Knowl Inf Syst (2015) 45:35–74
DOI 10.1007/s10115-014-0784-5

REGULAR PAPER

Classification of multivariate time series via temporal abstraction and time intervals mining

Robert Moskovitch · Yuval Shahar

Received: 12 June 2013 / Revised: 1 May 2014 / Accepted: 4 September 2014 /
Published online: 1 October 2014

Abstract Classification of multivariate time series data, often including both time points and intervals at variable frequencies, is a challenging task. We introduce the KarmaLegoSification (KLS) framework for classification of multivariate time series analysis, which implements three phases: (1) application of a temporal abstraction process that transforms a series of raw time-stamped data points into a series of symbolic time intervals; (2) mining these symbolic time intervals to discover frequent time-interval-related patterns (TIRPs), using Allen's temporal relations; and (3) using the TIRPs as features to induce a classifier. To efficiently detect multiple TIRPs (features) in a single entity to be classified, we introduce a new algorithm, SingleKarmaLego, which can be shown to be superior for that purpose over a Sequential TIRPs Detection algorithm. We evaluated the KLS framework on datasets in the domains of diabetes, intensive care, and infectious hepatitis, assessing the effects of the various settings of the KLS framework. Discretization using Symbolic Aggregate approXimation (SAX) led to better performance than using the equal-width discretization (EWD); knowledge-based cut-off definitions when available were superior to both. Using three abstract temporal relations was superior to using the seven core temporal relations. Using an epsilon value larger than zero tended to result in a slightly better accuracy when using the SAX discretization method, but resulted in a reduced accuracy when using EWD, and overall, does not seem beneficial. No feature selection method we tried proved useful.

R. Moskovitch (✉) · Y. Shahar
Department of Information Systems Engineering, Ben Gurion University, Beer-Sheva, Israel
e-mail: robertmos@gmail.com; robertmo@bgu.ac.il

Y. Shahar
e-mail: yshahar@bgu.ac.il

R. Moskovitch
Department of Biomedical Informatics, Systems Biology, and Medicine, Columbia University, New York, NY, USA

Regarding feature (TIRP) representation, mean duration performed better than horizontal support, which in turn performed better than the default Binary (existence) representation method.

Keywords Temporal knowledge discovery · Temporal abstraction · Time intervals mining · Frequent pattern mining · Classification

1 Introduction

The increasing availability of time-stamped electronic data (e.g., Electronic Health Records, in the case of the medical domain) presents a significant opportunity to discover new knowledge from multivariate, time-oriented data, by using various data mining methods. In the information security domain, it might enable classification of hardware devices into infected and non-infected, by their temporal behavior [15,16,35,36]. Moreover, it enables researchers to perform classification and prediction tasks, based on the temporal data. However, temporal data include not only time-stamped raw data, or time *points* (e.g., white blood-cell count value of 4,300, at 11:32 a.m., on February 9th, 1991), but also temporal *intervals*, possibly at a higher level of abstraction, which are either a part of the original raw input data (e.g., administration of a medication for 4 days), or are *abstractions*, or interpretations, derived from them (e.g., 3 weeks of *severe anemia* or of *Bone-marrow toxicity Grade III*). Figure 1 illustrates the problem of having various time point series data and time interval series data representing various temporal variables in the same input dataset. Not only are the time points and intervals intermixed, but the time point series might be sampled or recorded at different frequencies: Sampling might occur at a fixed frequency, which may further vary for each type of variable, as shown in Fig. 1 for time series (a) and (b), which is often the case for automated sampling; or at random periods, as often occurs in manual measurements, as illustrated by time series (c). Often, certain time-stamped data points might be missing, or their measurements might include an error. Raw data (and certainly abstractions derived from the data) might also be represented by time intervals, such as medication-administration periods, as shown in series (d), in which the duration of the events is constant, and in series (e), in which the temporal duration is varying.

Designing algorithms capable of learning from such data, characterized in various forms, is a challenging topic in temporal data mining research, especially for the purpose of classification. Common methods for classification of multivariate time series, such as Hidden Markov Models [26] or recurrent neural networks, time series similarity measures (e.g., Euclidean distance or Dynamic Time Warping [27]), and time series feature extraction methods (e.g., discrete Fourier transform, discrete wavelet transform or singular value decomposition), cannot be directly applied to such temporal data.

Hu et al. [7] point out that most of the studies in time series classification had unrealistic underlying assumptions, referring specifically to two relevant assumptions (1) that perfectly aligned atomic patterns can be obtained and (2) that the patterns are of equal lengths. While they are referring in their examples to univariate time series, the critique is at least as relevant to multivariate time series classification. However, the approach we will present here does not make any of these assumptions, and mines the data as they are, without any alignment preprocessing. Moreover, as we describe later, the temporal patterns we discover have various durations (lengths). Finally, the duration is not part of a pattern's explicit or implicit definition; the mined supporting instances can vary in their duration (length), but they are similar with respect to the temporal relations among their symbolic interval-based components.

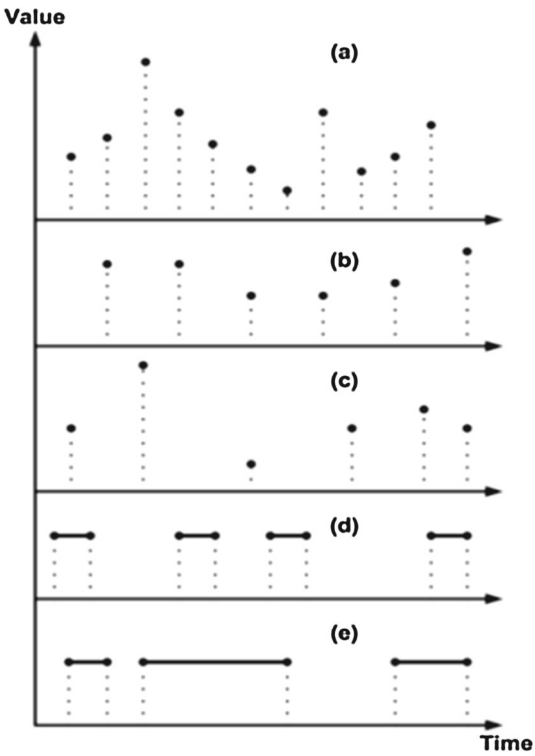

Fig. 1 Time-point (*a*, *b*, *c*) and time-interval data (*d*, *e*)

Thus, in order to classify multivariate time series datasets having various forms of time-stamped data, we propose to transform the time point series into symbolic time interval series representation. Such representation provides a uniform format of the various temporal variables, which enables to analyze the relations among the variables, such as by discovery of frequent temporal patterns. As we explain in Sect. 2, there are several approaches to that task, typically at a higher level of conceptual abstraction, which we refer to as temporal abstraction; some exploit context-sensitive knowledge acquired from human experts [33], and others are purely automatic [2,5,10]. Note that temporal abstraction can be applied to univariate as well as to multivariate time series. In the case of multivariate series, each univariate time series is abstracted independently. Conceptual representation is common in the medical domain and was found effective in medical information retrieval too [14,17].

Temporal abstraction enables us to overcome much of the problems of varying-frequency measurements and recordings, and of minor measurement errors and deviations in the raw data, through the creation of concepts (symbols) that are no longer time-stamped, raw data, but rather interval-based *abstractions*, or interpretations, of these data, and through the smoothing effect of these abstractions. In many instances, the temporal abstraction process also alleviates the problem of missing values, through a process of *interpolation* across temporal gaps that is inherent in several of the temporal abstraction methods (see Sect. 2). Note that raw data interval-based concepts such as "10 Days of Penicillin administration" might simply remain at the same level of abstraction. Whatever the temporal abstraction approach used, having the dataset represented by time intervals series, enables to discover frequent *Time-Interval-Related Patterns* (*TIRPs*) [5,8,11,20,23,24,29].

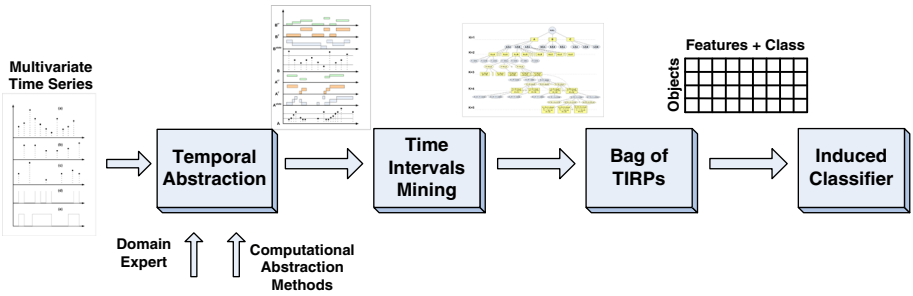

Fig. 2 KarmaLegoSification—the overall temporal classification process. The raw data time-point and time-interval series are abstracted into a uniform format of symbolic time intervals using domain knowledge or computational means. The symbolic time intervals are then mined, and a tree of enumerated temporal patterns is generated. The frequent TIRPs are used as features by one or more classification algorithms to produce (induce) a classifier

The use of TIRPs as features for the classification of multivariate time series was proposed first in [17,24]. This approach is inspired by the Bag-Of-Words in text categorization [31], in which words are used as features for the classification of a given document. In our case, the discovered TIRPs from a given period of multivariate time series are used as features, which we call a Bag-Of-TIRPs, and is the focus of this paper. Since then several more studies were published exploring the use of frequent TIRPs as classification features for multivariate time series classification [3,24,29].

The *KarmaLegoSification* (KLS) framework that is at the focus of this paper is presented in the general block diagram shown in Fig. 2. The input data include multiple instances of entities (e.g., patients, or hardware devices) whose multiple variables (e.g., Hemoglobin value or Number of processes) are described by multivariate time-point series. The time-point series are abstracted, based on domain knowledge [a *knowledge-based* (KB) method], or on other computational means, and transformed into symbolic-value time intervals (e.g., *Moderate-Anemia* from t_1 to t_2).

Then, the symbolic time intervals are mined using the *KarmaLego* algorithm, which we introduce in Sect. 3 in detail, to discover frequently repeating temporal patterns. The discovered TIRPs are used as features for the classifier.

After a TIRP tree is discovered using KarmaLego, several major application categories exist. These potential applications include: *Further temporal knowledge discovery*, in which a domain expert manually reviews the discovered TIRPs using a tool that we refer to as *KarmaLegoVisualization* (KLV) [17]; *Temporal clustering*, in which each temporal pattern is considered as a cluster of entities (e.g., patients, mobile devices) who have similar temporal behavior; extraction of *Prediction* rules, based on the discovered TIRPs and the transition probabilities between the components of the patterns; and *Classification*, in which the discovered TIRPs are used as features for a classification task, which is the focus of the current paper.

The main contributions of the current paper can be summed up as follows:

1. *Introducing a comprehensive architecture and process, KarmaLegoSification* (KLS), *for classification of multivariate time series*, which is based on a temporal abstraction process that can exploit domain knowledge, or, when needed, perform on-the-fly discretization, which introduces several fundamental concepts that define the output of the time intervals mining process;

2. Evaluating the effect of applying *several types of temporal abstraction methods, while also varying the number of bins* on the eventual classification performance.
3. *Quantitatively evaluating the difference* in classification performance when using Allen's original *seven temporal relations*, versus using an abstract version o*f only three temporal relations.*
4. Introducing *SingleKarmaLego* (SKL)—a new algorithm for *fast detection of multiple TIRPs within a single entity that is represented by a set of symbolic time intervals*, given a set of TIRPs as classification features to detect within the entity. SingleKarmaLego (SKL) can be shown to be significantly superior (both in its worst-case complexity, in its average-case behavior, and its empirical behavior) to a simpler, sequential TIRP detection (STD) method that attempts to query the single entity's record separately for each TIRP that is provided as a feature to detect.
5. *Introducing two novel representation metrics for features* (i.e., *TIRPs*)*, which exploit the e*xistence of multiple instances within each time-oriented record, in addition to a Binary r*epresentation, for the* purpose of classifica*tion.*
6. *Rigorously evaluating the KLS process and its several aspects, within several different real-life datasets*. As far as we know, no previous study has rigorously evaluated the effect all of these factors on the resulting classification, making it hard to assess their importance. Our evaluation addresses this situation.

The rest of this paper is organized as follows: We start by introducing briefly, in the Background (Sect. 2), the concepts of temporal data mining, temporal abstraction, temporal discretization, time interval mining, and classification based on patterns—specifically based on time intervals patterns. We then briefly introduce in the Methods (Sect. 3) a fast, time-interval mining method that we had developed, called KarmaLego, and show how it can be used to discover frequent TIRPs, and its customized version for the detection of TIRPs in a given entity, called SingleKarmaLego. We then explain exactly how TIRPs are used as features in our temporal data mining framework and describe in Sect. 4 our detailed evaluation of the efficacy of TIRP based classification of time-oriented data, and the precise effect of the various settings with which we had experimented. We summarize the main contributions and discuss their implications in Sect. 5. The "Appendix" presents an example of our empirical runtime evaluation of the SingleKarmaLego algorithm versus a STD algorithm, demonstrating its significant superiority (the full theoretical and practical analysis of the SingleKarmaLego algorithm is out of the scope of the current paper, and the reader is referred elsewhere for additional details).

2 Background

2.1 Temporal data mining

Temporal data mining is a sub-field of data mining, in which various techniques are applied to time-oriented data to discover *temporal knowledge*, i.e., knowledge about relationships among different raw data and abstract concepts, in which the temporal dimension is treated explicitly. Unlike common data mining methods, which are static, often ignoring the temporal dimension, or using only concise statistical abstractions of it, temporal knowledge discovery presents significant computational and methodological challenges. However, temporal data mining offers considerable understanding of various scientific phenomena and the potential

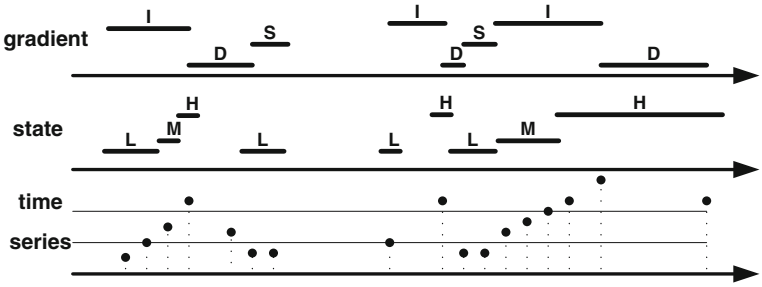

Fig. 3 A series of raw time series (at the *bottom*) is abstracted into an interval-based *state* abstraction that has three discrete values: Low (*L*), Medium (*M*), and High (*H*) (in the *middle*); and into a *gradient* abstraction (first derivative's sign) that has the values: Increasing (*I*), Decreasing (*D*), and Stable (*S*) (at the *top*)

for creation of richer and accurate classification models, representing explicitly processes developing a long time. An excellent survey of temporal data mining can be found in [28].

2.2 From time point series to time intervals series

Transforming from time point series into time intervals series, often called Temporal abstraction (*TA*), is increasingly used [2,6,10,30], is the segmentation and/or aggregation of a series of *raw*, *time-stamped*, multivariate data into a *symbolic time interval* series representation, often at a higher level of *abstraction* (e.g., instead of a series of Hemoglobin or liver-function measurements, characterizations such as "3 weeks of moderate anemia," or "5 months of decreasing liver functions"), suitable for human inspection or for data mining. TA, which typically includes also some form of interpolation [32], solves several common problems in mining raw time series data, such as high variability in the sampling frequency and temporal granularity, minor measurement errors, and missing values, through the smoothing effect of the output abstractions. Thus, discovering frequent temporal patterns in multivariate temporal data can benefit from a preprocessing phase of converting the raw time-stamped data into a series of uniform, symbolic time intervals. Figure 3 shows a TA process for one temporal variable.

There are several approaches to the TA task; some exploit context-sensitive knowledge acquired from human experts [30], and others are purely automatic and rely mostly on a discretization of the raw values and concatenation [2,6,10]. *Temporal discretization* refers to the process of discretization of a time series values, usually performed through unsupervised means, as a preprocessing step in transforming the time-stamped, raw value series into a set of symbolic, state-based time intervals.

2.3 Temporal discretization

Temporal discretization refers to the process of discretization of a time series values, usually performed through unsupervised means, as a preprocessing step in transforming the time-stamped, raw-concept series into a set of symbolic, state-based time intervals. Temporal Abstraction for time series mining in the form of time intervals was already proposed by Hoeppner [6]. Several common discretization methods, such as *Equal-Width Discretization* (*EWD*), which uniformly divides the ranges of each value, and *Equal Frequency Discretization* (*EFD*), do not consider the temporal order of the values; other methods, such as *Symbolic*

Aggregate approXimation (*SAX*) [9] (which focuses on a statistical discretization of the values) and *Persist* [10] (which maximizes the duration of the resulting time intervals), explicitly consider the temporal dimension. In previous studies, we have compared several versions of these methods, especially for the purpose of discretization of time-oriented clinical data [2] and eventually for the purpose of classification [17].

In the evaluations performed in the current study, we have used time interval-based abstract concepts generated through the use of both KB temporal abstraction and automated temporal discretization using EWD and SAX [9].

2.4 Mining time intervals

Mining time intervals is a relatively young research field that has mostly sprung during the past decade. Most of the methods use some subset of Allen's temporal relations [1]. As was presented by Moerchen [12] these suffer from "crispiness" and being sometimes over-detailed for knowledge discovery purposes, thus, in our framework, two ways of overcoming this are proposed and rigorously evaluated. One of the earliest studies in the area is that of Villafane et al. [38], which searches for containments of time intervals in a multivariate symbolic time interval series. Kam and Fu [8] were the first to use all of Allen's temporal relations to compose time interval series, but their patterns were ambiguous, since the temporal relations among the components of a pattern are undefined, except for the temporal relations among all of the pairs of successive intervals.

Höppner [5] was the first to define a non-ambiguous representation of time-interval patterns that are based on Allen's relations, by a k^2 matrix, to represent all of the pairwise relations within a k-intervals pattern. In the rest of this paper, we shall refer to a conjunction of temporal relations between pairs of intervals as a *TIRP*. The formal definition of a TIRP appears in Sect. 3 (Definition 5). Papapetrou et al. [23] propose a hybrid approach H-DFS, which combines the first indexing the pairs of time intervals and then mining the extended TIRPs in a candidate generation fashion. Papapetrou et al. used only five temporal relations: meets, matches (equal, in terms of Allen's relations), overlaps, contains, and follows, similar to Allen's temporal relations, and introduced an *epsilon threshold*, to make the temporal relations more flexible.

ARMADA, by Winarko and Roddick [39], is a relatively recent projection-based efficient time intervals mining algorithm that uses a candidate generation and mining iterative approach. Wu and Chen [40] proposed TPrefixSpan, which is a modification of the PrefixSpan sequential mining algorithm [25] for mining non-ambiguous temporal patterns from time interval-based events. Patel et al. [24] introduced IEMiner—a method inspired by Papapetrou's method, which extends the patterns directly, unlike Papapetrou et al.'s method, as in fact is done in the KarmaLego method. Patel et al. [24] had compared their method runtime to TPrefixSpan [40] and H-DFS [23] and found their method to be faster. Moskovitch and Shahar introduced the KarmaLego framework [18–20], which extends TIRPs directly and exploits the transitivity property to generate candidates efficiently, which was found faster than H-DFS, IEMiner and ARMADA. (We present the KarmaLego algorithm briefly in Sect. 3).

Other methods for time intervals mining were proposed, which either do not use Allen's temporal relations [12], or use only a subset of these relations, abstracted into a super-relation, such as *Precedes*, which is the disjunction of *Before, Meets, Overlaps, Equal, Starts*, and *Finished-By*, and which can form a basis for the discovery of a set of temporal association rules [29].

2.5 Classification via frequent patterns

The increased attention to the subject of mining time intervals has led several research groups to simultaneously propose using the discovered temporal patterns for classifying multivariate time series [17,24], including the suggestion of a highly preliminary version of the KarmaLegoS framework [17] whose full details and rigorous evaluation we present in Sects. 3 and 4, respectively.

Patel et al. [24] presented a time intervals mining algorithm, called IEMiner, and used the discovered patterns for classifying multivariate time series. A patterns selection measure, GAIN, which is actually a feature selection method, is proposed. The GAIN measure is an entropy-based measure, assumes that temporal patterns that occur in only one class are more discriminative. In addition to GAIN, the IEClassifier is introduced, a classification method that is designed specifically to classify data using temporal patterns [24]. Two versions of IEClassifier, each with a different criterion, were proposed. In the Best_Confidence version, the class having the highest confidence is selected, while in the Majority_Class version, the class to which the majority of the patterns discovered belong is selected. The two versions of IEClassifier were compared with common classifiers, such as C4.5 and SVM, when applied to a non-temporal representation of the data. In the data used in the study, The Majority_Class out-performed the other classification methods. Batal et al. [3] present a study in which time interval patterns classify multivariate time series, using an a priori approach, STF-Mine, for the discovery of temporal patterns, and the χ^2 (chi-square) measure for pattern selections.

Batal et al. [4] presented an approach for targeted events prediction based on time intervals patterns events. For that, they used only two temporal relations: "Before" (which is similar in semantics to Allen's relation) and "Overlaps" (the later relation is actually a disjunction of all Allen's temporal relations, except "Before"). Moerchen and Fradkin [13] introduce the use of semi-intervals mining with partial order to enable more flexibility, as opposed to Allen's [1] interval-based temporal relations. Two flexible representations were proposed, termed "SISP and SIPO," for semi-intervals patterns, inspired by Wu and Chen [40]. The authors evaluate their pattern representation, comparing it to Allen's temporal relations, on seven datasets, some originating from time intervals data, and some abstracted from multivariate time series using Persist [10], using the Precision and Recall measures to assess directly the value of the new features. The authors' results show that an additional number of frequent patterns were found with their disjunctive representation. However, since training and classification were not used, and accuracy results were not reported, it is hard to assess the effectiveness of these features, unlike the experiments that we present in our current study. Although in recent years, several studies have started to use time interval patterns for classification purposes, no study investigated the influence of the abstraction method on the accuracy, as well as the number of bins, which is one of the several contributions of our work.

3 Methods

We start by introducing our method for discovering frequent patterns, the KarmaLego framework; we then explain in detail how we extended that framework into the KarmaLegoS classification method, and in the next section, how we rigorously evaluated that method in several different domains.

3.1 KarmaLego: fast time intervals mining

The discovery of *TIRPs* is computationally highly demanding, since it requires generating all of Allen's seven basic temporal relations. For example, a naive generation of all TIRPs having five symbolic time intervals, with all possible temporal relations among them specified unambiguously, requires in theory generating up to $7^\wedge((5^2-5)/2) = 7^\wedge 10 = 282,475,249$ candidate TIRPs. In general, given a TIRP having k time intervals, we will have up to $7^\wedge((k^2-k)/2)$ candidate TIRPs.

To overcome this difficulty, we have developed *KarmaLego*, a fast algorithm which generates all of the patterns efficiently by extending TIRPs directly, and exploiting the transitivity property of the temporal relations to remove unrealistic candidates.

To increase the robustness of the discovered temporal knowledge, KarmaLego uses a flexible version of Allen's seven temporal relations. This is achieved by adding an epsilon value to all seven relations, as explained in the next subsection. Furthermore, we also limit the *before* temporal relation by a maximal allowed gap, as proposed by Winarko and Roddick [39].

To formally define the problem of mining time intervals and relevant measures for the classification task, we first present several basic definitions. These definitions will be used in the description of the methods.

3.2 Definitions

To better understand the KarmaLegoS methodology, we introduce several key definitions, some of which extend our initial definitions of the KarmaLego framework [20].

To define a flexible framework of Allen's temporal relations in KarmaLego, two relations are defined on time-stamped (point-based) data, given an epsilon value.

Definition 1 Given two time points t_1 and t_2 : $t_1 =^\varepsilon t_2$ iff $|t_2 - t_1| <= \varepsilon$ and $t_1 <^\varepsilon t_2$ iff $t_2 - t_1 > \varepsilon$. Based on the two relations $=^\varepsilon$ and $<^\varepsilon$ and the epsilon value, a flexible version for all of Allen's seven relations is defined, extending Papapetrou's definition, as shown in Fig. 4.

The introduction of the epsilon parameter to Allen's full set of temporal relations maintains the *Jointly Exhaustive and Pairwise Disjoint* (*JEPD*) conditions, as will be shown soon. The

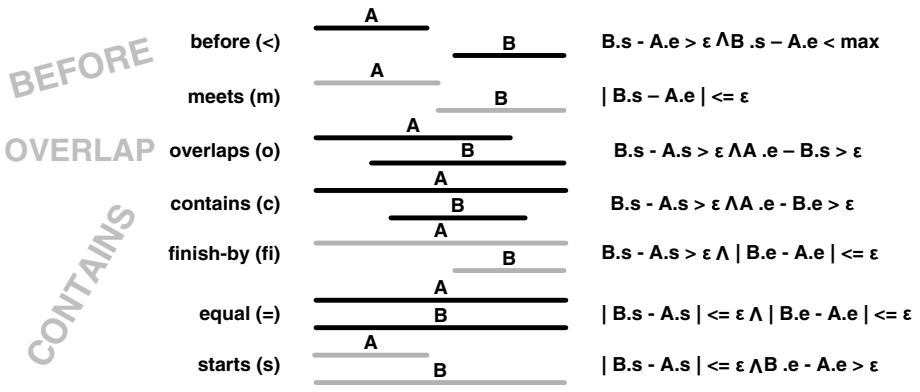

Fig. 4 A flexible extension, using the same epsilon value, for all Allen's seven relations, using the same Epsilon for all the relations

Jointly Exhaustive condition comes from probability theory and means that a set of events is jointly exhaustive if at least one of the events must occur. In the context of temporal relations, it means that the set of temporal relations, which are defined, must cover all of the optional temporal relations among two time intervals.

The pairwise disjoint condition means that two sets A and B are disjoint if their intersection is the empty set. In the context of temporal relations, it means that the introduction of the epsilon value as defined in Definition 1 and Fig. 4 keeps the set of the temporal relations mutually exclusive. This is indeed true, since the epsilon-extended temporal relation definitions appearing in Fig. 4 imply that for any two time intervals, exactly one (epsilon-extended) temporal relation applies.

In addition to a flexible (epsilon-based) semantics for Allen's seven temporal relations, we introduce a set of three abstract temporal relations, each of which is a disjunction of several of Allen's seven basic temporal relations (shown in Fig. 4 in gray labels): BEFORE = {before | meet} stands for the disjunction of Allen's before and meets relations, OVERLAP is simply Allen's overlap relation, and CONTAIN = {finish-by | contains | starts | equal} represents the disjunction of finished-by, contains, starts and equal.

Definition 2 A *symbolic time interval*, $I = <s, e, sym>$, is an ordered pair of time points, start-time (s) and end-time (e), and a symbol (sym) that represents one of the domain's symbolic concepts. To make the representation uncluttered, $I.s$, $I.e$, and $I.sym$ are used when the start-time, end-time, and symbol of an interval I are referred to.

Definition 3 A *symbolic time interval series*, $IS = \{I^1, I^2, \ldots, I^n\}$, where each I^i is a symbolic time interval, represents a series of symbolic time intervals, over each of which holds a start-time, end-time and a symbol.

Definition 4 A *lexicographical symbolic time-interval series* is a symbolic time interval series, sorted in the lexicographical order of the start-time, end-time using the relations $<^{\varepsilon}$, $=^{\varepsilon}$ and the symbols, $IS = \{I^1, I^2, \ldots, I^n\}$, such that:

$$\forall I^i, I^j \in IS(i < j) \wedge \left((I^i_{.s} <^{\varepsilon} I^j_{.s}) \vee \left(I^i_{.s} =^{\varepsilon} I^j_{.s} \wedge I^i_{.e} <^{\varepsilon} I^j_{.e}\right)\right.$$
$$\left. \vee \left(I^i_{.s} =^{\varepsilon} I^j_{.s} \wedge I^i_{.e} =^{\varepsilon} I^j_{.e} \wedge I^i_{.sym} < I^j_{.sym}\right)\right)$$

Since in our problem definition the time intervals are ordered lexicographically, we use only the seven temporal relations shown in Fig. 5.

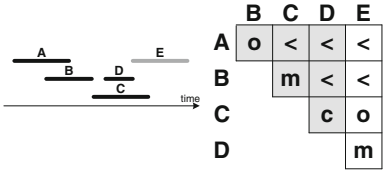

Fig. 5 An example of a time-interval-related pattern (TIRP), having five lexicographically ordered symbolic time intervals and all of their pairwise temporal relations. On the *left*, the actual five-symbol TIRP is displayed graphically, while on the *right*, a half-matrix representation is given presenting the pairwise temporal relations. Interval E is a candidate symbol that is being added to the current TIRP, and its temporal relations with the other four symbolic intervals are shown in the *last column* of the half matrix

Definition 5 A non-ambiguous lexicographical *Time-Interval-Related Pattern* (*TIRP*) P is defined as $P = \{I, R\}$, where $I = \{I^1, I^2, \ldots, I^k\}$ is a set of k symbolic time intervals ordered lexicographically and

$$R = \bigcap_{i=1}^{k-1} \bigcap_{j=i+1}^{k} r\left(I^i, I^j\right) = \left\{r_{1,2}\left(I^1, I^2\right), \ldots, r_{1,k}\left(I^1, I^k\right), r_{2,3}\left(I^2, I^3\right), \ldots, r_{k-1,k}\left(I^{k-1}, I^k\right)\right\}$$

defines all the temporal relations among each of the $(k^2-k)/2$ pairs of symbolic time intervals in I. Figure 5 presents a typical TIRP, represented as a half matrix of temporal relations. We will usually assume such a representation through the description of the KarmaLego algorithm.

One potential problem with Definition 5 is that it is purely qualitative; it ignores the precise quantitative durations of the time intervals that are the components of the TIRP. We focused on this problem, and on a possible solution of it, in [20].

Figure 5 presents the output of a temporal interval mining process, i.e., an example of a TIRP, represented, for efficiency purposes, as a half matrix. Thus, the half matrix on the right part of Fig. 5 presents all of the pairwise temporal relations among the TIRP's symbolic time intervals, ordered lexicographically, that defining it in a canonical, non-ambiguous fashion. Note that *half-matrix* representation (as opposed to a full matrix) is possible due to the fact that each of Allen's temporal relations has an inverse and that the *canonical* aspect is due to the lexicographical ordering, which leads to a unique half matrix for each TIRP.

Definition 6 Given a database of $|E|$ distinct entities (e.g., different patients), the *vertical support* of a TIRP P is denoted by the cardinality of the set E^P of distinct entities within which P holds at least once, divided by the total number of entities (e.g., patients) $|E|$: $ver\_sup(P) = |E^P|/|E|$. The vertical support is the term usually referred to as *support* in the context of association rules, itemsets, and sequential mining.

When a TIRP has vertical support above a minimal predefined threshold, it is referred to as *frequent*. Note that in pattern mining, such as association rules, sequential mining, and time intervals mining, *support* typically refers to the percentage of entities in the database supporting a pattern, which is actually the vertical support presented in Definition 6.

Since a temporal pattern can be discovered multiple times within a single entity (e.g., the same TIRP appears several times (multiple instances) in the longitudinal record of the same patient), we distinguish between two types of support: the *vertical* support and the *horizontal* support, which represents the number of patterns discovered within the longitudinal record of a specific entity, as defined in Definition 7.

Definition 7 The *horizontal support* of a TIRP P for an entity e_i (e.g., a single patient's record), $hor\_\text{sup}(P, e_i)$ is the number of instances of the TIRP P found in e_i. For example, the number of times that a particular temporal pattern was found in a particular patient's record.

Definition 8 The *mean horizontal support* of a TIRP P is the average horizontal support of all the entities E^P supporting P (i.e., for all entities that have a horizontal support ≥ 1 for TIRP P).

$$MeanHorizontalSupport(P, E^P) = \frac{\sum_{i=1}^{|E^P|} hor\_\sup(P, e_i)}{|E^P|}$$

Definition 9 The mean duration of the n supporting instances of the same *k-sized* TIRP P within an entity e (e.g., within a single patient's record; note that, per Definitions 7 and 8, an entity may have several supporting instances of a TIRP) is defined by:

$$MeanDuration(p, e) = \frac{\sum_{i=1}^{n}(Max_{j=1}^{k} I_e^{i,j} - I_s^{i,1})}{n}$$

where $I_s^{i,1}$ is the *start-time* (s) of the first time interval in the i instance (among n instances), and the Max operator selects the time interval having the latest *end-time* (e) among the k time intervals of an instance i. Note that according to the lexicographical order (Definition 4), the first time interval must have the earliest start-time, while the latest end-time can be of any of the time intervals in the instance.

As we shown in Sect. 4, the horizontal support and the mean duration of TIRP are potentially useful metrics when using TIRPs as features, for classification purposes.

Definition 10 (*The time-interval mining task*) Given a set of entities E, described by a symbolic time-interval series *IS*, and a minimum vertical support *min_ver_sup*, the goal of the tasks is to find all the TIRPs whose vertical support is above the *min vertical support* threshold.

3.3 The KarmaLego algorithm

KarmaLego is a TIRP discovery algorithm that we have defined and evaluated previously, as part of our research into effective TIRP-based classification [20]. Here, we summarize the key features of KarmaLego that are relevant to the current study.

The main body of the KarmaLego algorithm consists of two main phases, Karma and Lego (Algorithm 1). In the first phase, referred to as *Karma*, all of the frequent two-sized TIRPs, $r(I_1, I_2)$ having two symbolic time intervals I_1 and I_2 that are ordered lexicographically and are related by r, a temporal relation, are generated, discovered, and indexed, using a breadth-first-search approach. In the second phase, referred to as *Lego*, a recursive process extends the frequent 2-sized TIRPs, referred to as T^2, through efficient candidate generation, using the Lego procedure, into a tree of longer frequent TIRPs consisting of conjunctions of the 2-sized TIRPs that were discovered in the Karma phase. Eventually a tree of all frequent TIRPs is discovered and returned (Fig. 6).

The data structure supporting this process is a hierarchy of frequent TIRPs, as illustrated in Fig. 6. A TIRP is represented by a vector of symbols in their lexicographical order, and a two-dimensional array for the temporal relations representation, corresponding to the illustration of the half matrix of temporal relations shown in Fig. 4. Each TIRP is supported by a list of instances that are represented by a vector of time intervals instances that satisfy the TIRP's constraints regarding the symbols and temporal relations.

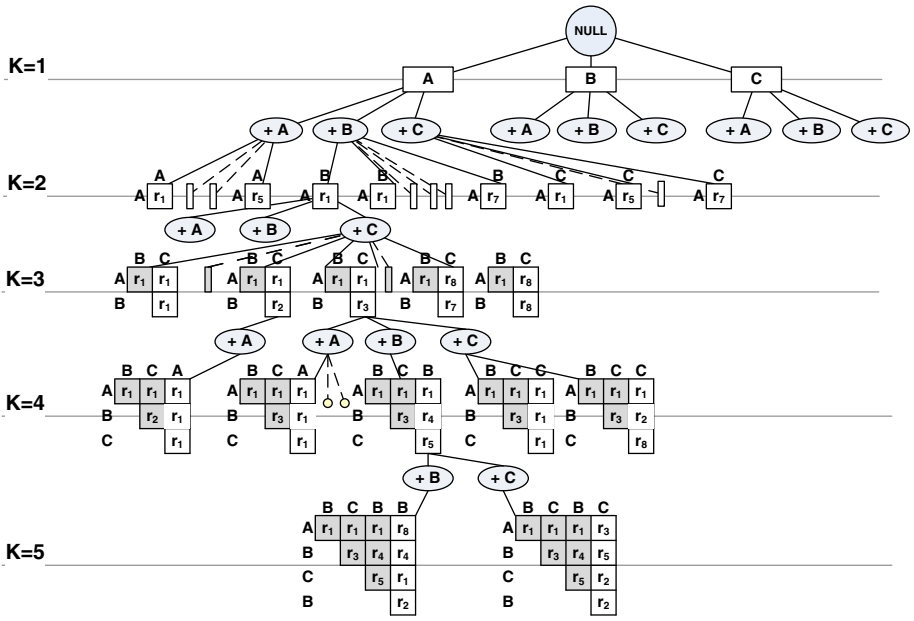

Fig. 6 A KarmaLego enumeration tree, in which the direct expansion of TIRPs is performed. Each node represents a TIRP as a half matrix of its relevant temporal relations

Algorithm 1 – KarmaLego
Input:
db – A database of |E| entities, representing for each entity *e* a set of symbolic time intervals out of |S| symbols;
min_ver_sup – the minimal vertical support threshold;
Output: *T* – an enumerated tree of all frequent TIRPs

1. T ← Karma(*db*, min_ver_sup)
2. Foreach $t \in T^2$ // T^2 is T at the 2nd level
3. Lego(T, t, min_ver_sup)
4. End Foreach
5. return T
6. End

Note that, for robustness purposes, we are using the flexible version of Allen's temporal relations (see Definition 1). However, *the KarmaLego algorithm is oblivious to the precise definition of temporal relations*. This is demonstrated in this study, in which two sets of temporal relations are used, Allen's original seven temporal relations versus an abstract form of three temporal relations.

3.3.1 Karma

In Algorithm 2 (*Karma*[1]), the first level of the enumeration tree is constructed based on the set of the frequent symbols *S*, as shown in Fig. 6. Then, in a single scan of the data each

[1] Karma—The law of *cause* and *effect* originated in ancient India and is central to Hindu and Buddhist philosophies.

entity e's pairs of symbolic time intervals (constrained by the maximal gap duration, shown in line 5) are indexed by their two symbols and the relation r among them, which creates the second level of the enumeration tree T^2. Finally, the 2-sized TIRPs that are not frequent are removed.

To enable later a highly efficient retrieval of the 2-sized TIRPs for two given symbols (i.e., pairwise relations between intervals that are annotated by two given symbols) for the recursive Lego step, T^2 is implemented by a two-dimensional square array of size $|T^1|^2$. $|T^1|$ is the number of the frequent symbols found in Karma. Each cell in the two-dimensional array contains a vector of size $|R|$ (the number of temporal relations) that holds a HashMap, which contains all of instances of the indexed pairs. Thus, retrieval of the indexed pairs from T^2 in the Lego phase is performed in $O(1)$, using as indices the two symbols and the temporal relation. As we explain later, this structure is also highly useful in SingleKarmaLego.

Algorithm 2 – Karma
Input:
db – A database of |E| entities (the overall set of entities being referred to as E), representing for each entity *e*, the lexicographically sorted vector of its symbolic time intervals, *e*.I;
min_ver_sup – the minimal vertical support threshold;
Output: T – an enumerated tree of up to 2-sized frequent TIRPs

1. $T \leftarrow \emptyset$
2. Foreach $e \in E$
3. Foreach $I^i, I^j \in e.I \wedge i<j$
4. $r \leftarrow$ the temporal relation among I^i, I^j
5. if(r > max_gap)
6. break
7. else
8. Index(T^2, < $e.I_{.sym}^{\ i}$, r, $e.I_{.sym}^{\ j}$>)
9. End Foreach
10. End Foreach
11. Foreach $t \in T^2$
12. if ver_sup(t) < min_ver_sup
13. Prune(t)
14. End Foreach
15. return T
16. End

3.3.2 Lego

The *Lego*[2] (Algorithm 3) receives a k-sized TIRP and extends it by all of the frequent symbols in T^1 and a set of predefined desired temporal relations R (Allen's seven relations, or other versions). This *direct extension* approach of generating TIRPs candidates is part of the KarmaLego framework's contribution and is presented in more detail elsewhere [20]. Then, a new TIRP is created of the extended size and a symbol s and a temporal relation r are set to t^c and the method Generate Candidate TIRPs generates all the corresponding TRIPs based on the various temporal relations settings. Then, for each candidate c in C, if the frequency of the supporting instances is above the minimal vertical support threshold, it is further extended by calling Lego.

[2] Lego—A popular game, in which modular bricks are used to construct different objects. [also, Le(t)go].

The *candidate generation* process can be performed by using a naïve candidate generation method, as has been done in previous algorithms, or through a much faster method, exploiting the transitivity of temporal relations, which leads to significant pruning of unviable candidate TIRPs, which is implemented in KarmaLego [20].

Algorithm 3 – Lego(T, t, min_ver_sup)

Input:
T – the enumeration tree after Karma was ran,
t – a TIRP that has to be extended,
min_ver_sup - the minimal vertical support threshold
Output: void

1. Foreach $sym \in T^1$
2. Foreach $r \in R$
3. Create new t^c of size ($t.size$ + 1)
4. $t^c.s[t^c.size - 1] \leftarrow sym$
5. $t^c.tr[t^c.tr\_size - 1] \leftarrow r$
6. $C \leftarrow 0$
7. C ← Generate_Candidate_TIRPs(t^c, *1*)
8. Foreach $c \in C$ // candidates
9. Search supporting instances(c, T^2)
10. if(ver_sup(c) > *min_ver_sup*)
11. $T \leftarrow T \cup c$ // c is frequent
12. Lego(T, c, *min_ver_sup*)
13. End Foreach
14. End Foreach
15. End Foreach
16. End

The entire process of searching for supporting instances is done using the 2-sized TIRPs. In each extension, the TIRP is extended by a relation to a symbol that has to be supported by the 2-sized TIRPs and its corresponding relations to the previous symbols. This process is repeated as necessary, while maintaining each time the correct entity and the relevant set of symbolic time intervals for which the (partial) conjunction holds to far. Finally, the vertical support (number of entities for which the full extended TIRP holds) is checked to determine whether to add it to the TIRP tree.

A potential problem that may occur when mining abstracted time intervals is the discovery of sequences of states of the same temporal variable. For example, given a temporal variable Hemoglobin A1c (HbA1c), its (state) abstracted symbols might be generated by any of the abstraction methods, representing different value ranges; these symbols might be, for example, HbA1c.1, HbA1c.2, and HA1c.3. Thus, a common occurrence is the discovery of sequences such as a symbolic time interval labeled by HbA1c.1 followed by a symbolic interval labeled by HbA1c.2, followed by a symbolic interval labeled by HbA1c.3, etc. ("followed" here is in the sense of the temporal relations *meet* or *before*). Most such potential TIRPs are of low value for classification purposes, since they occur frequently throughout the entire dataset and are common to all classes. In addition, discovering patterns that include such chains, resulting from the same temporal variable, incurs a high computational cost.

In our experiments, to avoid the discovery of such TIRPs, we did not expand TIRPs in Lego with symbols that resulted, from the same temporal variable. Thus, in line 1, we used all the symbols (*sym*) in T^1, except for those that result from the temporal variable from which

the last symbol in the extended TIRP was abstracted. Thus, a TIRP can contain symbols that were abstracted from the same temporal variable, but they cannot follow each other: At least one symbol that is not derived from the same temporal variable must be between them.

The complexity of a time intervals mining algorithm Since the current study is focused on the use of TIRPs discovered, one way or another, from some training set, for the specific purpose of classification of new, single entities, we shall not dwell too much here on the computational complexity of the KarmaLego algorithm. However, a rigorous empirical runtime evaluation was previously performed to compare the KarmaLego algorithm's performance to several state of the art algorithms, as shown in a recent study [20], demonstrating its significant runtime advantages.

As is often the case in complex search algorithms, especially heuristic search algorithms, it is not easy to accurately analyze the KarmaLego algorithm's theoretical runtime complexity, nor is it easy to assess theoretically the effect of the multiple improvements that we added to the algorithm, compared to previous methods. Nevertheless, by making several reasonable assumptions, we can assess the worst-case complexity for any TIRP enumeration algorithm, and in particular, the KarmaLego algorithm's *worst-case* complexity. The analysis is out of the scope of the current study, but for the reader's convenience, we summarize the results of this worst-case analysis, whose details appear elsewhere [21].

Assume:

S is the number of the symbolic concept types $C_1 \ldots C_s$;
E are the entities $E_1 \ldots E_m$, each with $N_i (i = 1 \ldots |E|)$ symbolic time intervals $I_1 \ldots I_{Ni}$;
$N =$ total number of symbolic time intervals for all entities $= \sum N_i$
R is the number of temporal relations, $R \leq 13$; after using a lexicographical ordering, $R \leq 7$ (without $F,\ O_i,\ D,\ A,\ M_i, S_i$).
$L =$ maximal number of symbolic time intervals in a TIRP, or its maximal length; Note that $L \leq \text{Max}(N_i), (i = 1 \ldots |E|)$.

As it turns out, the overall complexity of creating the fullest TIRP tree is bounded *in the worst case* by the following expression:

$$O\left(S^L R^{L(L-1)/2} N^2\right).$$

(Note that the Karma-phase $O(\text{N}^2)$ setup cost, for indexing all interval-pair temporal relations for access in $O(1)$, is already overshadowed at this point and thus can be disregarded.)

Note that the major factor affecting the worst-case complexity is the maximal possible-TIRP length L, measured in number of symbolic concepts. Of course, L cannot be assessed ahead of time, as it depends on each particular dataset and on the minimal vertical support required. However, the analysis does suggest that the most effective way of limiting the potential exponential explosion of a TIRP detection process is by limiting the maximal TIRP length L, regardless of the dataset's properties or the minimal vertical support value.

Nevertheless, we must emphasize that this worst-case, upper-bound complexity expression far overshoots the practical situation for the KarmaLego algorithm. Indeed, in our recent study, we had demonstrated the highly significant efficacy of the various computational steps that we had taken to reduce the average number of candidates generated for directly extending a TIRP [20]. The main reduction is achieved through the use of the transitivity of temporal relations, such that the overall number of candidates actually generated is smaller by an order of magnitude, and even two or three orders of magnitude, compared to the number

generated by previous algorithms, which do not exploit the transitivity property. In addition, we are using an especially efficient data structure to represent temporal relations between all symbolic interval pairs.

Thus, the worst-case complexity of TIRP enumeration presented here is a rather theoretical upper bound for all TIRP discovery methods, and in practice, as shown by the KarmaLego algorithm's evaluation, can be significantly much smaller.

The importance of discovering all of the entity's TIRPs KarmaLego discovers *all* of the frequent TIRPs of an entity (e.g., a patient record), including *all* of the instances of that TIRP over time within that entity. This *completeness* aspect actually enables us later to calculate the novel TIRP representations of horizontal support and mean duration.

It is important to note here that, in order to make the output of a time intervals mining algorithm such as KarmaLego complete, we must discover *all* of the *horizontally* supporting instances of a TIRP (see Definition 8). This is always the case, because we do not know ahead of time, which of the TIRP instances of size $k-1$ within a specific entity will be followed by a symbolic time interval that will (vertically) support an extended k-sized TIRP, and in particular, cannot assume that it will necessarily be the first instance that we discover.

For example, suppose we detect three instances of the 2-sized TIRP <A before B> within a given entity; we cannot know which one of them (if at all) will support the extended 3-sized TIRP, <(A before B) $\wedge$ (A before C) $\wedge$ (B overlaps C)>. For example, if only the third instance of the 2-sized TIRP is followed by a symbolic time interval C (having also the proper relations to A and B), discovering the first instance and stopping will not enable us to discover any evidence for the existence of the 3-sized TIRP within this entity, although it certainly does exist, and will decrease its overall vertical support within the overall dataset.

Therefore, it is also important to note that thus, *there is no additional real cost in discovering all of the instances of a given TIRP* within each entity, since finding all of the instances of a given pattern for each entity is actually *obligatory*, in order to ensure *completeness* in the discovery of all [possibly longer] TIRPs, even just for the purpose of determining whether these TIRPs exist!

(As we shall see later, determination of such an existence is crucial for classification purposes. An example is when determining whether a given [new] entity to be classified includes a feature, namely a TIRP, which was previously discovered in the training set. We shall refer to such an existence as a *binary* representation of that feature. Furthermore, detecting *all* of the TIRPs within the entity is important when the method of representing TIRPs as features uses the Horizontal Support of the entity for that TIRP [see Definition 7] or the TIRP's Mean Duration within that entity [see Definition 9].)

We are highlighting this important point, since in typical previous TIRP discovery methods, the algorithm stops when discovering the first instance of a new pattern and does not include the next one(s) in its potential considerations for extension. We consider such approaches to be wrong, since in general, it is impossible to know ahead of time, which instance of the pattern, for a given entity, will be (or can be) extended later to support extended TIRPs. Thus, *all* instances of the $k-1$ sized TIRP must be found to ensure that if the k-sized TIRP exists all, it will be discovered. In other words, if one's objective (as is the case here, when we discover all frequent TIRPs for a given set of entities, and certainly when we later look at TIRPs as classification features that might hold for each single entity) is completeness, i.e., to discover, within each entity, *all* of the different TIRPs existing within it, not just soundness, i.e., discovering *some* TIRPs that do exist in it, then, regardless of whether we are considering a single entity or a set of entities, *all* of the $k-1$-sized TIRPs

within each entity *must* be discovered, in order to enable discovery of *all* of the k-sized TIRPs in that entity (where k is the number of symbolic intervals in the TIRP).

3.4 KarmaLegoSification: classification of multivariate time series via TIRPs

The KLS framework for the classification of multivariate time series via TIRPs includes four main components, as was shown in Fig. 2: Temporal Abstraction, Time Intervals Mining, TIRP-based Feature Representation, and Classifier induction, thus producing a TIRP-based Classifier.

Each component in the KLS framework (see Fig. 2) has several potential settings. The temporal abstraction method settings (e.g., SAX, EWD, KB) and the number of discrete states (symbols) generated. The time-interval mining method (i.e., KarmaLego) settings include the epsilon value and the minimal vertical threshold. The TIRP-based representation settings include the feature selection method and the representation method of the TIRPs (a Boolean value representing an existence of the TIRP, versus actual numerical horizontal support), and the classifier-induction setting is the type of induction method used.

Before we delve into the KLS framework, it is important to understand the difference between the process of TIRP-based classification and (cross) validation, and the one often used in the context of standard classification experiments.

3.4.1 Time intervals mining: the training versus the classification phases

When using TIRPs for classification purposes, an important distinction must be made between the training phase and the classification (or validation) phase. This distinction usually does not exist in the case of a standard data mining task.

First, note that when classifying static data, such as determining whether a particular geological site contains oil, commonly a static features vector, such as the type of soil and its mineral content, represents the classified entities.

However, in the case of temporal classification, the entities are represented by multivariate time series, such as (in the case of clinical assessment) longitudinal, co-occurring, multiple laboratory tests, time intervals of various known diagnoses, or time intervals of various treatments, all accumulating over time. Thus, the features mostly consist of the discovered TIRPs.

Second, note as a result of the data representation nature, and the semantics of the TIRP discovery process, the features vector (consisting mostly of TIRPs), representing any particular entity, may vary according to the specific (discovered) TIRP tree that was used to represent the multivariate time series of that entity; the contents of that TIRP tree, in turn, depend on the particular entities used to generate the tree, since within each subset of entities used for the generation process, different TIRPs might pass the frequency threshold.

This insight has three major implications: (1) to the *training phase*, since *the features (TIRPs) may be different for each training set*—thus, when new entities are added to a time-oriented training set, the mining process should be applied again, using *all* of the entities, to produce the correct features (TIRPs), unlike the case of static classification tasks, in which the basic features (e.g., mineral content) are always the same, and are not dependent on the rest of the entities in the same class; (2) to the *classification phase*, since the TIRPs, once discovered using a population of entities, have to be detected (i.e., determined as existing or not) within the (longitudinal) record of a single entity, to assign, for that entity, a value to each feature; and (3) to the *evaluation phase*, since to perform the popular process of cross validation, in which a different portion of the dataset is used each time as a training set and

the other portion is used as a testing set, the mining process that defines *the very features to be used* (even before the processes of feature selection and classifier induction take place) should be repeated for the particular training set used, *in each iteration*.

Thus, unlike the case of static data mining tasks, in which we can use a static matrix of entities versus features, and, during the cross validation process, select each time a different subset of the feature-matrix rows (each row standing for an entity, represented by a vector of values of the *same* set of features), here *the cross validation process must first repeat the mining process* for the selected subset of entities, extract the features (TIRPs) specific to that subset, and construct a new feature matrix for each iteration in the cross validation, *before* the feature selection and learning (induction) phases.

Note that this dynamic state of affairs is qualitatively not unlike the case of text mining, in which, to construct an appropriate vector-space model, the cross validation process might involve re-computing a list of relevant terms, using measures such as term frequencies and inverse (within) document (term) frequencies (TF/IDF); but in the case of temporal classification, we encounter the issue at a significantly higher level of complexity. Note also that this observation is of course relevant to any pattern-based method for classification of multivariate time series, or, indeed, to any other data mining/machine learning task having dynamic features, that is, features that are not predefined, but rather, are defined on the fly upon examination of the training set.

3.4.2 The KarmaLegoSification methodology

To classify an entity, its symbolic time intervals series must be searched for the features, i.e., frequent TIRPs. When mining a single entity, i.e., finding all of the [frequent] TIRPs for a particular subject (e.g., a patient), the notion of minimal vertical support is meaningless, and thus, all of the relevant TIRPs existing within the record of that entity should be discovered. Moreover, in general, not *all* of the discovered TIRPs are relevant to the classification task: In reality, only the TIRPs that appear in the training data, and were thus defined as features, should be searched, as shown in Algorithm 4. Thus, we modified KarmaLego to be applied to a single entity, a process that we refer to as SingleKarmaLego.

Algorithm 4 – SingleKarmaLego

Input:
entity_sti – the single entity's symbolic time intervals records; that is, e.I
T – the enumeration tree of the [full] training set
Output: *Single_T* – an enumerated tree of TIRPs

1. *Single_T* ← SingleKarma(*entity_sti*, *T*, *epsilon*)
2. Foreach $t \in Single\_T^2$ // $Single\_T^2$ is *Single_T* at the 2nd level
3. SingleLego(Single_*T*, *T*, *t*, *2*, *epsilon*)
4. End Foreach
5. return *Single_T*
6. End

However, before describing the process of mining a single entity, we will briefly describe the process of training. Given a set of *E* entities divided into *C* classes, each class is abstracted temporally using the same definitions and mined for TIRPs, using the same minimal vertical support threshold. The result is *C* sets of TIRPs—each of which is related to one class. Finally, the union of all of the discovered TIRPs from each class is used for the feature-matrix representation. This set of TIRPs may include TIRPs that appear in all of the classes, or in

some of them. Although the set of TIRPs can be reduced by performing feature selection, eventually there is a set of TIRPs P, which can be used for the training and later for classification. Note that we do not mine all of the set E (as a single set) for TIRPs, since certain TIRPs characterizing some of the classes might not necessarily pass the frequency threshold when diluted by entities from all other classes.

Thus, in the classification phase, each single entity e has to be mined separately to search for occurrences of the specific TIRPs in P, as will be shown in Algorithm 4—SingleKarmaLego. Note that since the vertical support is meaningless, there is no stopping criterion for the algorithm and any discovered TIRP will be "frequent." However, since we are interested in discovering only the TIRPs that were discovered in the training phase, the algorithm should search for these TIRPs only. In Algorithm 4, the input is a set of TIRPs P, which was discovered in the training set (the unification of all the classes), which is actually the enumeration tree that was discovered. Thus, the goal is to follow the enumeration tree which is represented by P in order to generate and discover all the existing TIRPs in the entity. The mining process is actually the same as was described in KarmaLego, but without considering the minimal vertical support.

Algorithm 5 is similar to Algorithm 2 (Karma), but instead of generating all the possible TIRPs, it searches only for the TIRPs at the first and the second level in the given enumeration tree T that was discovered in the training set phase. After that, for all the TIRPs at the third level that appear in T, the SingleLego algorithm is called to further generate and search the extended TIRPs of this path in the tree T.

Algorithm 5 – SingleKarma
Input:
entity_stis – the symbolic time intervals of the single entity; that is, e.I
T – a [full] set of TIRPs that were discovered in the training phase;
Epsilon – the size of epsilon in the temporal relations;
Output: *Single_T* – an enumerated tree of TIRPs

1. *Single_T* ← Ø
2. Foreach $I^i, I^j \in$ entity_stis ∧ $i<j$
3. $r \leftarrow$ the temporal relation among I^i, I^j given *epsilon*
4. if(r > max_gap)
5. break
6. if $< I_{.sym}^{\ i}, r, I_{.sym}^{\ j} > \in T^2$
7. Index($Single\_T^2, < I_{.sym}^{\ i}, r, I_{.sym}^{\ j} >$) *//indexed as in Algorithm 2*
8. End Foreach
9. return *Single_T*; *//return a tree of up to two levels*
10. End

Algorithm 6 is similar to Algorithm 3 (Lego), but instead of generating all of the frequent TIRPs, it generates the TIRPs that appear in T, which were discovered in the training phase and are required for the classification, and searches for them in the second level (Single_T2) that was created in SingleKarma. The extended TIRP t is an object containing the TIRP properties, such as a vector of symbols, a vector representing the half matrix of the temporal relations, and a list of supporting instances (which is basically a vector of pointers to the sequence of time intervals), as described in more detail elsewhere [20]. The TIRP's object t data structure supports a highly efficient $O(1)$ time query regarding any particular relationship

between two symbols in Single_T^2, without starting the detection of the TIRP within the entity's set of symbolic interval from scratch, as a naïve TIRP detection method might do.

If supporting instances of a TIRP are found, SingleLego searches for its extended TIRPs, as long as they appear in T, recursively. Thus, in each extension, new TIRPs are created using an extended vector of symbols (i.e., adding the next, kth symbol), a vector of temporal relations (i.e., the temporal relations between the candidate symbolic time interval and the previous $(k-1)$ symbolic time intervals), and the list of supporting instances. This list's vertical support can only stay the same, or decrease, as the TIRP is being extended into increasingly specific patterns, although its horizontal support can increase, depending on the particular set of symbols prevalent for each entity.

Algorithm 6 – SingleLego

Input:

Single_T – the enumerated tree of the entity (sent by reference)
T – the enumeration tree of the training set,
t – the extended TIRP;
level – the level of the enumeration in the tree
Output: void

1. Foreach $t^{level+1} \in T$
   (Note that like t, $t^{level+1}$ is an object that stores, for any two symbols, all of the symbolic interval instances satisfying all of the relations between them)
2. $t^{level+1}.insts \leftarrow$Search supporting instances of $t^{level+1}$ in *Single_T*
3. if $|\, t^{level+1}.insts \,| > 0$ *//found at least one instance of this level in T, within Single_T*
4. SingleLego(*Single_T, T,* $t^{level+1}$, *level+1*) *//extend by another level*
5. End Foreach
6. End

The single mining process results with an enumeration tree Single_T, which is a sub-tree of T, containing all of the TIRPs that were discovered in the given entity e. Then, as we show in Sect. 4, a vector of features (TIRPs), having one of the TIRP representations, is created. The vector of features is then used for classification. Note that each TIRP of size $k \geq 2$ is considered a feature, i.e., not just the longest TIRPs, or the leaves of the TIRP tree. Once all of the size $k \geq 2$ TIRPs are detected for a single entity, their horizontal support and mean duration are calculated, to represent them for classification purposes.

3.4.3 Complexity advantages of the SingleKarmaLego algorithm

In this study, we used the SingleKarmaLego algorithm to detect TIRP features discovered by the KarmaLego algorithm within single entities to be classified. Since the core of the current study is on the relationship between the settings of the KLS framework and the resultant classification performance, we could have used in principle any other TIRP detection method without affecting the current study's conclusions. Thus, we will not dwell too much here on the computational advantages of the SingleKarmaLego algorithm. The details of the worst-case and average-case complexity analyses, and of an actual empirical evaluation of the SKL algorithm's runtime behavior versus that of a standard linear pattern-matching algorithm, are out of the scope of the current study and exist elsewhere [21].

However, it is of interest to note here that the SingleKarmaLego algorithm is actually significantly superior, computationally, to a simpler, STD method that attempts to find each feature (i.e., TIRP) within a vector of symbolic time interval instances of the given entity (sorted lexicographically). The advantage can be demonstrated even when searching only

for 2-sized TIRPs, looms much larger as the entity is queried for longer and longer TIRPs within the SingleLego recursive procedure, and can be demonstrated both for the worst-case complexity and for the "average" case [21].

The first and main advantage of the SingleKarmaLego algorithm stems from the one-time, preprocessing indexing step performed in SingleKarma, in which the Single_T^2 tree, representing all of the relations between two symbolic time intervals that appear in the entity, is created, represented as a matrix indexed by the two symbols. The result is a highly efficient structure for the purpose of query and retrieval of 2-sized TIRP instances, which supports, during the TIRP extension process, a very fast ($O(1)$) way of determining whether a given relation indeed exists between symbolic time interval instances of the two symbols.

The second advantage of SingleKarmaLego stems from the fact that, since a tree of TIRPs is created for the single entity's interval instances, adding the lth symbol to an $(l-1)$ sized TIRP exploits the data structure already created for the $(l-1)$ sized TIRP, to quickly check whether the lth symbol can indeed to be added to the list of feature TIRPs detected in the entity's record.

For the reader's convenience, we summarize the results of our detailed analyses and of our empirical evaluation by pointing out the following:

Denote by m the number of TIRPs discovered during the training phase, which need to be detected within the set of n symbolic time intervals of a specific entity to be classified, and by l the number of symbols in a TIRP to be detected within the entity. Denote by K the mean number of time intervals that exist within the max_gap time duration G which we allow between pairs of related time intervals (K and G are specific to each domain or data set).

We can now make the following observations, summarizing a more detailed theoretical and empirical study [21]:

(a) Even for the simplest case of a 2-sized TIRP, a simple sequential-search pattern-matching algorithm requires approximately $O(n^*K^*m^*l^2)$ operations, demonstrating *multiplicative* growth relative to the number of TIRPs to search for in the entity, while SingleKarmaLego requires approximately only $O(n^*K + m^*l)$ operations, demonstrating *additive* growth relative to the number of TIRPs m to find in the entity, a highly significant computational advantage. That advantage looms larger as the number of TIRP-queries, m, increases (in theory, that number might increase exponentially, relative to the number of symbolic concepts, as a result of the training phase), and as the number of symbols per TIRP that we need to find in the entity, l, increases (note that l is really limited only by the maximal number of intervals per entity within the original dataset used for training).

(b) Even when considering just the "practical" question of precisely when is SingleKarmaLego superior to a sequential-search pattern-matching approach in the typical, "average" case, we can show that, given highly reasonable assumptions regarding the input data, SingleKarmaLego is practically *always* superior to a naïve linear search, even when considering the average case, not just the worst-case analysis as in (a), and even for detecting within the single entity only two-sized TIRPs; again, the advantage looms much larger, once we need to find in that entity longer TIRPs.

(c) An empirical evaluation in several different domains confirms our analysis. A typical sample of the empirical results of running the SKL algorithm versus a sequential-search pattern-matching algorithm for the purpose of matching an increasingly large set of TIRPs to a single entity are shown in Figs. 23 and 24 in the Appendix. Shown are the mean times in seconds (on the vertical axis) of running both algorithms on batches of 50, 100, and 200 patient records in the ICU domain, each time with and input number of 40–120

TIRPs to be matched within each patient's interval-based record. The maximal gap was defined in this case as either five or ten domain-specific time units (seconds, in the case of the ICU domain).

The advantages of the SKL algorithm, which requires less computation time for all of the combinations we experimented with, are thus clear, both theoretically and empirically, for any number of entities and TIRPs to detect. Similar results were achieved also for the other two domains we experimented with [21]. Note that the computational advantages offered by the SKL algorithm might be highly significant: 200 entities, as used in our demonstration, is actually a rather small number; the typical number of entities involved might be very large. Examples include the analysis of clinical data of a whole population of patients, in which one might look for the existence of a disorder, whose probability might depend on detecting a set of TIRPs; or when classifying the data accumulating from a large set of mobile devices, for malware detection purposes; etc.

4 Evaluation

The objective of the evaluation of the KLS framework, whose goal is to use the discovered TIRPs for the purpose of classifying multivariate time series, was to assess the feasibility and general effectiveness of using TIRPs for classification and to assess the optimal settings of the method's various parameters (settings), given the three datasets. Thus, our findings regarding several of the best specific settings relevant to these three particular datasets, especially the optimal number of bins and the value of the epsilon parameter, are not intended to be generalized, although they do provide a useful indication for the optimal values of such settings and are helpful in examining more general issues, such as the optimal number of temporal relations to use and the best TIRP representation method[s].

Since the KLS framework includes quite a few potentially meaningful parameters, the number of required experiments was rather large.

The *research questions* posed were related to the varying parameters within the KLS framework (see additional details in the descriptions of the experimental design and of the results):

A. Which *state abstraction method* is the best for classification purposes?
   The goal of this question was to evaluate the EWD and SAX computational state abstraction methods suggested in this paper, in addition to the cut-offs defined by domain experts (knowledge based).
B. What is the *best number of bins* for classification purposes?
   The number of bins, which is the number of states in the state abstraction, determines the level of granularity of the abstraction method. Two options were evaluated: three and four states for the various abstraction methods.
C. Which feature selection measure is the best for classification purposes?
   As part of the training phase and the tabular construction, a feature selection filter was applied, including GainRatio and GAIN with several sets of features for comparison. The first objective was to determine whether the GAIN measure [24] is better than a common feature selection measure, such as GainRatio, in addition to the use of no feature selection (NoFS).
D. Which classifier-induction method is the best for generating a TIRP-based classifier?
   Several classifier-induction methods were used in the experiments, including the standard methods: Random Forest and Naïve Bayes, and the IE_Classifiers proposed by

Patel et al. [24]. For the comparison, only the Binary TIRP representation was used, in order to compare these classifiers to the standard classification algorithms. Note that in this evaluation, our objective was not to determine the optimal classifier-induction method; we simply wanted to compare the IE classifiers to a small number of common, generic induction algorithms, since we thought that standard existing classifiers might well suffice. Once we determined that the Random Forest method was superior to the other methods we used, in most of the first experiment, using a binary representation, it was used in the other two experiments, to evaluate the effect of the number of temporal relations, the various TIRP representations, and the value of the Epsilon parameter.

E. Which set of temporal relations is the best for classification purposes?
Two sets of temporal relations were evaluated in the KarmaLegoS framework, as defined in Definition 1 in Sect. 3.2: The full set of Allen's seven unique temporal relations, and a more abstract set of three temporal relations (BEFORE, OVERLAPS, CONTAINS).

F. Which TIRP representation is the best for classification purposes?
The representation of a TIRP value for an entity in the training and test phases included three options: binary (i.e., whether the TIRP exists at least once in the record) (B), horizontal support (HS) and mean duration (MeanD).

G. Which size of the *epsilon* value is the best for classification purposes?
The size of the epsilon value parameter (see Definition 1 in Sect. 3.2) determines the flexibility of the temporal relations. Five general epsilon values were investigated: epsilon = 0, which is the default crisp version of Allen's temporal relations, versus a larger size epsilon: 1, 3, 5, and 10 time units.

4.1 Datasets

4.1.1 The diabetes dataset

The diabetes dataset, provided as part of collaboration with Clalit Health Services (Israel's largest HMO), contains data on 2004 patients who have type II diabetes. The data were collected each month from 2002 to 2007. The dataset contains six concepts (laboratory values or interventions) recorded over time for each patient: hemoglobin-A1c (HbA1C) values (indicating the patient's mean blood glucose values over the past several months), blood glucose levels, cholesterol values, and several medications that the patients purchased: oral hypoglycemic agents (diabetic medications), cholesterol reducing stations, and beta blockers. The total amount of the diabetic medications was represented in the dataset in terms of an overall defined daily dose (DDD). Knowledge-based state abstraction definitions for abstraction of the raw data laboratory-test measurements into more meaningful concepts were provided by expert physicians from Ben Gurion University's Soroka Medical Center.

For the classification experiments, *determining the gender of the patients was used as the classification target*, since no clinical end points were provided in this particular dataset. Since there were 992 males and 1,012 females, the dataset was quite balanced. Table 1 contains the cut-off definitions used for each state in the case of the raw measurements included in the Diabetes dataset. The rest of the raw data consisted of medications, for each of which a DDD abstraction was defined.

4.1.2 The intensive care unit (ICU) dataset

The ICU dataset contains multivariate time series of patients who underwent cardiac surgery at the Academic Medical Center in Amsterdam, the Netherlands, between April 2002 and

Table 1 Laboratory-measurement data types and their cut-off (discretization) values for the knowledge-based state abstractions defined for each, in the case of the diabetes dataset

State	Glucose	
Blood glucose (mg/dL)		
1	<100	
2	100–125	
3	126–200	
4	>200	
State	HbA1c	
Hemoglobin A1C (%)		
1	<7	
2	7–9	
3	9–10.5	
4	>10.5	
State	LDL	
LDL Cholesterol (mg/dL)		
1	<100	
2	100–130	
3	130–160	
4	>160	
State	HDL-Male	HDL-Female
HDL Cholesterol (mg/dL)		
1	<35	<30
2	35–45	30–40
3	>45	>40

May 2004. Two types of data were measured: static data including details about the patient, such as age, gender, type surgery the patient underwent, whether the patient was mechanically ventilated more than 24 h during her postoperative ICU stay, and high-frequency time series, measured every minute over the first 12 h of the ICU hospitalization.

The data include mean arterial blood pressure (ABPm), central venous pressure (CVP), heart rate (HR), body temperature (TMP), and two ventilator variables, namely fraction inspired oxygen (FiO2) and level of positive end-expiratory pressure (PEEP), and low-frequency time-stamped data, including base excess (BE), cardiac index (CI), creatinine kinase MB (CKMB), and glucose. For the knowledge-based (KB) state abstraction, we used the definitions of Verduijn et al. [37]. In addition, the data were abstracted with computationally unsupervised and supervised abstraction methods. (See the experimental results section).

The dataset contains 664 patients; 196 patients were mechanically ventilated for more than 24 h. Nineteen patients had very few values and were removed. Thus, the experimental dataset included 645 patients of whom 183, or 28 %, were mechanically ventilated for more than 24 h. *The main classification goal was determining whether the patient needed ventilation after 24 h, given the data of the first 12 h.*

4.1.3 The hepatitis dataset

The hepatitis dataset contains the results of laboratory tests performed on patients with hepatitis B or C, who were admitted to Chiba University Hospital in Japan. Hepatitis A, B, and C are viral infections that affect the liver of the patient. Hepatitis B and C chronically inflame the hepatocyte, whereas hepatitis A acutely inflames it. Hepatitis B and C are especially important, because they involve a potential risk of developing liver cirrhosis and/or carcinoma of the liver.

The dataset contained time series data regarding laboratory tests, which were collected at Chiba University hospital in Japan. The subjects included 771 patients of hepatitis B and C who had tests performed between 1982 and 2001. The data included administrative information, such as the patient's demographic data (age and date of birth), pathological classification of the disease, date of biopsy, result of the biopsy, and duration of interferon therapy. Additionally it included the temporal records of blood tests and urinalysis. These tests can be split into two sub-categories, in-hospital and out-hospital test data. In-hospital test data contained the results of 230 types of tests that were performed using the hospital's equipment. Out-hospital test data contain the results of 753 types of tests, including comments of staff members, performed using special equipment at the other facilities. Consequently, the temporal data contained the results of 983 types of tests. We selected eleven variables which were found most frequent (occurring in most of the patients), including Glutamic-Oxaloacetic Transaminase (GOT), Glutamic-Pyruvic Transaminase (GPT), Lactate DeHydrogenase (LDH), TP, ALkaline Phosphatase (ALP), Albumin (ALB), Total BILirubin (T-BIL), Direct BILirubin (D-BIL), Indirect BILirubin (I-BIL), and Uric Acid (UA). Many patients had a limited number of tests, so we focused only on the variables occurring frequently, which included 204 patients who had Hepatitis B and 294 patients who had Hepatitis C. No (KB)-based abstractions were available for that domain, so only automated abstraction methods were used. *The objective was to classify patients as having either Hepatitis B or C.*

4.2 Experimental design

The evaluation of the KLS framework focused on the various parameters of the framework as presented earlier, by performing three key experiments, all using the discovered TIRPs as classification features, which together answer all of our research questions.

Experiment 1 Varying the number of state abstraction bins, type of feature selection method, and type of classification method.

We evaluated three state abstraction methods: knowledge-based (KB), EWD, and SAX (research question A); the number of bins used for automated discretization was either 3 or 4 (research question B). We also wanted to compare the GAIN measure suggested by Patel et al. [24] to the standard GainRatio measure as a feature selection method (research question C). To be able to compare the two measures, we set GAIN to 0.02, as was recommended by Patel, and used GainRatio with the same number of features. Additionally, we used both measures with the same number of 100 top preferred features for comparison. We used the option of NoFS, which we called NoFS, as a baseline. Finally, we also compared the performance of the standard classification algorithms RandomForest and Naïve Bayes, to that of the IE Classifiers (research question D).

Experiment 2 Varying the number of temporal relations and the TIRP representation.

The main research question here focused on the use of temporal relations (research question E): the use of Allen's seven temporal relations versus the use of only the three abstract

relations, as presented in Definition 1. Additionally, we compared the two novel TIRP representations (research question F), HS and MeanD, to the default Binary (existence) representation.

Experiment 3 Varying the epsilon value.

The research question here focused on the influence of the size of the Epsilon on the classification performance (research question G).

To answer the various research questions, we ran a set of evaluation runs with tenfold cross validation. Note that, as explained in Sect. 3.4.1, during the cross validation, in each iteration, the mining phase was repeated on the relevant dataset to extract the relevant features, and the rest of the steps were performed separately. We used a 60, 40 and 60 % minimal vertical support threshold for the Hepatitis, Diabetes, and ICU12h datasets, respectively. Determining the minimal vertical support is an important issue, since having a low threshold will discover a larger number of TIRPs, but might require a much longer runtime. Thus, we used thresholds that will discover TIRPs for all the experimental settings and that will lead to ending the discovery process within a reasonable time.

5 Results

5.1 Experiment 1

The first experiment focused on determining the best temporal abstraction methods (research questions A) and the best number of bins (research question B), as well as comparison of the GainRatio and the GAIN feature selection measures (research question C) and the RandomForest and Naïve Bayes standard classifiers to the IE Classifiers (research question D). To compare the feature selection measures, we used two options: We set GAIN to 0.02, which we termed GAIN002, and used GainRatio with the same number of top selected features (as at GAIN002), which we termed GR002; and we used both measures with the top 100 selected features, which we termed GAIN100 and GR100. To generate a baseline, we ran the evaluation also without feature selection, which we termed NoFS.

To decrease the settings' complexity and isolate the explored settings, we set the other settings to their default values: Temporal relations number = 7; Epsilon value = 0; TIRP representation = Binary; Feature selection = Gain002, GR002, Gain100, GR100 and NoFS; Classifiers = Random Forest, Naïve Bayes, IE_Confidence, IE_Majority. Since we wanted to evaluate also the various state abstraction methods and number of bins, we set them to: Abstraction methods = KB, EWD, and SAX; Number of bins for automated discretization = 3 and 4. We ran the experiment on each of the datasets using tenfold cross validation.

Figures 7, 8 and 9 show the results of experiment 1 on each of the datasets in addressing research questions A and B, *thus, each result is the mean of 200 evaluation runs*; the mean results of all the datasets are shown in Fig. 10, *and each result plotted is based on 600 evaluation runs*. However, since the results are based on three different datasets, please note that Fig. 10 should be considered mostly as a summary; future results based on a larger number of datasets are required for strengthening these conclusions.

In the case of the Diabetes dataset, using the KB abstraction method led to much better performance than using the SAX and EWD methods, while SAX with four states led to the best performance among the automated methods. For the ICU12h dataset, use of the EWD method produced better performance than the use of SAX, for either three or four bins, and use of the KB definitions resulted in a performance similar to using EWD with three bins.

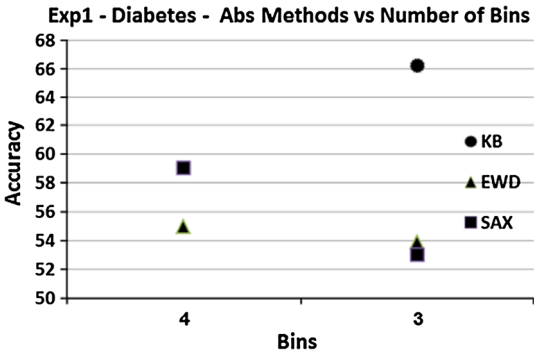

Fig. 7 The KB abstraction performs better than the EWD and SAX, and using four states was better than three, in the case of the Diabetes domain task

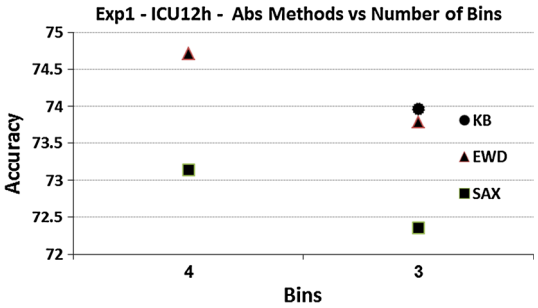

Fig. 8 EWD performed better than SAX, and using four states was better than using three, in the case of the ICU domain task

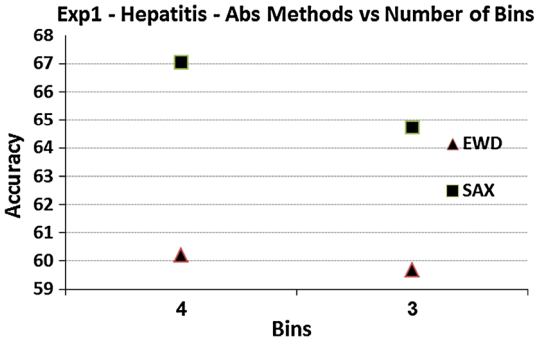

Fig. 9 SAX performed better than the EWD method, and using four states resulted in better performance than using three states, in the case of the Hepatitis domain task

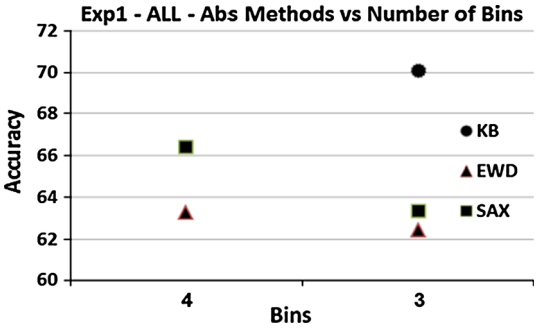

Fig. 10 On average, across all relevant domains and experimental settings, the KB abstraction method led to a superior performance, while SAX performed better than EWD when using either 3 or 4 bins

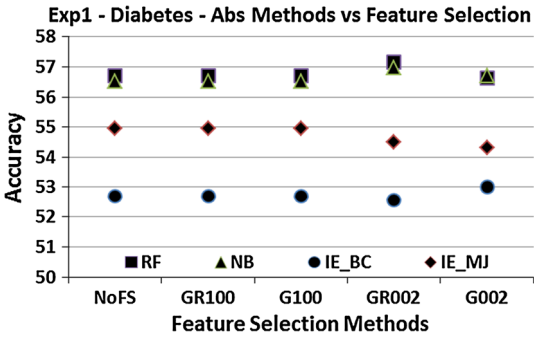

Fig. 11 The Random Forest (RF) and Naïve Bayes (NB) methods performed similarly on the diabetes dataset, outperforming the IE classifiers. The feature selection method (if any) did not have much effect

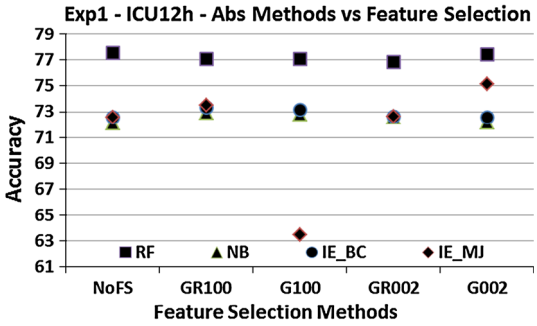

Fig. 12 The Random Forest (RF) method outperformed the other classifiers in the ICU domain. The feature selection methods mostly did not seem to be useful

For the Hepatitis classification task, SAX performed better than the EWD method, and using four states resulted in better performance than using three states.

The mean results averaged across all of the experimental variations (Fig. 10) show that using SAX led to better performance than using EWD, using either three or four bins, while using a KB abstraction (based on two datasets) led to better performance than using the computational methods. Note that each result is the mean of 200 evaluation runs, due to the various runs of the four classifiers and five features selection measures, as detailed in the next results.

Figures 11, 12, 13 and 14 show the results of the classifiers and the feature selection measures of Experiment 1 on each of the datasets addressing the research questions C and D, and the mean results of all the datasets. Each result in Figs. 11, 12, and 13 is the mean of 40 evaluation runs, and each result in Fig. 14 is the mean of 120 evaluation runs. In the Diabetes dataset, the Random Forest and Naïve Bayes induction methods performed similarly, both outperforming the IE Classifiers. The feature selection methods performed similarly, including the NoFS. In the ICU12h task, the Random Forest induction method outperformed the other classifiers. The feature selection methods performed similarly, while the Naïve Bayes method performed worst when using the Gain with the top 100 features. In the Hepatitis domain, the Random Forest method outperformed the other classifiers, and the Naïve Bayes classifier was slightly better than the IE Classifiers. Again, none of the feature selection methods seemed very useful, thus making our original research question a moot one, although using GR100 proved to be the best for the IE Classifiers and the Naïve Bayes methods.

The mean results across all the datasets show that the Random Forest induction method outperformed the rest of the classifiers; and that the feature selection methods were not useful. The low effectiveness of the feature selection can be explained by the relatively low number

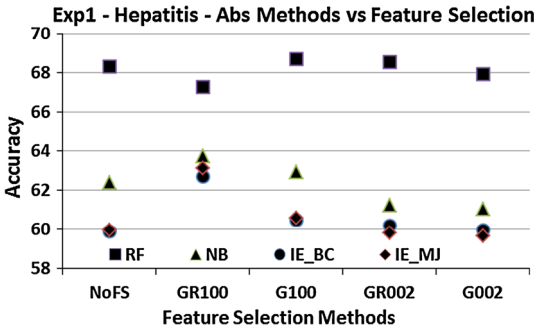

Fig. 13 The Random Forest (RF) method outperformed the other classifiers in the Hepatitis domain. The GR100 feature selection method performed best for the classifiers, except for RF, in which it was slightly less effective

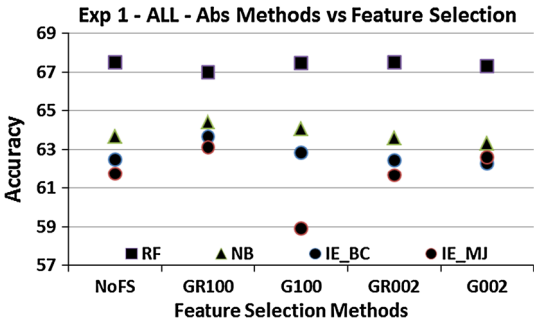

Fig. 14 Mean results across all datasets and settings. The Random Forest (RF) method outperformed the other classifiers, and the feature selection methods did not seem to contribute to the results

of TIRPs (features) discovered originally. However, as stated previously, since the results are based on three different datasets, please note that Fig. 14 should be considered mostly as a summary; future results based on a larger number of datasets are required for strengthening these conclusions.

5.2 Experiment 2

In experiment 2, we wanted to evaluate the performance of the temporal relations sets: Allen's original set of seven temporal relations versus the set of three abstract temporal relations that are proposed in Definition 1 (research question E), and to evaluate the additional two TIRPs (features) representations: the HS (Definition 7) and the MeanD (Definition 9) (research question F) in comparison with the default Binary representation.

The settings of this experiment were as follows: abstraction methods: KB, EWD and SAX; number of bins: 3 and 4 bins; temporal relations: Allen's seven relations (7) and our three abstract temporal relations (3); the epsilon value was set to 0; TIRP representation methods: HS and MeanD, in addition to the Binary representation. Given the results of experiment 1, we used here only the RandomForest classifier, which proved superior to the other induction methods, and used NoFS, since no clear value was demonstrated for any feature selection method.

Figures 15, 16, 17 and 18 show the effect of varying the number of temporal relations versus varying the TIRP representations, given the three different datasets, addressing research questions E and F, respectively. *Each result in Figs.* 15, 16 and 17 *is the mean of 70 evaluation runs and in Fig.* 18 *is the mean of 210 evaluation runs*. The results in the case of the Diabetes classification task show that using three abstract relations outperformed using all seven relations for any type of TIRP representation, which in fact didn't demonstrate any mean-

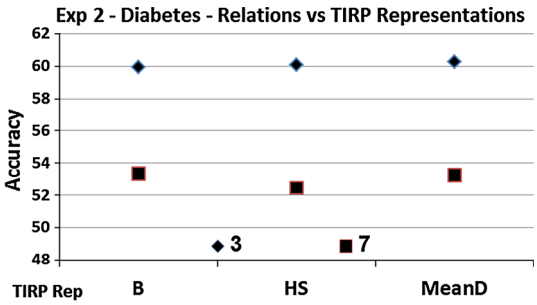

Fig. 15 The use of 3 temporal relations outperformed the use of 7 temporal relations, in the diabetes domain; the MeanD feature representation method was marginally better than the HS and Binary representations

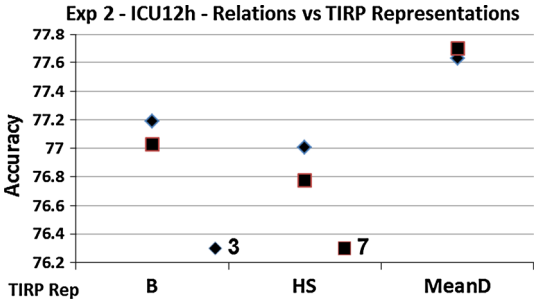

Fig. 16 The use of 3 temporal relations was slightly better than the use of 7 temporal relations, in the case of the ICU domain, while the MeanD representation was better than the Binary one, which was better than using the HS

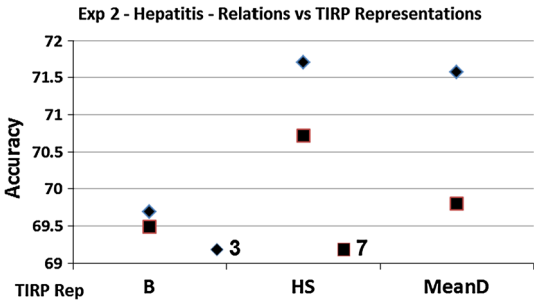

Fig. 17 The use of 3 temporal relations was better than the use of the 7 temporal relations, in the case of the Hepatitis domain, while the HS representation method was best

ingful influence on the performance. In the ICU12h classification task, using three temporal relations led to a better performance and the MeanD representation performed better than the Binary, which was better than the HS. In the Hepatitis classification task, using the set of three temporal relations was better than using the set of seven relations. The MeanD and HS representations performed better than the Binary TIRP representation; the HS representation performed better than MeanD, especially when using seven relations.

The mean results across all of the datasets and settings show in Fig. 18 that using three temporal relations was superior to the use of seven temporal relations and that the MeanD TIRP representation method performed somewhat better than the HS representation, which was somewhat better than the Binary representation. However, as before, since the results are based on three different datasets, please note that Fig. 18 should be considered mostly as a summary; future results based on a larger number of datasets are required for strengthening these conclusions.

Thus, we conclude that the use of three abstract temporal relations is better for the purpose of classification than the use of Allen's seven relations. The MeanD and HS TIRP representations are better than the default Binary representation. These two measures are unique to

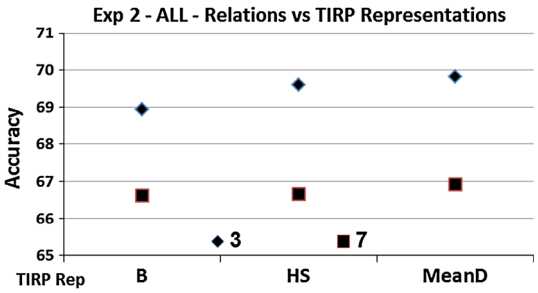

Fig. 18 Mean results across all datasets and settings. The use of 3 was slightly superior to the use of 7 temporal relations, while the MeanD TIRP representation performed somewhat better than HS, which was somewhat better than the Binary

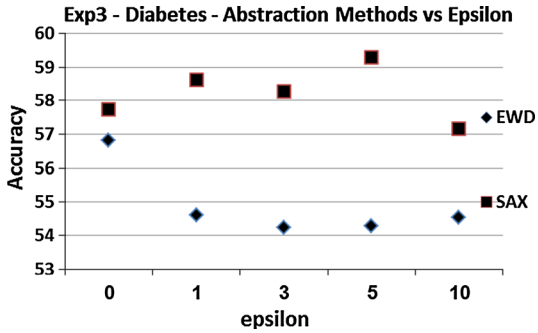

Fig. 19 In the diabetes domain, the use of a larger epsilon usually reduced accuracy when using the EWD discretization method, while usually improving it slightly when using the SAX method

the KarmaLego framework, which, unlike common mining methods, discovers all of the instances of a pattern within the same entity and does not stop at the first instance discovered (thus enabling it to calculate the horizontal support and the mean duration within each entity), which is enabled by its efficient candidate generation and computation.

5.3 Experiment 3

Experiment 3 evaluates the influence of the epsilon value on the classification accuracy (research question G). To isolate that influence, we used the following settings: As abstraction methods we used EWD and SAX, both using 3 bins. We used 3 bins for the analysis, since when using 4 bins, we did not discover a sufficient number of TIRPs for some of the epsilon values. The TIRP representation method was set to MeanD; we used NoFS—that is, no feature selection; and the induction method was Random Forest.

Figures 19, 20, 21 and 22 show the results of Experiment 3, based on the mean of tenfold cross validations (Fig. 22 results are the mean of 30 evaluation runs), addressing research question G. We used for the epsilon values the following values: 0, 5, 10, 20, and 50 for the ICU12h and the Hepatitis datasets. In the Diabetes domain, we used 0 as the default, 1, 2, 3, 5, and 10. We did not use 20 and 50, since the dataset contained a maximum of 60 time points for each patient, but instead we used 1 and 3.

The performance of the abstraction (discretization) methods in this experiment was the same as before; thus, SAX outperformed EWD most of the time, except for a mixed result in the case of the ICU12h dataset. In the Diabetes dataset increasing the Epsilon values improved slightly the performance for the SAX method, but the opposite happened for the EWD. In the ICU12h dataset increasing the Epsilon value decreased slightly the performance for SAX, and did not change much for the EWD, although for Epsilon = 5 it improved. In

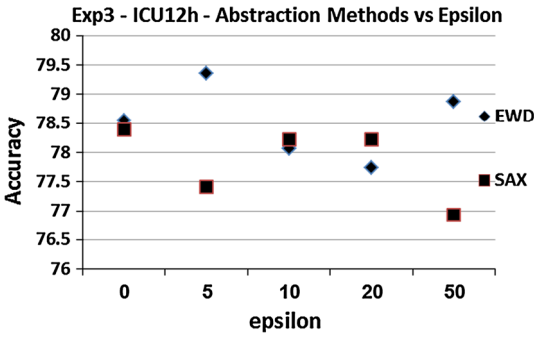

Fig. 20 In the ICU domain, the use of a larger epsilon had an inconsistent effect on accuracy; it usually led to lower performance, in the case of the SAX method

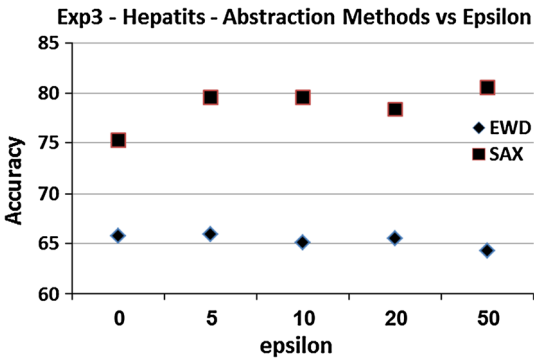

Fig. 21 In the Hepatitis domain, the use of a larger epsilon usually increased accuracy slightly when using the EWD method, while having little effect when using the SAX method

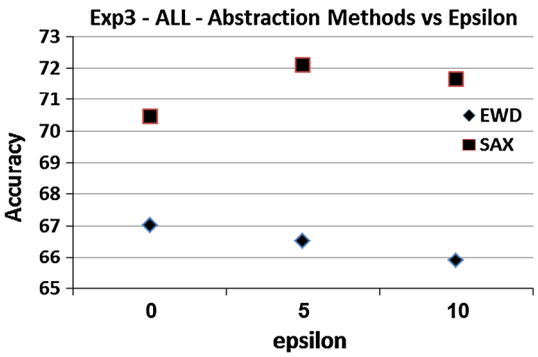

Fig. 22 The mean overall results across all datasets and settings show that using a larger epsilon tends to lead to a slightly better accuracy for SAX, but to a reduced accuracy when using EWD

the Hepatitis dataset, increasing the size of the Epsilon increased the accuracy when using the SAX method, but did not change it for the EWD method.

The mean results of the all the datasets indicate that increasing the value of the Epsilon decreased accuracy when using the EWD discretization method, while slightly increasing when using SAX. Thus, we conclude that increasing the Epsilon value is, in general, not recommended for purposes of an accurate classification.

In addition to the Accuracy measure, we measured the true-positive rate (TPR) and true-negative rate (TNR) for the minority class in each domain. We observed for SAX in the Diabetes dataset a TPR in the range of 57.8–59.5 %, and a TNR in the range of 56.4–59.2 %; in the ICU12h dataset, a TPR in the range of 89.5–93.5 %, and a TNR in the range of 35.9–45.9 %; and in the Hepatitis dataset, a TPR in the range of 85.5–89.7 %, and a TNR in the

range of 60.5–70 %. Across all of datasets, the mean TPR was in the range of 77.6–80.9 % and the mean TNR was in the range of 50.9–58.4 %.

6 Discussion and conclusions

In this study, we have introduced a comprehensive architecture and process, KLS, for classification of multivariate time series, which is based on a temporal abstraction process that can exploit either domain knowledge, when provided, or, when needed, perform on-the-fly discretization, and which introduces several fundamental concepts that define the output of the time intervals mining process.

Mining time interval patterns is computationally highly demanding, but this is true in general for multivariate time series mining. Transforming the time series into time intervals series reduces significantly the amount of data to analyze. The KarmaLego algorithm for fast temporal interval pattern mining is briefly introduced in this paper and appears in more detail elsewhere [20]. The algorithm performs temporal interval pattern mining efficiently, mostly due to its data structure and to its efficient candidate generation, which exploits the transitivity property of temporal relations. In order to avoid TIRPs that include sequences of symbols abstracted from the same temporal variable, in Lego the candidate generation excludes symbols that are from the same temporal variable as the last symbol in the current TIRP. This issue was considered by Batal et al. [3,4], however, further investigation, especially of the semantic aspect of meaningful TIRPs needs to be carried out, as started by Shknevsky et al. [34].

Although, in principle, any algorithm for TIRP detection within a single entity could be used to examine our research questions, the KLS framework uses *SingleKarmaLego* an algorithm for fast TIRPs mining in a single entity that is represented by symbolic time intervals, whose use for this purpose we have introduced in this study. SingleKarmaLego is based on the principles underlying the KarmaLego algorithm for fast TIRPs mining [20]. Since in a single entity, the notion of minimal vertical support is meaningless, the detection task is performed by mining only the TIRPs that were discovered as frequent in the training set and used in the classification model. SingleKarmaLego can be shown to be superior to a Sequential TIRPs Detection (STD) algorithm, both in its worst-case analysis and in its average-case analysis, as well as in an empirical evaluation, as we demonstrate in detail in a separate report [21]. The benefits of the SingleKarmaLego algorithm quickly increase, as either the number of TIRPs to be detected, or the number of entities within which to detect these TIRPs, increase. Thus, we have used the SingleKarmaLego algorithm to examine all of the research questions in this study.

Our evaluation of the KLS framework and its properties was not designed to examine or enhance the accuracy measures per se, but rather, to investigate the feasibility of the overall classification approach, using only TIRPs, and, in particular, to answer the specific research questions that we raised regarding its various settings.

With respect to assessing the best abstraction method (research question A), usually, the SAX discretization method, using either 3 or 4 bins (research question B) and considering all of the test domains, proved superior to the EWD method, which is a popular default, although, when available, the knowledge-based (based on domain knowledge cut-off definitions) abstraction was, on average, quite superior to both, in particular within the diabetes domain. This is a somewhat surprising observation, since it is certainly not obvious that domain experts (clinicians, in this case) necessarily create abstractions that are useful not only for their daily use, such as for continuous monitoring or therapeutic (diagnostic) pur-

poses, but also for (possibly future, or external) classification purposes [37]. However, these results suggest that use of domain-specific-based abstraction should be considered at least as an option, when available.

Our experiences with the various feature selection methods (research question C) have shown no meaningful differences between the standard methods and the measures proposed by Patel et al. [24], whose performance was quite similar to the performance without feature selection. For these particular datasets, the Random Forest induction method proved superior to the other methods we experimented with (research question D), including other methods reported in a similar context of temporal data mining, and was thus the default method for the second and third experiments.

As part of our evaluation, which was performed within three data sets originating from three highly different medical domains (unlike most previous studies, which had typically used just a single dataset), we have quantitatively evaluated the difference between the use of Allen's original basic seven temporal relations, versus the use of an abstract version of three temporal relations that we proposed here (research question E). Note that the proven effectiveness of using only three (abstract) relations greatly reduces the computational complexity of the TIRP enumeration phase in the core KarmaLego algorithm, which is highly (exponentially) sensitive to the number of temporal relations. In future research, one could examine even the effect of using only two abstract relations, such as *Before* and *Overlaps* (i.e., the rest of Allen's relations), as suggested by some authors [4] (better terms might be *Before* and *Coincide*), or focus on the relation *Precedes* (Before, Meets, or Overlaps) versus the rest of Allen's relations, as suggested by other authors [29], etc.

Our original thought was that the discovery of a larger number of TIRPs, or feature, might result in a better performance. This was sometimes true, but only for the Hepatitis dataset; more TIRPs were discovered when SAX was applied with either 3 or 4 bins and the result was a better accuracy than when using EWD, as shown in our mean results analysis. However, for the Diabetes and the ICU datasets, the opposite is true. Higher accuracy was achieved in general with fewer discovered TIRPs (features). Thus, it seems that discovering a larger number of TIRPs does not necessarily result in a higher accuracy; rather, it seems that it is the quality of the TIRPs that matters and that quality seems to be influenced mainly by the abstraction method, as was shown in our analysis. This observation can lead to a very interesting direction for future work, namely, a classification-driven temporal discretization method, in which the performance of a classifier using the abstraction is the main abstraction-performance measure, which we currently examining.

As another part of the evaluation, we have introduced and assessed two novel representation metrics for TIRPs features, which exploit the capability of the KarmaLego framework to compute the number and the mean duration of multiple instances of discovered TIRPs within each time-oriented record, in addition to a Binary, or Boolean (purely existential) representation, for the purpose of classification (research question F). These representation methods were shown to have an added value, when compared to the default Binary representation method. This enhanced representation is achieved without requiring any additional computational cost, since, as pointed out in the comments to the Lego algorithm, to preserve the completeness of the TIRP discovery process, i.e., in order to discover *all* of the different TIRPs within an entity, whether we need one or more instances of each TIRP, *all* of the instances of each TIRP have to be discovered within each entity, in any case.

We also conclude, given yet another aspect of the evaluation, that increasing the Epsilon value (research question G) beyond 0 (the default value) is in general *not* recommended for purposes of classification, in particular when reducing the number of temporal relations to only three (thus already increasing flexibility). This conclusion might alleviate some potential

difficulties, since determining the right Epsilon value might be quite problematic and might in general require several trials for each new dataset. One possibility for explaining our results is that using a larger value for the epsilon parameter decreased the classification performance, since it decreased the number of frequent TIRPs discovered, which in turn, decreased the number of features and thus, eventually, the classification accuracy. Another possibility is that using an Epsilon value, although increasing robustness, has also resulted in somewhat more opaque (ambiguous) relations, thus perhaps losing some information that might be useful for classification purposes, and thus, overall, leading to little or no enhancement in accuracy.

One of the settings of the experiments which, to highlight the other aspects, we did not try to vary, is the minimal vertical support. We would like in future work to experiment with lower levels of minimal vertical support and evaluate their contribution. In addition, we did not use any single symbolic intervals as features (i.e., only TIRPs including two or more symbols). We will examine in the future the effect of including such single symbols as well. In some cases, a single time interval of, say, "high blood pressure" might be sufficient as an important feature.

Overall, we conclude that TIRP-based classification, as performed within the KLS framework, is quite feasible, leads to a reasonable performance, and is potentially highly useful for classification and prediction purposes in time-oriented, multivariate-data domains. The enhanced performance achieved using a knowledge-based (cut-offs) abstraction in the case of the Diabetes dataset shows that better classification might be achievable through the use a better informed discretization method, although not necessarily based on domain knowledge, a direction that we plan to explore in our future work. Such a method is especially crucial in new domains, in which knowledge-based definitions do not necessarily exist. A related problem is the use of TIRPs as features for prediction of outcome events, as was demonstrated by Moskovitch et al. [22] in prediction of clinical procedures in Electronic Health Records.

Acknowledgments The authors wish to thank Marion Verduijn for sharing the ICU12h dataset, and Prof. Avi Porath from the Soroka Academic Medical Center for his assistance regarding the diabetes dataset. For insightful discussions on time intervals mining and classification using TIRPs, we would like to express our thanks to Christos Faloutsos, Christian Freksa, Panagoitis Papapetrou, Fabian Moerchen, Dhaval Patel and Iyad Batel, as well as to Guy Ezra in his help with some of the implementations. The authors also wish to acknowledge the highly useful comments of the anonymous reviewers, which have significantly improved this manuscript. This work was supported in part by grants from Deutsche Telekom Laboratories, HP labs Innovation Research Program

Appendix: Comparing the SingleKarmaLego algorithm to a sequential TIRPs detection algorithm

The theoretical advantages that can be demonstrated through a worst-case and average-case analyses of the SingleKarmaLego (SKL) algorithm, compared to a Sequential TIRPs Detection algorithm, are described in detail elsewhere, as are the complete results of the empirical runtime evaluation [21].

In our full experiment, as described in that study, we compared the runtime of the SKL algorithm to a Sequential TIRPs Detection (STD) algorithm in three very different medical domains. Here, we present just a sample of the results. We start by describing the Sequential TIRPs Detection algorithm we compared SKL to.

Algorithm 7 – Sequential TIRPs Detection

Input:

entity_stis – the symbolic time intervals of the single entity;
tirpsList – list of TIRPs to detect
Epsilon – the size of epsilon in the temporal relations;
max_gap – the maximal time duration allowed in the before temporal relation
Output: *tirps (containing their detected instances)*

```
1. Foreach tirp in tirpsList
2.     Foreach I^i ∈ entitiy_stis
3.         If( I_sym^i == tirp.s[0])
4.             j ← i
5.             tIdx ← 1
6.             instance ← I_sym^i
7.             while(j < |entitiy_stis| && tIdx < tirp.size)
8.                 If(Duration(I_sym^j – I_sym^i) > max_gap);i,j are time-interval indices
9.                     break
10.                If(I_sym^j == tirp.s[tIdx])
11.                    If any relations among I_sym^j and instance.Is ≠ tirp.r
12.                        break
13.                    Else
14.                        instance ← instance ∪ I_sym^j
15.                        If(tIdx == tirp.size)
16.                            tirp.instances ← tirp.insts ∪ instance
17.                            tIdx ← 1
18.                            instance ← I_sym^i
19.                    EndIf
20.            Endwhile
21.        EndIf
22.    EndForeach
23. EndForeach
24. End
```

We first describe the STD algorithm for TIRPs matching; we then show the runtime results on one of the three datasets used in the current study (the ICU dataset).

Algorithm 7 describes a typical Sequential TIRPs Detection algorithm. The algorithm accepts a vector of the single-entity symbolic time intervals (entity_stis) and a list of TIRPs to detect (tirpsList). The algorithm detects all of the instances for each of the TIRPs (line 1). For each TIRP, it starts by going over all of the symbolic time intervals (line 2). If the first symbol of the current TIRP (tirp.s[0]) was detected, a while loop starts to look for the TIRP's next symbols and verifies that their temporal relations with the previous detected symbolic time intervals are the same as in the TIRP temporal relations definition (tirp.r). If the time duration between the *i*-interval and the *j*-interval is larger than max_gap, the search for instances is stopped.

In the full set of experiments, we ran both the SKL algorithm and the Sequential TIRPs Detection methods on all of the three datasets used in the current paper, with a varying number of TIRPs to detect and number of entities to detect, that demonstrate typical condition of evaluation run, as well as give a good estimation of the runtime for a single entity (when dividing the runtime in the number of entities to detect).

Here, we present several typical runtime results for the ICU dataset. The ICU set has a mean of 292 symbolic time intervals in each entity; we show the runtime results when getting as input 40, 60, 80, 100, and 120 TIRPs to detect, and 50, 100, or 200 entities in which to detect these TIRPs, to demonstrate the increasing difference in runtime of the Sequential TIRPs Detection algorithm, which requires longer and longer computation times, in comparison with the SKL algorithm, whose runtime grows only slightly as the number of entities significantly increases.

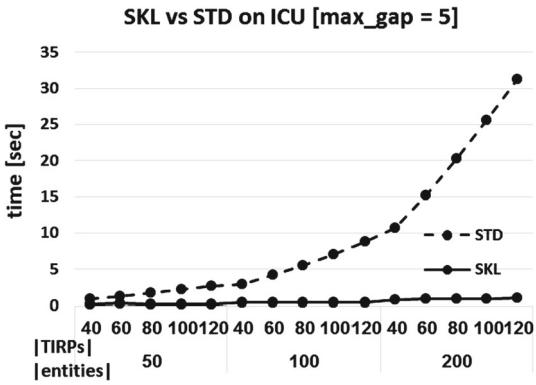

Fig. 23 A runtime comparison of the SingleKarmaLego (SKL) and Sequential TIRP Detection (STD) algorithms to detect a set of TIRPs within multiple given entities and a number of TIRPs to detect, on the ICU dataset. The maximal gap here was 5 domain time units (here, seconds). While SKL's runtime is growing only slightly with the increase in the number of entities and number of TIRPs, the runtime of STD is increasing quickly, especially when applied to 200 entities

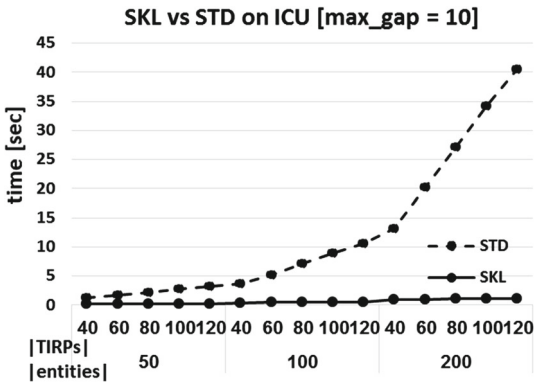

Fig. 24 A runtime comparison of the SingleKarmaLego (SKL) and Sequential TIRP Detection (STD) algorithms to detect a set of TIRPs within multiple given entities and a number of TIRPs to detect, on the ICU dataset. The maximal gap here was 10 domain time units (here, seconds). While SKL's runtime is growing only slightly with the increase in the number of entities and number of TIRPs, the runtime of STD is increasing quickly already when applied to 100 entities. Compared to the max_gap = 5, the STD algorithm is slowing down sooner

Figures 23 and 24 show the runtime comparison of the SingleKarmaLego (SKL) algorithm versus the Sequential TIRPs Detection (STD) algorithm on the ICU dataset, with two sizes of max_gap of 5 and 10 time units in the domain (seconds in the case of the ICU domain). On each max_gap, the algorithms were ran on either 50, 100, or 200 entities, as well as on 40, 60, 80, 100, or 120 TIRPs to detect. Each graph shows the runtime in second on the vertical axis and the horizontal axis shows both the number of TIRPs at the top and the number of entities in the bottom. As can be seen in the following figures, the advantages of the SKL algorithm, which requires less computation time for all of the combinations we experimented with, are clear for any number of entities and TIRPs to detect.

References

1. Allen JF (1983) Maintaining knowledge about temporal intervals. Commun ACM 26(11):832–843
2. Azulay R, Moskovitch R, Stopel D, Verduijn M, de Jonge E, Shahar Y (2007) Temporal discretization of medical time series—a comparative study. In: IDAMAP 2007, Amsterdam, The Netherlands,
3. Batal I, Valizadegan H, Cooper G, Hauskrecht M (2012a) A temporal pattern mining approach for classifying electronic health record data. ACM Transaction on Intelligent Systems and Technology (ACM TIST), Special Issue on Health Informatics
4. Batal I, Fradkin D, Harrison J, Moerchen F, Hauskrecht M (2012b) Mining recent temporal patterns for event detection in multivariate time series data. In: Proceedings of knowledge discovery and data mining (KDD), Beijing, China

5. Höppner F (2001) Learning temporal rules from state sequences. In: Proceedings of IJCAI Workshop on Learning from Temporal and Spatial Data (WLTSD-01), Seattle, USA, pp 25–31
6. Höppner F (2002) Time series abstraction methods—a survey workshop on knowledge discovery in databases, Dortmund
7. Hu B, Chen Y, Keogh E (2013) Time series classification under more realistic assumptions. In: Proceedings of SIAM data mining
8. Kam PS, Fu AWC (2000) Discovering temporal patterns for interval based events. In: Proceedings DaWaK-00
9. Lin J, Keogh E, Lonardi S, Chiu B (2003) A symbolic representation of time series with implications for streaming algorithms. In: 8th ACM SIGMOD DMKD workshop
10. Mörchen F, Ultsch A (2005) Optimizing time series discretization for knowledge discovery. In: Proceedings of the Eleventh ACM SIGKDD international conference on knowledge discovery in data mining, Chicago, Illinois, pp 660–665
11. Mörchen F (2006) Algorithms for time series knowledge mining. In: Proceedings of KDD
12. Moerchen F (2006) A better tool than Allen's relations for expressing temporal knowledge in interval data. In: Workshop on temporal data mining
13. Moerchen F, Fradkin D (2010) Robust mining of time intervals with semi-interval partial order patterns. In: Proceedings of SIAM data mining
14. Moskovitch R, Hessing A, Shahar Y (2004) Vaidurya-a concept-based, context-sensitive search engine for clinical guidelines. Medinfo 11:140–144
15. Moskovitch R, Stopel D, Verduijn M, Peek N, de Jonge E, Shahar Y (2007) Analysis of ICU patients using the time series knowledge mining method. In: IDAMAP 2007, Amsterdam, The Netherlands
16. Moskovitch R, Gus I, Pluderman S, Stopel D, Glezer C, Shahar Y, Elovici Y (2007) Detection of unknown computer worms activity based on computer behavior using data mining. In: IEEE Symposiyum on Computational Intelligence and Data Mining, Honolulu, Hawaii
17. Moskovitch R, Shahar Y (2009) Vaidurya: a multiple-ontology, concept-based, context-sensitive clinical-guideline search engine. J Biomed Inform 42(1):11–21
18. Moskovitch R, Shahar Y (2009) Medical temporal-knowledge discovery via temporal abstraction. In: AMIA 2009, San Francisco, USA
19. Moskovitch R, Peek N, Shahar Y (2009) Classification of ICU patients via temporal abstraction and temporal patterns mining. In: IDAMAP, Verona, Italy
20. Moskovitch R, Shahar Y (2013) Fast time intervals mining using transitivity of temporal relations. Knowl Inf Syst. doi:10.1007/s10115-013-0707-x
21. Moskovitch R, Shahar Y (2014) Fast detection of time intervals related patterns, TechReport 11/14. Ben Gurion University, Beer Sheva, Israel
22. Moskovitch R, Walsh C, Hripcsak G, Tatonetti N (2014) Prediction of biomedical events via time intervals mining. In: Proceedings of ACM SIGKDD workshop on connected health at big data Era (BigCHat2014), New York, US
23. Papapetrou P, Kollios G, Sclaroff S, Gunopulos D (2009) Mining frequent arrangements of temporal intervals. Knowl Inf Syst 21(2):133–171
24. Patel D, Hsu W, Lee ML (2008) Mining relationships among interval-based events for classification. In: Proceedings of the 2008 ACM SIGMOD international conference on management of data, pp 393–404
25. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC (2001) PrefixSpan: mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th international conference data engineering (ICDE '01)
26. Rabiner LR (1989) A tutorial on Hidden Markov Models and selected applications in speech recognition. Proc IEEE 77(2):257–286
27. Ratanamahatana C, Keogh EJ (2005) Three myths about dynamic time warping data mining. In: Proceedings of SIAM data mining
28. Roddick J, Spiliopoulou M (2002) A survey of temporal knowledge discovery paradigms and methods. IEEE Trans Knowl Data Eng 4(14):750–767
29. Sacchi L, Larizza C, Combi C, Bellazi R (2007) Data mining with temporal abstractions: learning rules from time series. Data Mining Knowl Discov 15(2):217–247
30. Shahar Y (1997) A framework for knowledge-based temporal abstraction. Artif Intell 90(1–2):79–133
31. Shahar Y (1998) Dynamic temporal interpretation contexts for temporal abstraction. Ann Math Artif Intell 22(1–2):159–192
32. Shahar Y (1999) Knowledge-based temporal interpolation. J Exp Theor, Artif Intell 11:102–111
33. Shahar Y, Chen H, Stites D, Basso L, Kaizer H, Wilson D, Musen MA (1999) Semiautomated acquisition of clinical temporal-abstraction knowledge. J Am Med Inform Assoc 6(6):494–511

34. Shknevsky A, Moskovitch R, Shahar Y (2014) Semantic considerations in time intervals mining. In: Proceedings of ACM SIGKDD workshop on connected health at big data Era (BigCHat2014), New York, US
35. Stopel D, Boger Z, Moskovitch R, Shahar Y, Elovici Y (2006a) Application of artificial neural networks techniques to computer worm detection. In: International joint conference on neural networks, pp 2362–2369
36. Stopel D, Boger Z, Moskovitch R, Shahar Y, Elovici Y (2006b) Improving worm detection with artificial neural networks through feature selection and temporal analysis techniques. In: Proceedings of the third international conference on neural networks, Barcelona
37. Verduijn M, Sacchi L, Peek N, Bellazi R, de Jonge E, de Mol B (2007) Temporal abstraction for feature extraction: a comparative case study in prediction from intensive care monitoring data. Artif Intell Med 41:112
38. Villafane R, Hua K, Tran D, Maulik B (2000) Knowledge discovery from time series of interval events. J Intell Inf Syst 15(1):71–89
39. Winarko E, Roddick J (2007) Armada—an algorithm for discovering richer relative temporal association rules from interval-based data. Data Knowl Eng 1(63):76–90
40. Wu S, Chen Y (2007) Mining non-ambiguous temporal patterns for interval-based events. IEEE Trans Knowl Data Eng 19(6)742–758

Robert Moskovitch holds a B.Sc., M.Sc., and a Ph.D. in Information Systems Engineering from Ben Gurion University, and is currently a post doctoral research scientist at the Department of Biomedical Informatics at Columbia University. Prior to that, he headed several Research and Development projects in Information Security at Deutsche Telekom Innovation Laboratories at BGU. He has served on several journal editorial boards, as well as on program committees of several conferences and workshops in Biomedical Informatics and in Information Security. He published more than fifty peer reviewed papers in leading journals and conferences, several of which had won best-paper awards.

Yuval Shahar is a professor and previous chair of BGU's Information Systems Engineering department, and the Josef Erteschik Chair in Information Systems Engineering. He holds a B.Sc. and an M.D. (Hebrew University), an M.Sc. in computer science (Yale University), and a Ph.D. in Medical Information Sciences (Stanford University). After a decade at Stanford as a researcher and faculty member (Medicine and Computer Science), he has joined BGU in 2000 to found and head its Medical Informatics Research Center. His research focuses on temporal reasoning, temporal data mining, therapy planning, information visualization, knowledge acquisition, knowledge representation, and decision analysis, applied to biomedicine, homeland security, and information security. He published more than 200 papers. Among multiple awards, Prof. Shahar was granted in 1995 the NIH 5-year FIRST career award; in 2005 an IBM Faculty Award, and in 2008 an HP Worldwide Innovation Program award. He was elected in 2005 as an International Fellow of the *American College of Medical Informatics* (ACMI).