Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 69 (2015) 76 – 85

Procedia Computer Science

7th International Conference on Advances in Information Technology

Mining top-*k* regular episodes from sensor streams

Komate Amphawan[a,*], Julie Soulas[b], Philippe Lenca[b]

[a] *Computational Innovation Laboratory, Faculty of Informatics, Burapha University, Chonburi, 20131, Thailand*

[b] *Institut Mines-Telecom, Telecom Bretagne UMR 6285 Lab-STICC, Technopôle Brest Iroise CS 83818 29238 Brest Cedex 3, France*

Abstract

The monitoring of human activities plays an important role in health-care applications and for the data mining community. Existing approaches work on activities recognition occurring in sensor data streams. However, regular behaviors have not been studied. Thus, we here introduce a new approach to discover top-*k* most regular episodes from sensors streams, *TKRES*. The top-*k* approach allows us to control the size of the output, thus preventing overwhelming result analysis for the supervisor. *TKRES* is based on the use of a simple top-*k* list and a *k*-tree structure for maintaining the top-*k* episodes and their occurrence information. We also investigate and report the performances of *TKRES* on two real-life smart home datasets.

Peer-review under responsibility of the Organizing Committee of IAIT2015

Keywords: Data mining; Activities of Daily Living; Episode discovery; Data stream; Sliding window; Regularity

1. Introduction

Due to improvements in medicine and quality of life, people now live longer, and the proportion of elderly people increases around the world. However, the aging populations are frailer than the younger generations. One of the big challenges of the coming years is thus to help them keep living independently at home, comfortably and safely. Thanks to the development of sensor technologies, smart homes and ambient assisted living systems have spread out over the last decade [16].

The sensors and devices disseminated in the house register traces of the activity in the home setting. Activity reflects health, and mining these traces is thus most informative on the person's health condition. A huge part of the current research focuses on supervised activity recognition, and use for example hidden Markov models and conditional random fields [22], emerging patterns [8] or support vector machines [11]. These techniques use annotated data, which is hard and expensive to get [5]. The annotation process also needs to be performed again for every home setting and patient, since it generalizes poorly.

There is thus a growing interest for unsupervised analysis techniques, such as event streams partitioning and clustering [3], or frequent [15] and periodic [17] episode discovery. The relationships between the episodes are investigated as well [9]. However promising these techniques appear, the human activity that generated the events is not explicitly

* Corresponding author. Tel.: +66-3810-3097; fax: +66-3839-3245.
E-mail address: komate@gmail.com

Peer-review under responsibility of the Organizing Committee of IAIT2015
doi:10.1016/j.procs.2015.10.008

identified. The analysis and understanding of the results may thus be overwhelming for the supervisor (physician, caregiver). One way to reduce this downside is to reduce the number of extracted patterns to a user-defined suitable amount, or to use different metrics to assess the interest of the patterns, such as regularity or length. Such analyses can help to optimize energy consumption or to detect anomalies better than if we rely solely on frequency.

Thus, we propose *TKRES* for the discovery and update of the top-k most regular episodes in an event stream. The rest of the paper is organized as follows: section 2 presents the formalisms and defines the problem at hand. Section 3 reviews some of the associated literature, namely episode discovery, regular pattern mining, and event stream handling. Section 4 presents our contribution, and section 5 present the performance evaluation of *TKRES*. Finally, section 6 concludes and presents lines of study for future work.

2. Problem definition

We here present the concepts used for the discovery of episodes from sensor streams. We also introduce the problem of mining top-k regular episodes.

A stream of events is a potentially infinite *sequence of ordered events*:

$$DS = \langle (e_1, t_1), (e_2, t_2), \dots (e_i, t_i), \dots \rangle$$

where (e_i, t_i) is the i^{th} event in the sequence, with e_i the *event label*, taking values in a finite alphabet ξ; and t_i the *timestamp* of the event. The events are ordered by their timestamps (for all i, $t_i \leq t_{i+1}$).

An *episode* $E = \{e_1, e_2, \dots e_n\}$ is a *set* (unordered, no duplicate event labels) of n event labels in ξ. Episodes group labels that occur together, thus highlighting the relationships between the events. They constitute an abstraction to characterize the activities occurring in the house without prior or expert knowledge. For example, while the cooking activity cannot be recognized directly, we can discover the frequently reoccurring episode composed of motion sensor activations in the kitchen and the use of cooking appliances. In the coming definitions, E refers to an episode with n event labels $\{e_1, e_2, \dots e_n\}$.

Our goal is to discover regular episodes in the recent past, and update this knowledge when new events occur. We thus consider a sliding window model, where a window W is composed of m consecutive batches, i.e. $W = \langle B_i, B_{i+1}, \dots B_{i+m-1} \rangle$, where each batch B_i is a sequence of events. When a new batch of data B_{i+m} arrives, B_i becomes outdated and is removed from W. The batches span over a user-specified time unit (such as one day, one week, etc.).

Definition 1 (Occurrence of episode E). There is an *occurrence* o of E between times t_1 and t_n if there exists a permutation p of $(1, \dots n)$ and n timestamps $(t_1, \dots t_n)$ such that $o = \langle (e_{p(1)}, t_1), \dots (e_{p(n)}, t_n) \rangle$ is a subsequence of the window W. o is said to be a T_{ep}-occurrence if $t_n - t_1 < T_{ep}$, which we use as a constraint: only the T_{ep}-occurrences are considered in the regularity measures.

Definition 2 (Minimal occurrence of episode E). Let o be an occurrence of E, spanning from t_1 to t_n. o is a *minimal occurrence* if E has no occurrence o', spanning from t'_1 to t'_n such that $t_1 \leq t'_1$, $t'_n \leq t_n$, and $t'_n - t'_1 < t_n - t_1$.

Definition 3 (Non-overlapping occurrences of episode E). Let o and o' be a minimal occurrences of E spanning respectively from t_1 to t_n and from t'_1 to t'_n. o and o' are *non-overlapping* if $\min(t_n, t'_n) < \max(t_1, t'_1)$. The list of the non-overlapping minimal T_{ep}-occurrences of E is noted NMO^E.

Definition 4 (Regularity of episode E). Let NMO^E be the ordered sequence of the non-overlapping minimal occurrences of E. The regularity r^E of episode E can be defined as the maximal value of the following cases:

- The regularity between the start time of the window (t_{sw}) and the start time of the first minimal occurrence in NMO^E (t_1): $r_{sw} = t_1 - t_{sw}$
- The regularity between each pair of consecutive occurrences in NMO^E o_u, spanning from t_1 to t_n, and o_{u+1}, spanning from t'_1 to t'_n: $r_u = t'_1 - t_n$
- The regularity between the last occurrence $o = \langle (e_{p(1)}, t_1), \dots (e_{p(n)}, t_n) \rangle$ in NMO^E and the last timestamp of the window t_{ew}: $r_{ew} = t_{ew} - t_n$

One can notice that an episode E is more regular than another episode E' if its regularity value is lower.

Definition 5 (Top-k regular episodes). Let us consider the list of episodes sorted by ascending regularity. An episode E is a top-k regular episode if there are no more than $k-1$ episodes having regularity values lower than that of E.

With the user-given set of parameters: a number of desired interesting episodes k, a batch duration, a number m 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 sliding over a sensor data stream DS.

3. Literature Review

The traditional transactional databases contain information on the relationships between the items (the itemsets). For example in a retail market basket dataset, each transaction contains the products (items) a customer bought. Any subset of the transaction is an itemset. In event databases, these relationships are not explicit, and need to be searched in the data. This process is called *episode discovery*, and was introduced by [14]. It was later enhanced to cover different support counting techniques, such as window-based support [13], minimal occurrences [12], or minimal non-overlapping occurrences [24], which we also consider here (see definitions 2 and 3). Computational enhancements have also been proposed, exploring different search strategies, such as breadth-first [23] and depth-first [20] searches; as well as different pruning techniques: closed episodes [23,20], episode length constraints and top-k most frequent patterns [21]. The interest of the episodes is systematically evaluated based on their support or frequency.

However, characteristics other than frequency can also characterize interesting episodes, especially in the context of human activity monitoring: [4] explain that routines take an important role in the life of an elderly person. Indeed, routines allow them to keep control over their environment and reduce anxiety. Periodicity and regularity appear thus as interesting interest measures in the context of human behavior monitoring. The periodicity of the episodes has already been investigated [17]. Introduced in [19], *regularity* focuses on the maximal gap between the transactions covering an itemset. The concept has been extensively studied and enhanced in the context of transactional databases [2,18,1], mostly coupled with frequency measures.

The sensors disseminated in the house generate *events streams*. [7] describes the characteristics of such data sources and the constraints they set on the processing algorithms. In particular, the algorithm should use a fix amount of main memory, scan the data only once, and adapt to concept drifts. The popular sliding window framework conforms to these constraints, and has been used for episode mining in the past [15,10].

Our proposition gathers in an unprecedented combination: parallel episode mining, using regularity measures over event streams. In order to do so, we propose an adaptation of the regularity definition to non-transactional temporal data. We also propose, describe and analyze an efficient algorithm, *TKRES*, for the discovery and update of the top-k most regular episodes.

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. *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 4.1), that is to say mining 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 4.2), updating 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.

4.1. Initial mining

As described in algorithm 1, *TKRES* 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 (lines 2–5). The k event labels with the lowest regularity are collected and ordered in the top-k list. The k^{th} least regular event label gives an upper bound to the maximal regularity an episode may have and still be of interest.

TKRES then builds the k-tree to generate longer episodes. It reduces its memory consumption by limiting the depth of the tree to dot. This threshold is defined as the smallest possible depth enabling to hold k episodes in the tree. It is computed based on the size of the event label alphabet ξ. For example, if the events take values among 5 labels and the value of k is 25, then $dot = 3$.

The episode construction step considers pairs of episodes in the k-tree (starting from depth 1 to dot). For each pair of episodes X and Y, a new entry of $Z = X \cup Y$ is created and set to be a child node of X. Based on the occurrence times of X and Y, the occurrence times of $X \cup Y$ are computed and added to the corresponding node in the k-tree. The corresponding regularity is computed and compared to the least good regularity in the top-k list. If need be, the top-k list is updated (lines 8–14).

There is no guarantee that all the top-k regular episodes are shorter than dot. The pairs of episodes at depth dot and higher having a better regularity than the k^{th} most-regular episode are combined and their union is investigated. If the new episode belongs in 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 15–21).

Algorithm 1 *TKRES*–initial mining

Input: k: number of episodes to be discovered,
m batches of sensor data $\langle B_1, \ldots B_m \rangle$
Output: top-k list containing in the top-k most regular episodes,
k-tree, containing the occurrence information for the episodes on $\langle B_1, \ldots B_m \rangle$

1: Initialize the tree, and create entries for all single events
2: **for each** batch B_i **do**
3: **for each** event (e_j, t_j) in B_i **do**
4: update e_j's occurrence information in the tree with timestamp 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 dot of the k-tree to be created
8: **for** depth $d = 1$ to $dot - 1$ **do**
9: **for each** episodes X and Y at depth d having $d - 1$ common labels **do**
10: Merge episodes X and Y to be Z (it contains thus $d + 1$ labels)
11: Create a node for Z in the k-tree, set to be a child of X
12: Get the occurrence times for Z from X and Y. Infer its regularity r^Z
13: **if** $r^Z < r^{k^{th}}$ **then**
14: Remove the k^{th} episode from the top-k list, insert Z
15: **for** depth $d = dot$ to $|\xi|$ **do**
16: **for each** pair of episodes X and Y at depth d with $d - 1$ common labels where $r^X \leq r^{k^{th}}$ and $r^Y \leq r^{k^{th}}$ **do**
17: Merge episodes X and Y to be Z
18: Get the occurrence times for Z from X and Y. Infer its regularity r^Z
19: **if** $r^Z < r^{k^{th}}$ **then**
20: Remove the k^{th} episode from the top-k list, insert Z
21: Create a node for Z in the k-tree, set to be a child of X on depth $d + 1$

4.2. 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 dot, and the entries of top-k regular episodes at depth higher than dot. However, when a new batch of sensor data arrives, the contents of the top-k might not be up to date anymore. Algorithm 2 details the steps for the maintenance of the data structures.

TKRES first removes all episodes from the top-k list, as well as the nodes in the k-tree which are in depth higher than dot. Moreover, occurrence information of the oldest batch B_i is also removed from all entries in the k-tree (lines 1–3). The events in the new batch of data B_{i+m} are investigated and the occurrence information on the single-label episodes is updated, as well as the regularity. The k most regular length-1 episodes are gathered in the top-k list (lines 4–7).

The longer episodes are generated thanks to a process similar to the one described in algorithm 1 and section 4.1: through the merging of pairs of episodes. For the episodes at depth lower than $dot - 1$, we simply need to update the occurrence information with what occurs during the new batch, the rest is already in the tree (lines 8–15). But for episodes in depth dot and higher, *TKRES* has to intersect all of occurrence information, since it was not saved (lines 16–22). After this merging and intersection process, we gain a complete set of top-k regular episodes contained in the top-k list and the k-tree for mining the next coming batch of sensors data.

Algorithm 2 *TKRES*–mining a new incoming batch of sensor data

Input: k: number of episodes to be discovered, B_{i+m}: the new batch,
k-tree, with the occurrence information for the episodes on $\langle B_i, \ldots B_{i+m-1} \rangle$
Output: top-k list containing in the top-k most regular episodes,
k-tree, with the occurrence information for the episodes on $\langle B_{i+1}, \ldots B_{i+m} \rangle$

1: Empty the top-k list
2: Remove all the nodes at depth higher than dot
3: For each node, remove the occurrence times occurring during B_i
4: **for each** event (e_j, t_j) in the new batch B_{i+m} **do**
5: Collect t_j in the node for e_j in the k-tree
6: Recompute the regularity for the episodes at depth 1
7: Collect the k labels with the lowest regularity into the sorted top-k list
8: **for** depth $d = 1$ to dot **do**
9: **for each** episodes X and Y at depth d having $d - 1$ common labels **do**
10: Merge episodes X and Y to be Z
11: Get the occurrence times for Z from X and Y (Only for what is occurring during B_{i+m}). Infer its regularity r^Z
12: If it is not already in the tree, create a node for Z, set to be a child of X
13: **if** $r^{X \cup Y} < r^{k^{th}}$ **then**
14: Remove the k^{th} episode from the top-k list, insert Z instead
15: **for** depth $d = dot$ to $|\xi|$ **do**
16: **for each** pair of episodes X and Y at depth d with $d - 1$ common labels where r^X and $r^Y \leq r^{k^{th}}$ **do**
17: Merge episodes X and Y to be Z
18: Get the occurrence times for Z from X and Y. Infer its regularity r^Z
19: **if** $r^Z < r^{k^{th}}$ **then**
20: Remove the k^{th} episode from the top-k list, insert Z instead
21: Create a node for Z in the k-tree, set to be a child of X on depth $d + 1$

5. Experimental study on real home activity monitoring datasets

In this section, we investigate the runtime and output of *TKRES* on two real-life datasets, *Twor2009* (#7) and *Aruba* (#17)), from the CASAS project[1 6], and analyzing the sensor recordings generated by the people living in the two houses.

The *Twor2009* dataset contains data from motion sensors, item sensors in the kitchen, door sensors, and water usage sensors. The *Aruba* dataset contains sensor data from a home where a volunteer adult lived, such as motion sensors, door closure sensors and temperature sensors. Table 1 summarizes the characteristics of the two datasets. Most sensors give a binary information (motion sensors are either ON or OFF, the doors are either OPENed or CLOSEd), but some give numerical information (temperature readings for the *Aruba* dataset, water usage for the *Twor2009* dataset). We chose to leave these raw data points without preprocessing.

Table 1. Characteristics of the two datasets

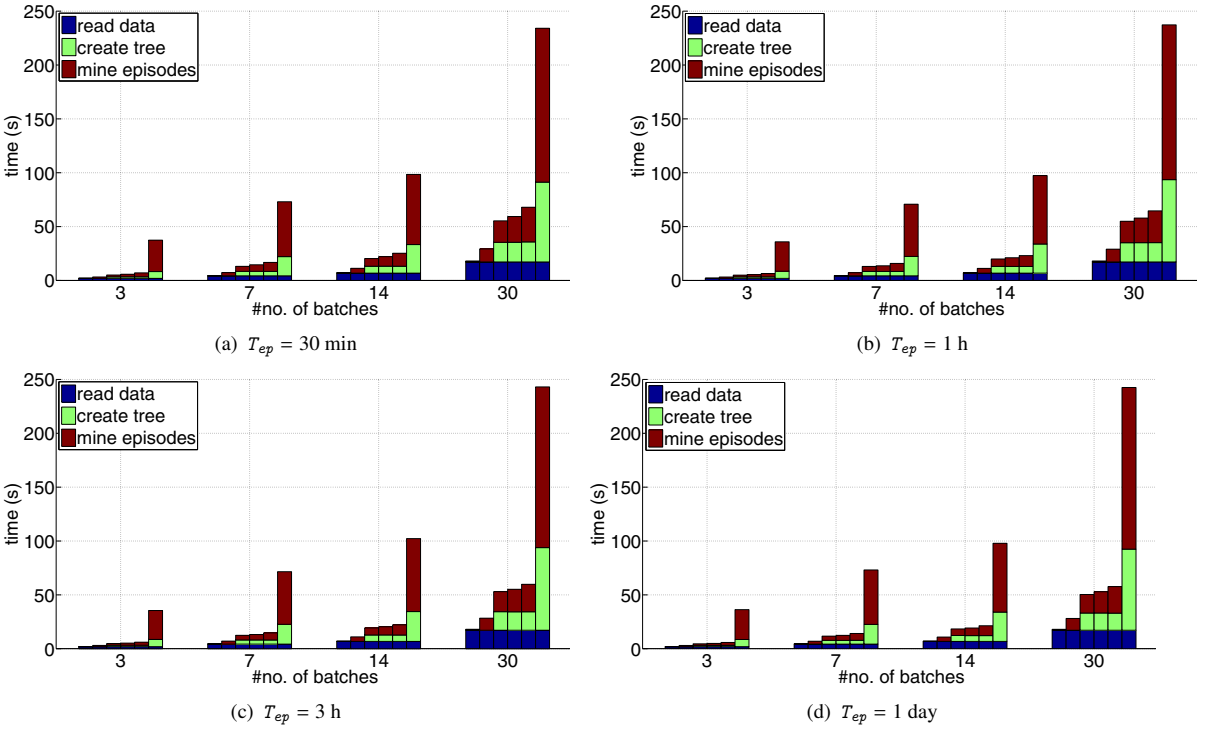
	Aruba	*Twor2009*
Start	2010-11-04	2009-02-02
End	2011-06-11	2009-04-04
Duration	7 months	2 months
# habitants	1	2
Sensors	Motion detectors, door sensors, temperature	Motion detectors, water usage, door sensors
# events	1 719 558 (1 602 985 without temperature data)	137 788 (130 097 without water events)
# labels	351 (157 without temperature data)	4 997 (135 without water events)

In order to assess the performance and scalability of *TKRES*, we ran experiments with different values for k (ranging from 10 to 5000), the size of the window (ranging from 3 to 30 batches), and the maximal duration T_{ep} of the occurrences (ranging from 30 minutes to 1 day). Figures 1 and 2 show the runtime for the initial mining on the two datasets. Each subplot presents the time performance of *TKRES* for a value of T_{ep}, and contains 4 groups (one per window length) of 6 bars (one per value of k, which takes successively the values 10, 50, 100, 500, 1000, 5000). The total execution time in each bar is split between the three mining subtasks: reading the data, creating the tree up to depth *dot*, and mining the results. One notices in particular that the runtime increases with k: since more results are requested, the k-tree is bigger and *TKRES* spends more time building it and mining the results. In addition, runtime increases as the size of windows increases: there are more occurrences to process.

Figures 3 and 4 illustrate the average runtime for mining new coming batch sensor stream. The runtime for each experiment is split the four mining subtasks: (i) removing occurrence information of the old outdated batch, (ii) reading sensors data from the new batch, (iii) updating tree up to depth *dot* and (iv) mining the other episodes. In particular, we can observe that the time for the removal of old information, the reading of new data and the update of the tree are small compared to the time needed for mining the results. This shows the interest of storing part of the tree and updating it, at least up to depth *dot*. After that, the results are computed again each time the window changes, hence the longer runtime for this mining subtask. However if the whole tree was just stored and updated, it is the memory usage that would be heavy. *dot* allows a trade-off between memory and speed.

Both the *Twor2009* and *Aruba* datasets, the most regular episodes are related to temperature events. The reason for that is that these sensors are programmed to trigger regular measures. Though the performance results presented in figures 1–4 use the complete event dataset, we also tested *TKRES* on filtered datasets, containing only sensors triggered by human activity. Table 2 present the top-10 regular episodes for the *Aruba* dataset, with $m = 3$ and $T_{ep} = 30$ min. The map of the apartment is available with the dataset, but was not included here for readability matters. Sensors

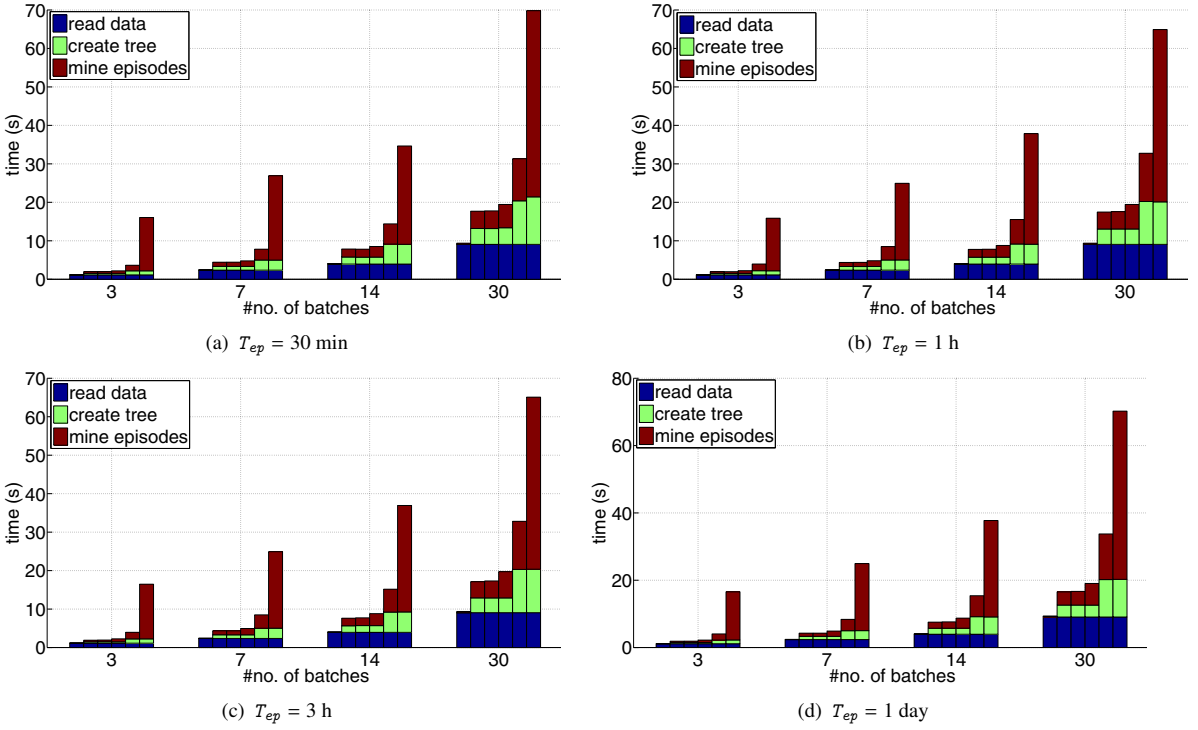
[1] http://ailab.wsu.edu/casas/datasets/



(a) T_{ep} = 30 min (b) T_{ep} = 1 h

(c) T_{ep} = 3 h (d) T_{ep} = 1 day

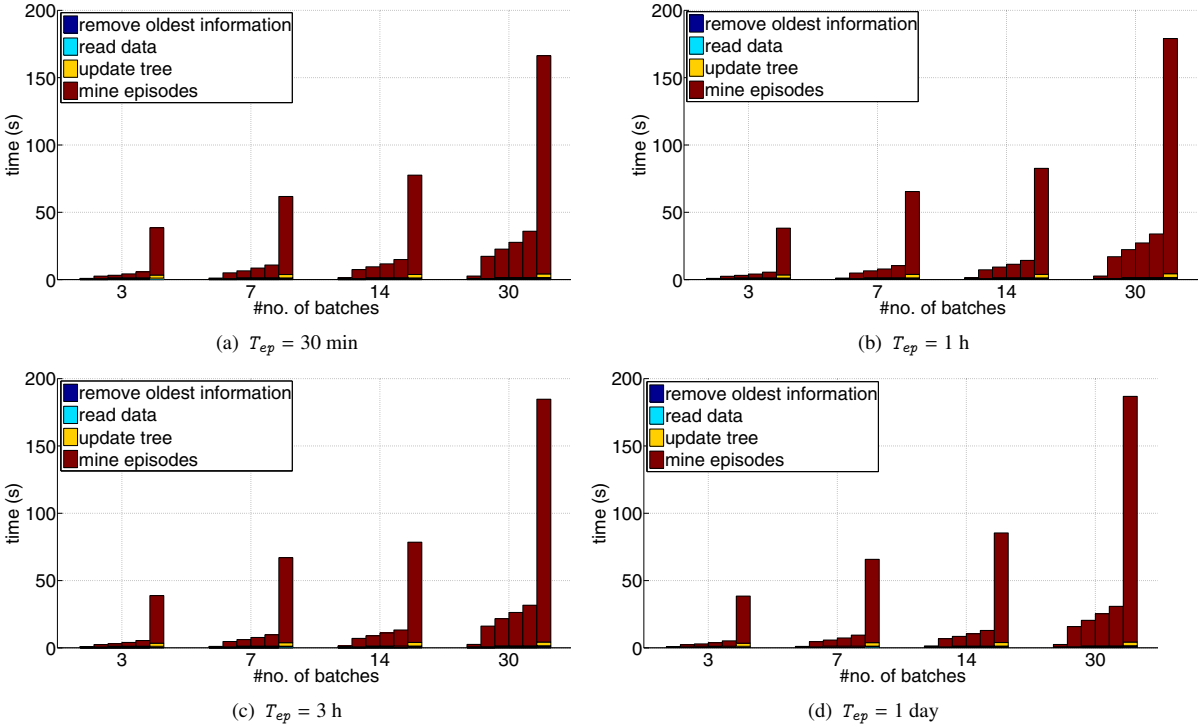
Fig. 1. Runtime of the initial mining on *Twor2009*



(a) T_{ep} = 30 min (b) T_{ep} = 1 h

(c) T_{ep} = 3 h (d) T_{ep} = 1 day

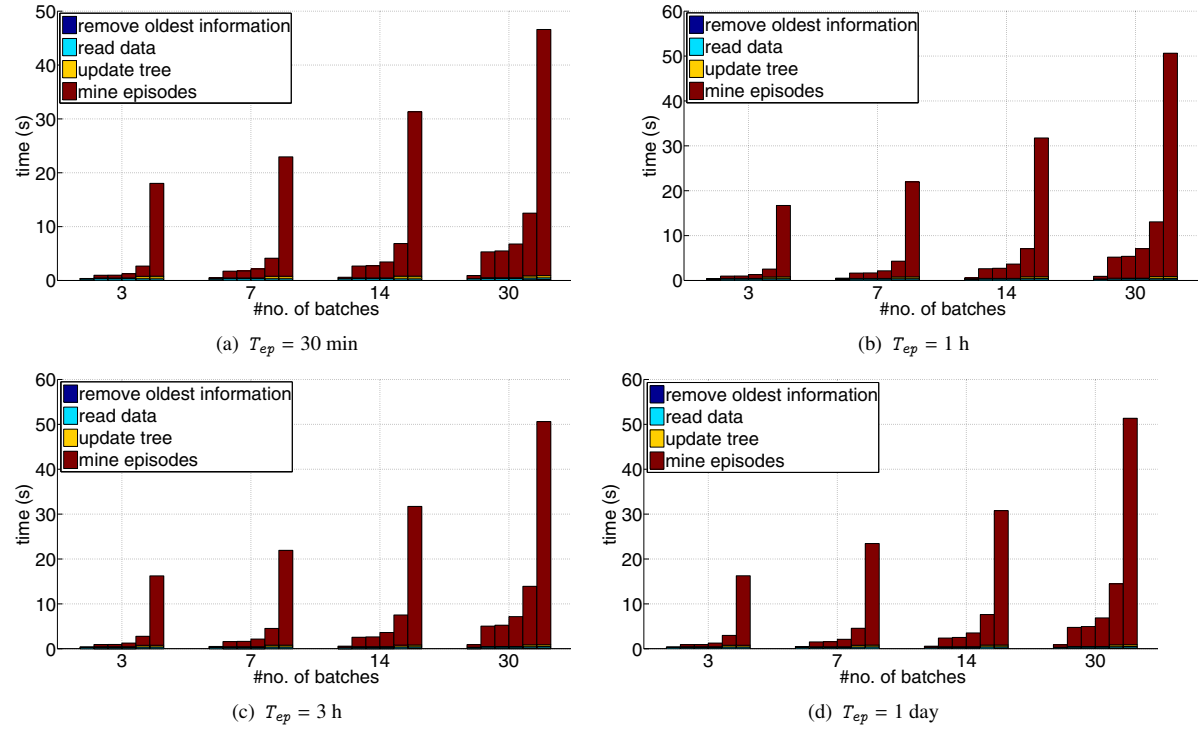
Fig. 2. Runtime of the initial mining on *Aruba*



(a) T_{ep} = 30 min (b) T_{ep} = 1 h

(c) T_{ep} = 3 h (d) T_{ep} = 1 day

Fig. 3. Runtime of mining update on *Twor2009*



(a) T_{ep} = 30 min (b) T_{ep} = 1 h

(c) T_{ep} = 3 h (d) T_{ep} = 1 day

Fig. 4. Runtime of mining update on *Aruba*

Table 2. Top-10 regular episodes on *Aruba* without temperature sensors

Top-10	Episode	Regularity
1	M002, M003	4.24 h
2	M002	4.24 h
3	M003, M007	5.67 h
4	M002, M003, M007	5.67 h
5	M002, M007	5.67 h
6	M007	5.68 h
7	M002, M005	5.68 h
8	M002, M004	5.68 h
9	M002, M003, M004	5.68 h
10	M003, M005, M007	5.68 h

M001–M007 are located in the main bedroom, with M004 recording movements from and towards the bathroom. The inhabitant thus seems to come very regularly to her bedroom. The most regular episodes show regular trajectories in the apartment, and could help the physicians improve the layout of the apartment, based on the behavior of the monitored person.

6. Conclusion

We address the problem of mining top-k regular episodes from sensors stream. The main objective is to push measures of regularity to the episode mining problem and to maintain computational time efficiency. Moreover, the top-k approach allows the user to control the number of desired output episodes. To discover such episodes, we present an efficient sing-pass algorithm, named *TKRES*, using a simple top-k list structure to collect the output and a k-tree structure to maintain the occurrence information of the episodes. We propose to make a trade-off between the computational time to update episode knowledge when data changes and memory usage, thanks to the setting of a depth boundary *dot* on the k-tree: the episodes shorter than *dot* are fully investigated and stored, the longer episodes are only if they belong to the top-k list. Experimental results on two 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, several interest measures, such as the length of episodes, their frequency or their periodicity could be combined with the regularity to better target interesting episodes. This will allow comparative studies against traditional approaches which mainly use one or two interest measures. It would also be interesting to investigate closed regular episodes.

References

1. Amphawan, K., 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., Chongsuphajaisiddhi, V. (Eds.), IAIT. Vol. 409 of Communications in Computer and Information Science. Springer, pp. 1–12.
2. Amphawan, K., Lenca, P., Surarerks, A., 2009. Mining top-K periodic-frequent pattern from transactional databases without support threshold. In: Papasratorn, B., Chutimaskul, W., Porkaew, K., Vanijja, V. (Eds.), IAIT. Vol. 55 of Communications in Computer and Information Science. Springer, pp. 18–29.
3. Avci, U., Passerini, A., 2013. A fully unsupervised approach to activity discovery. In: Salah, A. A., Hung, H., Aran, O., Gunes, H. (Eds.), Human Behavior Understanding Workshop. Vol. 8212 of Lecture Notes in Computer Science. Springer, pp. 77–88.
4. Bergua, V., Bouisson, J., Dartigues, J.-F., Swendsen, J., Fabrigoule, C., Prs, K., 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.
5. Cleland, I., Han, M., Nugent, C. D., Lee, H., Zhang, S., McClean, S. I., Lee, S., 2013. Mobile based prompted labeling of large scale activity data. In: Nugent, C. D., Coronato, A., Bravo, J. (Eds.), IWAAL. Vol. 8277 of Lecture Notes in Computer Science. Springer, pp. 9–17.
6. Cook, D. J., Schmitter-Edgecombe, M., 2009. Assessing the quality of activities in a smart environment. Methods of information in medicine 48 (5), 480.

7. Gama, J., 2013. Data stream mining: the bounded rationality. Informatica (Slovenia) 37 (1), 21–25.
8. Gu, T., Wu, Z., Tao, X., Pung, H. K., Lu, J., 2009. epsicar: An emerging patterns based approach to sequential, interleaved and concurrent activity recognition. In: PerCom. IEEE Computer Society, pp. 1–9.
9. Jakkula, V. R., Crandall, A. S., Cook, D. J., 2009. Enhancing anomaly detection using temporal pattern discovery. In: Kameas, A. D., Callagan, V., Hagras, H., Weber, M., Minker, W. (Eds.), Advanced Intelligent Environments. Springer US, pp. 175–194.
10. Lin, S., Qiao, J., Wang, Y., 2014. Frequent episode mining within the latest time windows over event streams. Appl. Intell. 40 (1), 13–28.
11. Lustrek, M., Kaluza, B., 2009. Fall detection and activity recognition with machine learning. Informatica (Slovenia) 33 (2), 197–204.
12. Mannila, H., Toivonen, H., 1996. Discovering generalized episodes using minimal occurrences. In: Simoudis, E., Han, J., Fayyad, U. M. (Eds.), KDD. AAAI Press, pp. 146–151.
13. Mannila, H., Toivonen, H., Inkeri Verkamo, A., 1997. Discovery of frequent episodes in event sequences. Data Min. and Knowl. Discov. 1 (3), 259–289.
14. Mannila, H., Toivonen, H., Verkamo, A. I., 1995. Discovering frequent episodes in sequences. In: Fayyad, U. M., Uthurusamy, R. (Eds.), KDD. AAAI Press, pp. 210–215.
15. Rashidi, P., Cook, D. J., 2010. Mining sensor streams for discovering human activity patterns over time. In: Webb, G. I., Liu, B., Zhang, C., Gunopulos, D., Wu, X. (Eds.), IEEE ICDM. pp. 431–440.
16. Rashidi, P., Mihailidis, A., 2013. A survey on ambient assisted living tools for older adults. IEEE Journal of Biomedical and Health Informatics 17 (3), 579–590.
17. Soulas, J., Lenca, P., Thépaut, A., 2013. Monitoring the habits of elderly people through data mining from home automation devices data. In: Correia, L., Reis, L. P., Cascalho, J. (Eds.), EPIA. Vol. 8154 of Lecture Notes in Computer Science. Springer, pp. 343–354.
18. Tanbeer, S. K., Ahmed, C. F., Jeong, B., 2010. Mining regular patterns in data streams. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (Eds.), DASFAA. Vol. 5981 of Lecture Notes in Computer Science. Springer, pp. 399–413.
19. Tanbeer, S. K., Ahmed, C. F., Jeong, B., Lee, Y., 2009. Discovering periodic-frequent patterns in transactional databases. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T. B. (Eds.), PAKDD. Vol. 5476 of Lecture Notes in Computer Science. Springer, pp. 242–253.
20. Tatti, N., Cule, B., 2011. Mining closed episodes with simultaneous events. In: Apté, C., Ghosh, J., Smyth, P. (Eds.), KDD. ACM, pp. 1172–1180.
21. Tzvetkov, P., Yan, X., Han, J., 2003. TSP: mining top-k closed sequential patterns. In: IEEE ICDM. IEEE Computer Society, pp. 347–354.
22. van Kasteren, T., Englebienne, G., Krse, B., 2010. An activity monitoring system for elderly care using generative and discriminative models. Personal and Ubiquitous Computing 14, 489–498.
23. Zhou, W., Liu, H., Cheng, H., 2010. Mining closed episodes from event sequences efficiently. In: Zaki, M. J., Yu, J. X., Ravindran, B., Pudi, V. (Eds.), PAKDD. Vol. 6118 of Lecture Notes in Computer Science. Springer, pp. 310–318.
24. Zhu, H., Wang, P., He, X., Li, Y., Wang, W., Shi, B., 2010. Efficient episode mining with minimal and non-overlapping occurrences. In: IEEE ICDM. pp. 1211–1216.