Quantitative Episode Trees*

Mirco Nanni[1] and Christophe Rigotti[1,2]

[1]KDD Laboratory, University of Pisa and ISTI-CNR Pisa, Italy
[2]INSA-LIRIS UMR 5205 CNRS, Lyon, France

Abstract. Among the family of the local patterns, episodes are commonly used when mining a single or multiple sequences of discrete events. An episode reflects a qualitative relation *is-followed-by* over event types, and the refinement of episodes to incorporate quantitative temporal information is still an on going research, with many application opportunities. In this paper, focusing on serial episodes, we design such a refinement called *quantitative episodes* and give a corresponding extraction algorithm. The three most salient features of these quantitative episodes are: (1) their ability to characterize main groups of homogeneous behaviors among the occurrences, according to the duration of the *is-followed-by* steps, and providing quantitative bounds of these durations organized in a tree structure; (2) the possibility to extract them in a complete way; and (3) to perform such extractions at the cost of a limited overhead with respect to the extraction of standard episodes.

1 Introduction

Sequential data is a common form of information available in several application contexts, thus naturally inducing a strong interest for them among data analysts. A decade-long attention has been paid by researchers in data mining to study forms of patterns appropriated to this kind of data, such as sequential patterns [1] and episodes [7]. In particular, in this paper we will focus on *serial episodes*, that are sequences of event types extracted from single or multiple input sequences, and that reflect a qualitative relation *is-followed-by* between the event types.

Episodes have natural applications into several domains, including for instance the analysis of business time series [2], medical data [8], geophysical data [9] and also alarm log analysis for network monitoring (especially in telecommunications) [5]. However, in many applications episodes clearly show some limitations, due to the fact that the information provided by the *is-followed-by* relation is not always enough to properly characterize the phenomena at hand. This, in particular, pulls our research toward the refinement of episodes to incorporate quantitative temporal information, able to describe the time intervals observed for the *is-followed-by* relation.

* This research is partly funded by EU contracts IQ IST-FP6-516169, and GeoPKDD IST-FP6-014915.

In this paper, we propose a refinement of episodes called *quantitative episodes*, that provides quantitative temporal information in a readable, tree-based graphically representable form. These quantitative episodes describe the main groups of homogeneous behaviors within the occurrences of each episode, according to the elapsed times between the consecutive event types of the episode. Moreover, they are not provided in an isolated way, but in trees giving a global view of how the occurrences of the corresponding episode differentiate in homogeneous groups along the elements of the pattern. From a computational point of view, the main interest of the quantitative episodes is that they can be mined in a sound and complete way without increasing the cost of extractions significantly when compared to extractions of episodes alone. This is achieved through an extraction algorithm that tightly integrates episode extraction with a computationally reasonable analysis of temporal quantitative information.

This paper is organized as follows: in Section 2 some preliminary definitions needed concerning episodes are recalled from the literature; Section 3, then, introduces quantitative episodes; Section 4 presents the principle of an algorithm for efficiently extracting quantitative episodes, which is evaluated experimentally in Section 5; finally, in Section 6 we briefly review the related literature and conclude with a summary in Section 7.

2 Preliminary definitions

We briefly introduce standard notions [7], or give equivalent definitions when more appropriated to our presentation.

Definition 1. ***(event, event sequence,operator*** $\sqsubseteq$***)*** Let E be a set of *event types* and $\prec$ a total order on E. An *event* is a pair denoted (e, t) where $e \in E$ and $t \in \mathbb{N}$. The value t denotes the time stamp at which the event occurs. An *event sequence* S is a tuple of events $S = \langle (e_1, t_1), (e_2, t_2), \ldots, (e_l, t_l) \rangle$ such that $\forall i \in \{1, \ldots, l-1\}$, $t_i < t_{i+1} \vee$ $(t_i = t_{i+1} \wedge e_i \prec e_{i+1})$. Given two sequences of events S and S', S' is a *subsequence* of S, denoted $S' \sqsubseteq S$, if S' is equal to S or if S' can be obtained by removing some elements in S.

Definition 2. ***(episode, occurrence, minimal occurrence, support)*** An *episode* is a non empty tuple α of the form $\alpha = \langle e_1, e_2, \ldots, e_k \rangle$ with $e_i \in E$ for all $i \in \{1, \ldots, k\}$. In this paper, we will use the notation $e_1 \rightarrow e_2 \rightarrow \ldots \rightarrow e_k$ to denote the episode $\langle e_1, e_2, \ldots, e_k \rangle$ where '$\rightarrow$' may be read as 'is followed by'. The *size* of α is denoted $|\alpha|$ and is equal to the number of elements of the tuple α, i.e., $|\alpha| = k$. The *prefix* of α is the episode $\langle e_1, e_2, \ldots, e_{k-1} \rangle$. We denote it as *prefix*(α). An episode $\alpha = \langle e_1, e_2, \ldots, e_k \rangle$ *occurs* in an event sequence S if there exists at least one sequence of events $S' = \langle (e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k) \rangle$ such that $\forall i \in \{1, \ldots, k-1\}, t_i < t_{i+1}$ and $S' \sqsubseteq S$. The pair $\langle t_1, t_k \rangle$ is called an *occurrence* of α in S. Moreover, if there is no other occurrence $\langle t'_1, t'_k \rangle$ such that $[t'_1, t'_k] \subset [t_1, t_k]$, then the pair $\langle t_1, t_k \rangle$ is called a *minimal occurrence* of α. The support of α in S, denoted *support*(α, S), is the number of minimal occurrences of α in S.

Intuitively, a minimal occurrence is simply an occurrence that does not strictly contain another occurrence of the same episode. These episodes and their occurrences correspond to the *serial* episodes of [7]. For instance, let $S = \langle (a, 0), (b, 1), (c, 1), (b, 2)\rangle$ be an event sequence and $\alpha = a \rightarrow b$ be an episode. Then, α has two occurrences in S: $\langle 0, 1\rangle$ and $\langle 0, 2\rangle$. The former is a minimal occurrence, while the latter is not, since $[0, 1] \subset [0, 2]$. Notice that there is no occurrence of episode $\alpha' = b \rightarrow c$.

These definitions, and the ones introduced in the rest of the paper, are given for a single sequence S, but they extend trivially to multiple sequences. In that case the support is the sum of the number of occurrences in all sequences.

3 Quantitative episodes

3.1 Informal presentation

The idea of quantitative episodes essentially consists in dividing the set of occurrences of an episode into homogeneous, significantly populated groups. Homogeneity, in particular, is obtained when on each step, made of two consecutive elements of the episode, the occurrences in the same group show similar transition times (i.e., similar times elapsed between an element and the next one within the episode). The result can be graphically summarized through a tree-like structure, as the one depicted in Figure 1 that represents homogeneous groups of occurrences of an episode $\alpha = A \rightarrow B \rightarrow C \rightarrow D$. The figure can be read in the following way:

- The episode has 1000 occurrences in the sequence of events, and this value is written under the first event of the episode.
- Among these 1000 occurrences, there are 2 subgroups that show homogeneous duration for step $A \rightarrow B$: one (the upper branch of the split) corresponds to transition times between 2 and 10, and covers 500 occurrences; the other (lower branch) corresponds to transition times in interval $[15, 20]$ and covers 400 occurrences. Notice that 100 occurrences of $A \rightarrow B \rightarrow C \rightarrow D$ are lost, meaning that they exhibit a rather isolated duration for step $A \rightarrow B$ and cannot be associated with other occurrences to form a significantly populated group.
- In the largest group obtained above, all occurrences present similar step durations for steps $B \rightarrow C$ and $C \rightarrow D$, and are kept together in a single group. The other group, containing 400 occurrences, is split further into homogeneous groups w.r.t. duration of step $B \rightarrow C$. Notice that the resulting homogeneous groups overlap, sharing a subset of occurrences and resulting in non-disjoint time intervals. Indeed, we can observe that the total count of occurrences in the two groups (205+202) is greater than the original total amount (400), since some occurrences are counted twice.

– One of these two groups is further split into two (disjoint) groups while the other is not.
– Each path from the root to a leaf in the tree corresponds to a group of occurrences that shows an homogeneous behavior along all the steps of the episode, and covers a sufficient number of occurrences (in this example, at least 90). This homogeneous behavior can be represented by the sequence of time intervals on the path, and can be added to the episode as a *quantitative* feature to form a *main grouping quantitative episode*. The tree in Figure 1 depicts four such patterns (one for each path from the root to a leaf). The tree relates these patterns together, showing how the occurrences can be differentiated into groups along the steps of the episode.

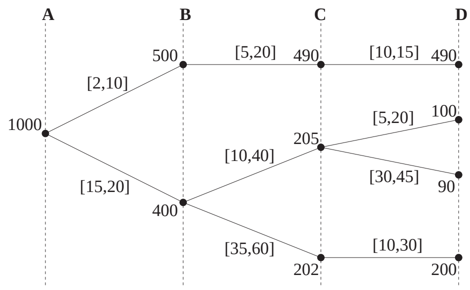

Fig. 1. Tree of quantitative episodes for episode $\alpha = A \rightarrow B \rightarrow C \rightarrow D$.

3.2 Quantitative episode definition

Definition 3. ***(quantitative episode)*** A *quantitative episode* (*q-episode*) is a pair $P = \langle \alpha, IT \rangle$ where α is an episode of size $k > 1$, and $IT = \langle it_1, \ldots, it_{k-1} \rangle$, with $\forall i \in \{1, \ldots, k-1\}, it_i = [a_i, b_i] \subset \mathbb{N}^+$ (i.e., it_i is an interval in $\mathbb{N}^+$). The size of P, denoted $|P|$ is defined as $|P| = |\alpha|$.

The it_i intervals are intended to represent values of elapsed time between the occurrences of two consecutive event types of the episode α. For instance $\langle A \rightarrow B \rightarrow C \rightarrow D, \langle [15, 20], [10, 40], [5, 20] \rangle \rangle$ is one of the q-episodes depicted in Figure 1.

To handle the time stamps of the events corresponding to all event types within an episode the definition of occurrence needs to be modified as follows.

Definition 4. ***(occurrence)*** An *occurrence* of an episode $\alpha = \langle e_1, e_2, \ldots, e_k \rangle$ in an event sequence S is a tuple $\langle t_1, t_2, \ldots, t_k \rangle$ such that there exists $S' = \langle (e_1, t_1), (e_2, t_2), \ldots, (e_k, t_k) \rangle$ satisfying $\forall i \in \{1, \ldots, k-1\}, t_i < t_{i+1}$ and $S' \sqsubseteq S$.

Notice that subsequence S' in the definition above can be formed by non-contiguous elements of sequence S. Using this definition of occurrence, the notion of minimal occurrence can be redefined accordingly.

Definition 5. ***(minimal occurrence)*** An occurrence $\langle t_1, \ldots, t_k \rangle$ of an episode α in event sequence S is a *minimal occurrence* if (1) there is no other occurrence $\langle t'_1, \ldots, t'_k \rangle$ of α such that $[t'_1, t'_k] \subset [t_1, t_k]$, and (2) if $k > 2$ then $\langle t_1, \ldots, t_{k-1} \rangle$ is a minimal occurrence of *prefix*(α).

As we will consider only minimal occurrences of episodes, we will simply use the term *occurrence*, when there is no ambiguity.

For a step $e_i \to e_{i+1}$ in an episode α, and its durations among a set of occurrences of α, now we define how these duration values are grouped. Informally, groups correspond to maximal sets of duration values that form *dense* intervals, where dense means that any sub-interval of significant size w_s contains a significant number of values n_s. More precisely, $w_s \in \mathbb{R}, w_s \geq 1$ and $n_s \in \mathbb{N}^+$ are termed the *density parameters* and characterize the groups in the following definition.

Definition 6. ***(occurrence groups)*** *Let $\mathcal{O}$ be a set of occurrences of episode α and i be an integer parameter such that $1 \leq i < |\alpha|$ (i identifies a step $e_i \to e_{i+1}$). Let $\Delta_i(x) = t_{i+1} - t_i$ for any occurrence $x = \langle t_1, \ldots, t_{|\alpha|} \rangle$ (i.e., the duration of step $e_i \to e_{i+1}$ for occurrence x). Then, the* occurrence groups *of $\mathcal{O}$* at level *i, denoted as group$(\mathcal{O}, i)$, are defined as follows:*

$$\begin{aligned} group(\mathcal{O}, i) = \{\ g \mid\ & g \text{ is a maximal subset of } \mathcal{O} \text{ s.t.:} \\ & \forall a, b \in [\min_{x \in g} \Delta_i(x), \max_{x \in g} \Delta_i(x)], \\ & \quad b - a \geq w_s \ \Rightarrow\ |\{x \in g \mid \Delta_i(x) \in [a, b]\}| \geq n_s\} \end{aligned}$$

For example, consider the set of occurrences $\mathcal{O} = \{x_1, \ldots, x_8\}$ having the respective durations 3,4,6,6,9,15,16,16 for step $e_i \to e_{i+1}$ (i.e., the values of Δ_i). Let the density parameters be $w_s = 3$ and $n_s = 2$ (i.e., at least two elements in any sub-interval of size 3). Then $group(\mathcal{O}, i) = \{\{x_1, \ldots, x_5\}, \{x_6, x_7, x_8\}\}$ (corresponding respectively to the durations $3, 4, 6, 6, 9$ and $15, 16, 16$).

The next definition specifies the tree structure of the occurrence groups.

Definition 7. ***(occurrence group tree)*** *Let $\mathcal{O}$ be the set of occurrences of episode α. Then, the* occurrence group tree *(*group tree *for short) of α is a rooted tree with labelled edges such that:*

- *the tree has $|\alpha|$ levels, numbered from 1 (the root) to $|\alpha|$ (the deepest leaves);*
- *each node v is associated with a set $v.g$ of occurrences of α;*
- *the root is associated with $root.g = \mathcal{O}$, i.e., with all the occurrences of α;*
- *if a node v at level i, $1 \leq i < |\alpha|$, is such that $group(v.g, i) = \{g_1, \ldots, g_k\}$, then it has k children $v_1, \ldots, v_k$, with $v_j.g = g_j, i \in \{1, \ldots, k\}$.*
- *each edge connecting node v at level i with its child v_j is labelled with the interval $[\min_{x \in v_j.g} \Delta_i(x), \max_{x \in v_j.g} \Delta_i(x)]$;*

Notice that such tree is unique, up to permutations in the order of the children of each node. Then, the main grouping q-episodes correspond simply to the sets of occurrences that have not been separated from the root to a leaf and that have a significant size.

Definition 8. ***(main grouping q-episode)*** *A q-episode $P = \langle \alpha, IT \rangle$ is said to be* a main grouping q-episode *if the group tree of α contains a path from the root to a leaf v such that:*

- *the labels of the edges met along the path correspond to the intervals in IT;*
- *and $|v.g|$, called the support of P, is greater or equal to σ_g, a user defined minimum group size.*

For instance, Figure 1 depicts a tree of main grouping q-episodes for $\alpha = A \rightarrow B \rightarrow C \rightarrow D$ and $\sigma_g = 90$ (a group tree restricted to paths forming main grouping q-episodes).

Since a minimal occurrence of α can be obtained only by extending a minimal occurrence of *prefix*(α), we have the following simple property that is used as a safe pruning criterion in the extraction principle.

Theorem 1. *Let α be an episode such that $|\alpha| > 1$. If there exists a main grouping q-episode $\langle \alpha, IT \rangle$, then there exists a main grouping q-episode $\langle prefix(\alpha), IT' \rangle$.*

4 Extracting q-episodes

In this section we present the principles of an algorithm called *Q-epiMiner* to find all main grouping q-episodes. It interleaves frequent episode extraction and group tree computation in a tight and efficient way. A more detailed presentation of the algorithm is given in the report [10].

Let $\alpha = \langle e_1, \ldots, e_n \rangle$ be an episode. For each event type e_i in α, $i > 1$, we consider a list D_i that collects the durations between e_{i-1} and e_i, i.e., the values $\Delta_{i-1}(x)$ for all occurrences x of α, and we suppose that each D_i is sorted by increasing duration value. By convention, for the sake of uniformity, D_1 contains a duration of 0 for all occurrences (there is no element before e_1).

In the following, we describe how these lists $D_1, \ldots, D_n$ can be used to compute the group tree of pattern α, and then how they can be updated when expanding α with an event type e_{n+1}.

Splitting one node. Splitting the group of occurrences of α associated to one node of the tree at level i (to obtain its children at level $i + 1$) can be done simply by a single scan of the elements in the group if these elements are ordered by the duration between e_i and e_{i+1}. We use a function named *splitGroup* performing this simple treatment. We suppose that it takes as input a list of occurrences in a group, sorted by duration of $e_i \rightarrow e_{i+1}$, and gives as output a collection of all maximal sublists satisfying the density criterion.

Computing the whole tree. Suppose that we have already computed the groups of occurrences denoted $g_1, \ldots, g_k$ that are associated respectively to the nodes $v_1, \ldots, v_k$ of a level i of the tree. These groups are split in the following way to obtain the nodes of the next level. Firstly, we create for each node v_j an empty list denoted $v_j.sortedGroup$. Then we scan D_{i+1} from first to last element, and for each occurrence found in D_{i+1} if the occurrence is in a group g_j then we insert the occurrence at the end of $v_j.sortedGroup$. Now, we have at hand for each v_j its group of occurrences sorted by increasing duration between e_i and e_{i+1}. Then, we can apply on each $v_j.sortedGroup$ the *splitGroup* function to compute the children of v_j and their associated groups of occurrences and thus obtain the next level of the group tree. Repeating this process allows to build the group tree in a levelwise way, taking advantage of the sorted lists $D_1, \ldots, D_n$. In the following, we assume that such a tree is computed by a function $computeTree$, applied on a tuple $\langle D_1, \ldots, D_n \rangle$.

Obtaining the information needed to compute the tree. The other key operation is the efficient computation of the sorted lists $D'_1, \ldots, D'_n, D'_{n+1}$ of a pattern $\alpha \rightarrow e$. Suppose that we know the list L_e of occurrences of $\alpha \rightarrow e$, and the sorted lists $D_1, \ldots, D_n$ of occurrences of α. Then, the main property used is that $D'_1, \ldots, D'_n$ are sublists of, respectively, $D_1, \ldots, D_n$, since each occurrence of $\alpha \rightarrow e$ comes from the expansion of an occurrence of α. So a list D'_i can be obtained simply by scanning D_i from the first to the last element and picking (in order) the occurrences in D_i that have been extended to form an occurrence of $\alpha \rightarrow e$. The result is a list D'_i sorted by increasing duration between e_{i-1} and e_i. The case of the list D'_{n+1} is different since it contains the same occurrences as L_e, so D'_{n+1} is simply a copy of L_e, but has to be sorted by increasing duration (between e_n and e_{n+1}). Having at hand the sorted lists $D'_1, \ldots, D'_n, D'_{n+1}$ we can then compute the group tree of $\alpha \rightarrow e$ by calling $computeTree(\langle D'_1, \ldots, D'_n, D'_{n+1} \rangle)$.

Integration with the extraction of episodes. One remaining problem to be solved is to build the occurrence list of the episode under consideration (as the list L_e for $\alpha \rightarrow e$). Fortunately, several approaches to extract episodes, or closely related patterns like *sequential patterns*, are based on the use of such occurrence lists (e.g., [7, 9, 13]), providing the information needed to update the duration lists D_i. Due to space limitation we will not detail this principle here. The basic idea is that if we store in a list L the locations (positions in the data sequence) of the occurrences of a pattern α, then for an event type e, we can use[1] L to build the list L_e of occurrences of $\alpha \rightarrow e$. In our case, for the occurrences of an episode $\alpha = \langle e_1, \ldots, e_n \rangle$ the location information stored in L are simply the time stamps of the last element e_n of α, sorted by increasing value. We use a function *expand* that takes the input sequence S and L, and that returns a set $\mathcal{L}_{exp}$ of tuples $\langle e, L_e \rangle$. The set $\mathcal{L}_{exp}$ contains for each event type e, the list L_e of locations of occurrences of $\alpha \rightarrow e$. As for L, the location information in L_e are the time stamps of the last element of $\alpha \rightarrow e$ and L_e is sorted by increasing location value.

The overall enumeration strategy of the episodes used is a standard depth-first prefix-based strategy because it fits both with the episode extraction and

[1] Together with other information, like the data sequence itself, or the location of the occurrences of e.

with the use of the sorted lists D_i to derive the sorted lists D'_i to compute the group trees. The strategy can simply be sketched as follows: when an episode α is considered we use it as a *prefix* to expand it and to obtain new episodes of the form $\alpha \rightarrow e$, and then, one after the other, we consider and expand each of these $\alpha \rightarrow e$.

Pruning strategy and correctness. Consider an episode α such that all leaves at level $|\alpha|$ of its group tree are associated to groups of size strictly less than σ_g (α has no corresponding main grouping q-episode, but α itself can have a support greater or equal to σ_g). By Theorem 1, we can also safely avoid the expansion of α, since this expansion cannot correspond to any main grouping q-episode. The exhaustive enumeration strategy of the episodes and the safety of the pruning strategy ensure the correctness of the general extraction principle.

5 Experiments

In this section we present the results of a set of experiments mainly aimed at studying how the size of the input data and the value of some input parameters impact on the performances of the *Q-epiMiner* algorithm described in this paper. The experiments presented are made on datasets containing several sequences. As mentioned previously, the definitions extended trivially to that case (the support is simply the sum of the support in all sequences). The only change in the abstract algorithm is that the occurrence locations are not simply time stamps, but sequence identifiers together with time stamps in the sequences. The algorithm was implemented in C, and all experiments were performed on a Intel Xeon 2Ghz processor with 1Gb of RAM over a Linux 2.6.14 platform.

5.1 Performance analysis on synthetic datasets

In order to collect large datasets having controlled characteristics, we randomly generated them by means of the Quest Synthetic Data Generator from IBM[2], by varying the number of input sequences generated (from 10K to 250K), the sequence length[3] (from 5 to 70) and the number of different event types used (from 5K to 20K). Where not specified hereafter, the following default parameter values were adopted: 100K input sequences, sequence length equal to 25, 5K event types, $w_s = 8$ and $n_s = 4$.

The curves in Figure 2(left) show the execution times of the prototype over datasets of increasing size and for three different numbers of event types used in the data. The σ_g parameter was set to 40 for 10K sequences and then was increased proportionally, up to 1000 for 250K sequences. As we can see, the execution time always grows almost linearly, having a higher slope when fewer event

[2] `http://www.almaden.ibm.com/software/projects/iis/hdb/Projects/data_mining/mining.shtml`

[3] The parameter of the generator controlling the number of events per time stamp was set to 1.

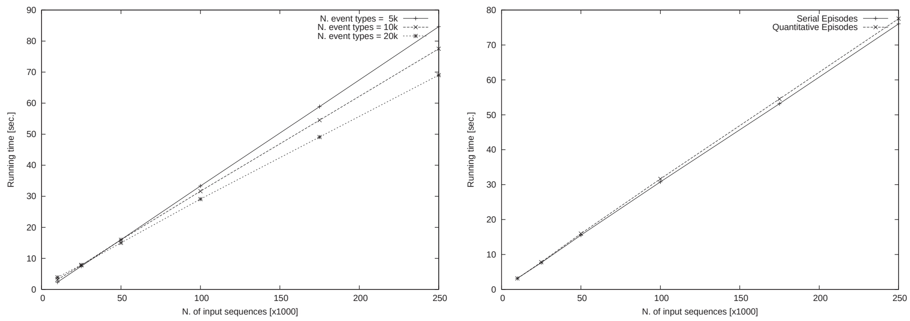

Fig. 2. Scalability w.r.t. number of input sequences.

types are in the data[4]. A similar scalability analysis is provided in Figure 2(right), where *Q-epiMiner* is compared against the extraction of serial episodes having at least a support of σ_g (this extraction is performed using the frequent episodes mining technique embedded in *Q-epiMiner*, without computing the durations, groups and trees, and implemented with the same low level optimizations). The values of σ_g were the same as in the previous experiment. The two curves are very close, meaning that the overhead introduced by the computation of main grouping q-episodes is well balanced by the pruning it allows. Finally, similar results are obtained by varying the length of the input sequences (see Figure 3(left)), where both curves have an apparently-quadratic growth (σ_g was set to 80 for length 5 and then was increased proportionally, up to 1120 for length 70). Obviously, for very long sequences usual episode constraints, like maximum window size, might be used [7].

Figure 3(right) reports the behaviour of the prototype when the minimum size of the groups is varied from 100 to 2000, and again its comparison to the mining of frequent serial episodes at minimum support σ_g. Here also, the two algorithms behave very similarly, this time showing a fast drop in the execution time as σ_g grows – as usual for frequent pattern mining algorithms. Due to space limits, we do not report the results obtained by varying the density parameters, that, however, seemed to have only a very limited impact on execution times of the algorithm on this kind of data.

5.2 Experiments on a real dataset

In this set of experiments we used real world data consisting of the July 2000 weblog from the web server of the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley[5]. In a preprocessing step, all non-HTML pages where removed and user sessions were extracted,

---

[4] Fewer event types with the same number of sequences leads to higher supports for the remaining event types and more frequent patterns of large size.

[5] http://www.cs.berkeley.edu/logs/http

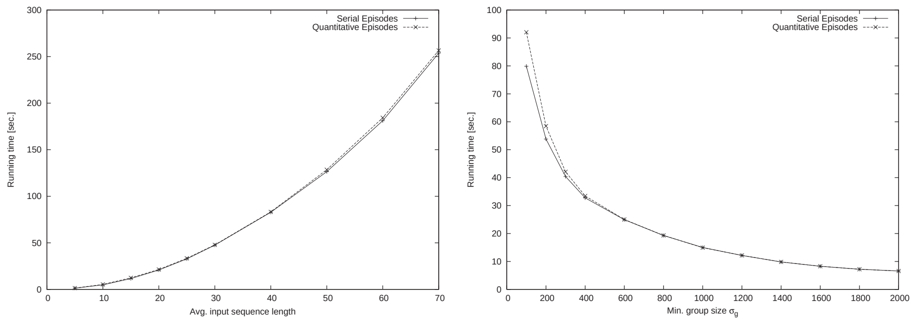

Fig. 3. Scalability w.r.t. input sequence length and min. group size σ_g, with 100K sequences.

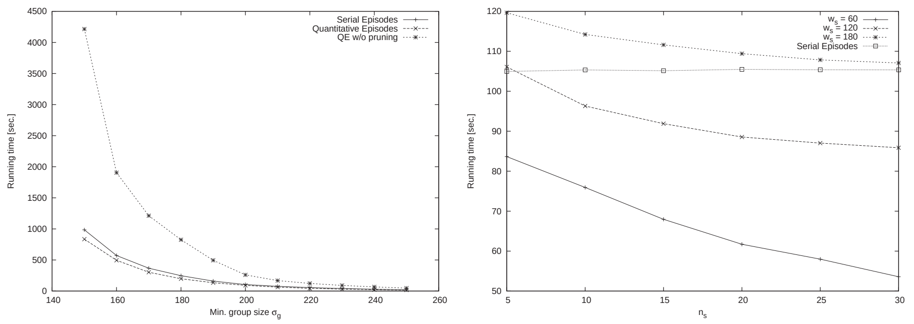

Fig. 4. Berkely dataset: Scalability w.r.t. σ_g and effects of the density parameters.

resulting in 90295 user sessions (used as input sequences) of average length of 13.0 with 72014 distinct pages.

In Figure 4 two graphs are plotted that describe the performances of the *Q-epiMiner* prototype on the Berkeley dataset for different minimum group sizes (graph on the left, with $w_s = 120$ and $n_s = 15$) and different density parameters (on the right, with $\sigma_g = 200$). The first plot confirms the results obtained on synthetic data, i.e., execution times drop very quickly as σ_g decreases. Moreover, an additional curve is plotted that represents a version of *Q-epiMiner* that does not apply any pruning based on the absence of a main grouping q-episode, but only applies a pruning based on the support of the episodes (an episode is not expanded only when its support is strictly less than σ_g). This curve shows the effectiveness of the full pruning made by *Q-epiMiner*. It should also be noticed that on this dataset, *Q-epiMiner* performs even better than the serial episode miner (with minimum support set to σ_g), confirming the fact that the pruning capabilities of the prototype are able to balance its potential overhead.

Finally, Figure 4(right) shows that, quite reasonably, the execution time decreases with larger minimum density parameter n_s (since they allow a stronger

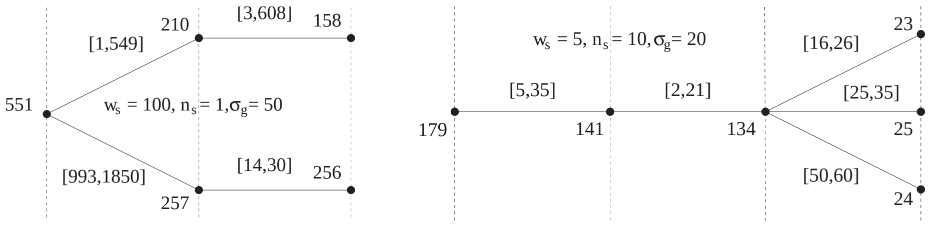

Fig. 5. Examples of trees of main grouping q-episodes.

pruning), and increases with larger window sizes w_s (which acts in the opposite direction).

We conclude this section by providing in Figure 5 two sample outputs obtained from the Berkeley dataset. In particular, we notice that the first tree contains two groups that split at the first step, showing well separated intervals of times ($[1, 549]$ against $[993, 1850]$). On the contrary, the second one contains three groups that split only at the third step, two of which overlap ($[16, 26]$ and $[25, 35]$). In both cases, each time a group splits some of the occurrences it contains are lost, i.e., they are not part of any subgroup (of size at least σ_g) created by the split.

6 Related work

The need of quantitative temporal information in patterns over event sequences has been pointed in recent works in the data mining literature [12, 3, 11, 4, 6, 9].

An important difference between these approaches and the q-episodes introduced here, is that the former provide patterns in isolation, while q-episodes are related in tree structures. Such trees give a global view of how the occurrences of a pattern differentiate in homogeneous groups along the sequence of event types (from the first to the last element of the pattern).

Different notions of intervals are also considered. In [6] the intervals are not determined by the data but are fixed by the user; only the interval between the beginning and the end of a pattern is considered in [9]; and in [3] intervals are derived from intervals of occurrences of patterns of size two only.

The other approaches [12, 11, 4] compute the intervals from the data and for all pattern lengths, as in the case of the q-episodes. However, among these approaches, only [4] considers an exhaustive extraction (at the cost of intrinsically expensive algorithmic solutions), while the others compute only *some* of the patterns using heuristics and/or non-deterministic choices.

Finally, it should be noticed that the overhead of computing the quantitative temporal information was not assessed in these previous works.

7 Conclusion

In this paper we introduced *quantitative episodes*, an extension of serial episodes that refines standard episodes by integrating quantitative temporal information. A tight integration of episode extraction and group tree computation allowed to obtain a complete and efficient algorithm that adds a negligible overhead to the extraction of serial episodes, as assessed by the experimental results on performances. We think that these features, and the possibility of an easy-to-grasp representation of the output into a graphical tree-like structure, make the approach suitable for many applications.

References

1. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proc. of ICDE*, pages 3–14, 1995.
2. G. Das, L. K.I., H. Mannila, G. Renganathan, and P. Padhraic Smyth. Rule discovery from time series. In *Proc. of KDD*, pages 16–22, 1998.
3. C. Dousson and T. V. Duong. Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *Proc. of IJCAI*, pages 620–626, 1999.
4. F. Giannotti, M. Nanni, and D. Pedreschi. Efficient mining of temporally annotated sequences. In *Proc. of the SIAM Conference on Data Mining (SDM)*, 2006.
5. K. Hatonen, M. Klemettinen, H. Mannila, P. Ronkainen, and H. Toivonen. TASA: Telecomunications alarm sequence analyzer or: How to enjoy faults in your network. In *Proc. of IEEE Network Operations and Management Symposium*, pages 520–529, 1996.
6. Y. Hirate and H. Yamana. Sequential pattern mining with time intervals. In *Proc. of PAKDD*, 2006.
7. H. Mannila, H. Toivonen, and A. Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, 1(3):259–298, November 1997.
8. N. Meger, C. Leschi, N. Lucas, and C. Rigotti. Mining episode rules in STULONG dataset. In *Proc. of the ECML/PKDD Discovery Challenge*, 2004.
9. N. Meger and C. Rigotti. Constraint-based mining of episode rules and optimal window sizes. In *Proc. of PKDD*, pages 313–324, 2004.
10. M. Nanni and C. Rigotti. Quantitative episode trees. Technical report, 17 pages, 2006.
11. A. Vautier, M.-O. Cordier, and R. Quiniou. An inductive database for mining temporal patterns in event sequences. In *Proc. of ECML/PKDD Workshop on Mining Spatial and Temporal Data*, 2005.
12. M. Yoshida et al. Mining sequential patterns including time intervals. In *Proc. of SPIE Conference on Data Mining and Knowledge Discovery: Theory, Tools and Technology II*, 2000.
13. M. Zaki. Spade: an efficient algorithm for mining frequent sequences. *Machine Learning, Special issue on Unsupervised Learning*, 42(1/2):31–60, Jan/Feb 2001.