is a routine with parameters  $F = F^{max}$ ,  $min_{rep} = min_{rep}^{max}$  and  $S = 0.5$ . And let  $S^{max}$  be the largest support  $S$  such that the given event is a routine with parameters  $F = F^{max}$ ,  $min_{rep} = min_{rep}^{max}$  and  $S = S^{max}$ . Given a particular setting for the parameters  $F, S$  and  $min_{rep}$ , the routine discovery algorithm determines if there is a routine with those parameters, and the smallest period  $l$  such that the event is a routine with those parameters. The period discovery procedure takes as its input the sequence of event occurrence times, and finds  $F^{max}$ ,  $min_{rep}^{max}$  and  $S^{max}$  in that order using the routine discovery algorithm. The period discovery procedure outputs the period found by the routine discovery algorithm with the parameters  $F = F^{max}$ ,  $min_{rep} = min_{rep}^{max}$  and  $S = S^{max}$ . In our implementation of the period discovery procedure, we approximate  $S^{max}$  in increments of 0.1.

## VIII. EXPERIMENTAL EVALUATIONS

The period discovery algorithm given in Section VII was evaluated with a mix of tests using artificially generated data as well as data collected by our deployed testbeds [1]. The artificial data was used to stress test the algorithms while also trying to preserve the similarity to data generated by humans. The evaluation is comprised of four main components. The first component demonstrates the effectiveness of the period discovery algorithm using bedtime data collected by our testbed. The second component verifies the basic properties of the period discovery algorithm by evaluating the algorithm on artificially generated sequences of event occurrence times which satisfy the requirements to be a routine with a specified period and routine parameters  $F$ ,  $S$  and  $min_{rep}$ . More specifically, the generated routines are comprised of numerous valid segments (routine occurrences) and gaps (interruptions in routine occurrences) between the valid segments. The noise in the form of “extra” events which do not occur as part of the routine is determined by a parameter  $\eta$  which is the fraction of event occurrence times which are chosen uniformly randomly from the interval of observation. The third set of experiments evaluates the algorithm on artificially generated event occurrence times which are generated *without* using routine parameters  $F$ ,  $S$  and  $min_{rep}$ . Instead, beginning at some random offset, the interval of observation is considered as a sequence of contiguous intervals of length  $l$  where  $l$  is the period of the routine, and, ideally, an event occurrence time is chosen uniformly randomly from each of the intervals. To approximate sensor failure and routine interruptions, a user specified parameter  $\gamma$  determines the probability that an event occurrence time is not generated for an interval in the sequence, and as before, a parameter  $\eta$  determines the fraction of event occurrences which are generated as noise. The last set of experiments evaluates the period discovery algorithm on artificially generated event occurrence times

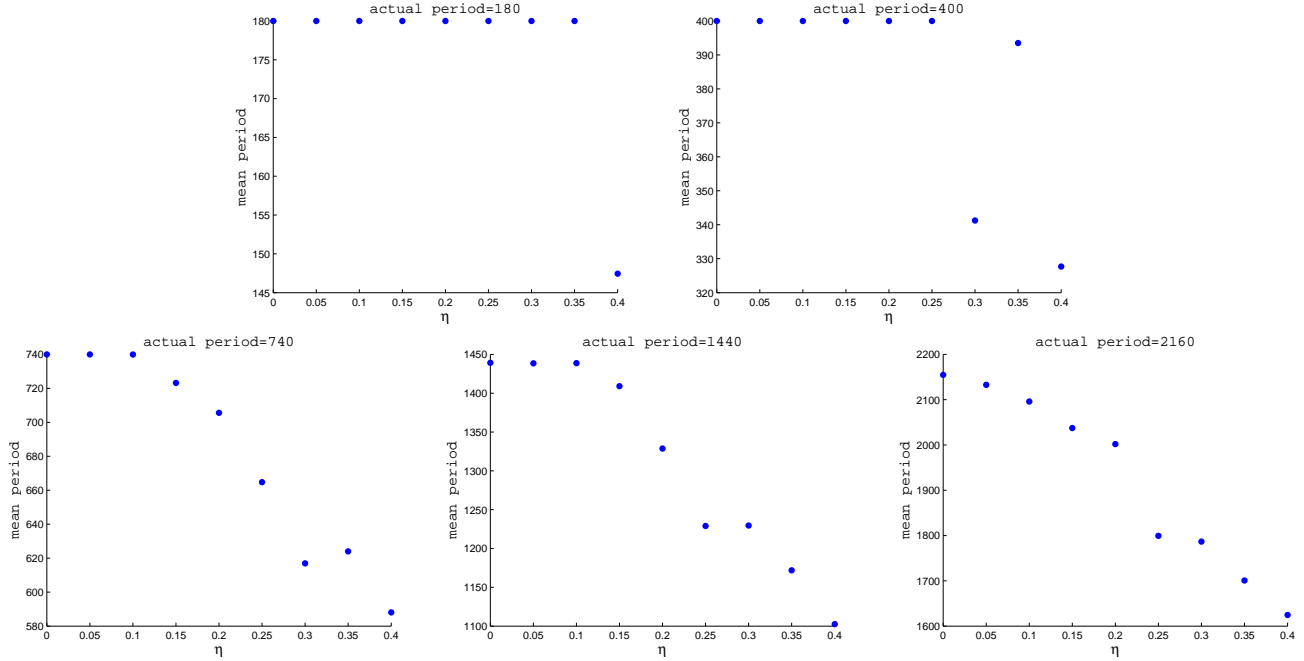


Figure 3.

which comprise a routine with “build-in gaps”. In the routines considered in the previous experiments, gaps in routines are viewed as the result of routine interruptions or sensor failures. However, some routines have build-in gaps in which periodically occurring intervals of identical lengths generally do not have occurrences of the event. As before, user specified parameters  $\eta$  and  $\gamma$  determine respectively the probability that a routine is interrupted and the noise level.

#### A. Basic case using testbed data

To test the base case of the algorithm we considered a sequence of bedtimes collected from our testbed over an interval of one month. In the data set, bed times often had differences on the order of two to three hours across different days, i.e., 10:23pm versus 1am. Using the period discovery procedure, we find a period of 1440 minutes, i.e., 24 hours for a dataset with no gaps. Another dataset with some gaps in the data, where the person did not stay at home during some of the weekends resulted in a similar outcome; 1435 minutes which is approximately 24 hours.

This test demonstrates the basic properties of the algorithm using an obvious case, that of human bedtime. Other types of analysis of the data which only takes into account periodicity, or even approximate periodicity, between the sleep times, would likely be insufficient.

#### B. Generating event occurrence times using routine parameters

The next test verifies the basic properties of the period discovery algorithm, and evaluates the robustness of the

algorithm in the presence of noise and routine interruptions. In terms of human behavior, noise will manifest itself as the occurrence of an event outside the expected form (i.e the occurrence of a bedtime event in the middle of the day that represents a very occasional nap). For this we artificially generated a set of routines with some specified period  $l$  and routine parameters  $F$ ,  $S$  and  $min_{rep}$  which are composed of numerous valid segments, where the “gaps” between the valid segments denoting routine interruptions have lengths which are random integer multiples of the specified period. A noise parameter  $\eta$ , a real number between 0 and 1, determines how many of the event occurrence times are chosen uniformly randomly from the entire interval of observation; i.e., if  $\eta = 0.1$ , then enough random event occurrence time are generated so that 10% of the event occurrence times are not generated specifically for the routine, but are chosen randomly from the interval of observation. For evaluation purposes we generated routines where the finest time unit corresponds to 1 minute and the length  $T$  of the interval of observation is two months. The results for different specified periods  $l$ , where  $l = 180, 400, 740, 1440, 2160$  minutes, are shown in Figure 3. For each specified period  $l$ , we generate twenty different sequences of event occurrence times which satisfy the requirements to be a routine with the specified period  $l$  and routine parameters  $F, S$  and  $min_{rep}$ . Each sequence of event occurrence times is used as input to the period discovery procedure. The periods found by the period discovery algorithm are then averaged over the twenty trials to obtain a mean period. We find that the period discovery procedure is in general effective at determining the actual

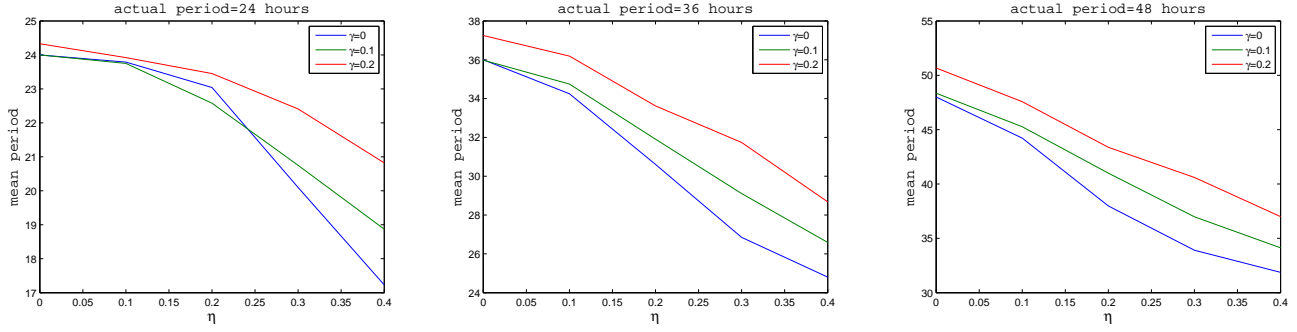


Figure 4.

period of the routine when noise parameter  $\eta$  is less than or equal to 0.2.

### C. Generating event occurrence times without routine parameters

In this test we generate event occurrence times without using routine parameters  $F$ ,  $S$  and  $min_{rep}$ , and thus more closely approximate how an actual sequence of event occurrence times are obtained. We take as the basic time unit 1 hour. We use four parameters to generate a sequence of event occurrence times, namely  $l$ ,  $T$ ,  $\eta$  and  $\gamma$ . The parameter  $T$  is the length of the interval of observation, and  $l$  specifies the period of the event. So if  $l = 3$ , then beginning at some random offset  $\omega$  the event should ideally occur in each time interval  $[(i-1)l + \omega, il + \omega]$ , for  $i = 1, 2, 3, \dots$ , over the entire interval of observation. For each of the intervals  $[(i-1)l + \omega, il + \omega]$ ,  $i = 1, 2, 3, \dots$ , the parameter  $\gamma$  determines the probability that an event which should occur in the interval does not occur in the interval. Hence,  $\gamma$  is the probability that either a sensor malfunctions and does not observe an event which took place, or that the routine is interrupted. The noise parameter  $\eta$  is between 0 and 1, and determines how many of the event occurrences are *not* generated specifically as part of the routine, but are instead chosen uniformly randomly from the interval of observation. If  $\eta = 0.1$  for example, then enough occurrence times are chosen uniformly randomly from the entire interval of observation so that they make up 10% of the total number of event occurrences. The results for different specified periods  $l$ , where  $l = 24, 36, 48$  hours, and different settings for  $\eta$  and  $\gamma$ , are shown in Figure 4. For each specified period  $l$  and the specified  $\eta$  and  $\gamma$ , we generate 100 different sequences of event occurrence times. Each sequence of event occurrence times is used as input to the period discovery algorithm. The periods found by the period discovery algorithm are averaged over the 100 trials to obtain a mean period. From this test, we find that the actual period of a routine can be closely approximated by the period discovery algorithm when the parameters  $\gamma$  and  $\eta$  are less than or equal to 0.2.

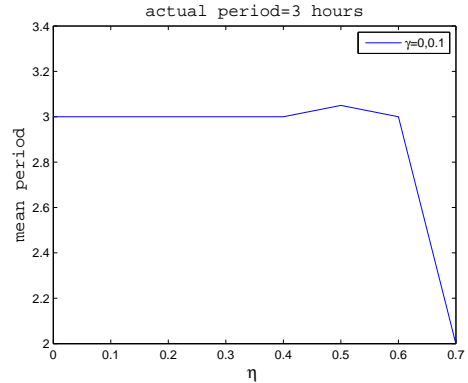


Figure 5.

### D. Routines with build-in gaps

The routines we considered above do not have any “build-in” gaps. An example of a routine with a build-in gap is an event which (ideally) occurs in the six hour intervals  $[5am, 11am]$ ,  $[11am, 5pm]$ , and  $[5pm, 11pm]$  every day, but does not occur in the interval  $(11pm, 5am)$ . The interval  $(11pm, 5am)$  is the build-in gap of the routine, and the period of the routine is six hours. The *repetition factor* of the routine is three since each occurrence of the routine involves the event occurring in three consecutive intervals before a build-in gap occurs. A routine with a build-in gap correspond to meals and bathroom visits where the event presumably would not occur when the subject is sleeping very soundly in the middle of the night. We generate sequences of event occurrence times corresponding to routines with a build-in gap as follows. A period, say  $l = 6$  hours is first specified, and also the time interval of the day over which the routine occurs, i.e., an event occurs in each of the intervals  $[5am, 11am]$ ,  $[11am, 5pm]$ , and  $[5pm, 11pm]$  everyday. As before, we let  $\gamma$  denote the probability that an event does not occur in an interval in which it is suppose to occur, and we let the noise parameter  $\eta$  denote the fraction of the total number of event occurrence times which are chosen uniformly randomly from the interval of observation.

We first generated sequences of event occurrence times

for a routine where the event occurs in the five intervals  $[5am, 8am]$ ,  $[8am, 11am]$ ,  $[11am, 2pm]$ ,  $[2pm, 5pm]$ ,  $[5pm, 8pm]$  every day (ideally), with a build in gap in the interval between 9pm and 6am of the next day. This routine has period  $l = 3$  and repetition factor five. For each setting of  $\eta$  and  $\gamma$ , we generate twenty sequences of event occurrence times using the specified  $\eta$  and  $\gamma$ , and after inputting each sequence into the period discovery procedure, we average the results over the 20 trials to obtain a mean period. We find that for  $\gamma \leq 0.1$  and  $\eta < 0.4$ , the mean period approximated closely the actual period of three hours. See Figure 5. We next generated sequences of event occurrence times for a routine where the event occurs in the three intervals  $[6am, 12pm]$ ,  $[12pm, 6pm]$  and  $[6pm, 12am]$  every day (ideally), with a build in gap in the interval between 12am and 6am. This routine has period  $l = 6$  and repetition factor three. We obtain a mean period for each setting of  $\eta$  and  $\gamma$  in the same manner as before. As in the case for  $l = 3$ , we find that for  $\gamma \leq 0.1$  and  $\eta < 0.4$ , the mean period approximated closely the actual period.

We next considered a routine with a build-in gap and a repetition factor of two. More specifically, the routine is of an event which occurs in the intervals  $[8am, 4pm]$  and  $[4pm, 12am]$ , with a build-in gap of  $(12am, 8am)$ . The repetition factor here is two. We found that even under ideal conditions where  $\eta = 0$  and  $\gamma = 0$ , the period discovery routine determines a period of 12 hours, and not the actual period of 8 hours. The problem here is that when the repetition factor is less than three, there are multiple ways to “parse” the sequence of event occurrences. In other words, an event which occurs within 24 hours in two consecutive intervals of 8 hours followed by a build in gap of 8 hours can also be said to occur every twelve hours with no build-in gaps. Without prior knowledge of whether build-in gaps are present or not, it is impossible for the period discovery procedure to choose between 8 and 12. Such alternative parsings are less likely when repetition factor is three or more. While we have no theoretical results towards this end, we find from the experimental evaluations that the period discovery algorithm can be used to effectively approximate the actual routine period when the repetition factor of the routine is greater than two.

## IX. CONCLUSION

In this paper we have presented an algorithm for determining the period of a single event routine. Our simulations and analytical proofs in our technical report [10] have verified, that the algorithm works correctly and has reasonable response to noise to be useful in practical applications. Our evaluation used testbed data for the case of bedtime, a single event routine that is easy to study in the absence of ground truth data. The results in this paper demonstrate that the algorithm has practical applications on single event data but also has interesting implications for events and human

routines that are hard to sense with simple sensors. By being able to identify which sensor events occur with the same periodicity, we hope that in the near future we will be able to consume the sensor measurements collected by our testbeds in new ways. The fact that we can discover events with the same periodicity suggests that we will be able to identify the components of a routine in a more abstract way. In the immediate future, our plan is to apply the algorithm on our collected datasets, some of which spanning more than one year, to identify and study the components of routines that happen periodically. The outcomes of this analysis will be targeted to human behavior analysis in the study of depression and cognitive decline processes.

## REFERENCES

- [1] A. Bamis, D. Lymberopoulos, T. Teixeira, and A. Savvides. Towards precision monitoring of elders for providing assistive services. In *Proceedings of the 1st international conference on Pervasive Technologies Related to Assistive Environments*. ACM New York, NY, USA, 2008.
- [2] Banu Özden, Sridhar Ramaswamy, and Abraham Silber-schatz. Cyclic association rules. In *Proceedings of ICDE '98*, pages 412–421, 1998.
- [3] Jiawei Han. Efficient mining of partial periodic patterns in time series database. In *Proc. Int. Conf. on Data Engineering*, pages 106–115, 1999.
- [4] Heikki Mannila, Heikki Mannila, Hannu Toivonen, Hannu Toivonen, A. Inkeri Verkamo, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1:259–289, 1997.
- [5] Jiong Yang, Wei Wang, and P.S. Yu. Mining asynchronous periodic patterns in time series data. *Knowledge and Data Engineering, IEEE Transactions on*, 15(3):613–628, May-June 2003.
- [6] Heng Ma and Joseph L. Hellerstein. Mining partially periodic event patterns with unknown periods. In *Proc. ICDE*, pages 205–214, 2001.
- [7] D. Lymberopoulos, A. Bamis, and A. Savvides. A methodology for extracting temporal properties from sensor network data streams. In *Proceedings of MobiSys '09*, New York, NY, USA, 2009. ACM.
- [8] A. Wood, G. Virone, T. Doan, Q. Cao, L. Selavo, Y. Wu, L. Fang, Z. He, S. Lin, and J. Stankovic. ALARM-NET: Wireless sensor networks for assisted-living and residential monitoring. *University of Virginia Computer Science Department Technical Report*, 2006.
- [9] D. Lymberopoulos, A.S. Ogale, A. Savvides, and Y. Aloimonos. A sensory grammar for inferring behaviors in sensor networks. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 251–259. ACM New York, NY, USA, 2006.
- [10] J. Fang, A. Bamis, and A. Savvides. Discovering recurring events with unknown periods. Technical report, Yale University, New Haven, CT., May 2009.