

Temporal Patterns Discovery from Multivariate Time Series via Temporal Abstraction and Time-Interval Mining

Robert Moskovitch^{1,2}, Yuval Shahr¹

¹ Department of Information Systems Engineering, Ben Gurion University, Israel.

² Telekom Innovation Laboratories at Ben Gurion University, Ben Gurion University, Israel.
{robertmo,yshahr}@bgu.ac.il

Abstract. The increasing availability of temporal multivariate data provides an exceptional opportunity to understand better various domains. However, temporal variables can be measured as time-points in different frequencies and varying gaps, or time-intervals. To overcome this problem, we propose to apply temporal abstraction, which transforms raw time-point series into symbolic intervals series, which can be mined to discover temporal patterns. We introduce a framework for multivariate time series analysis via temporal abstraction, including KarmaLego – a fast algorithm for discovery of frequent time interval related patterns, which extends temporal patterns directly, and exploits the transitivity property of temporal relations for efficient candidate generation. Furthermore, we refer to the lack of definition of time interval duration in the common non ambiguous definition of time interval patterns, for which we propose to pre-cluster the symbolic time intervals. We present a rigorous comparison of the KarmaLego runtime performance to several existing temporal-pattern mining methods, which demonstrates it to be significantly faster, especially with large datasets and low levels of minimal vertical support. Finally, evaluating the effects of pre-clustering by interval duration, showed that it leads to a decrease in the time intervals duration variance, accompanied by a corresponding decrease in the number of discovered patterns.

Keywords: Temporal Knowledge Discovery, Temporal Abstraction, Time Intervals Mining, Frequent Pattern Mining.

1 Introduction

The increasing use and availability of longitudinal electronic data presents a significant opportunity to discover new knowledge from multivariate, time-oriented data, by using various data mining methods. In particular, discovery of frequent temporal patterns has significant potential benefits. Thus, in the medical domain, it might lead to the *clustering* of patients who have a similar temporal pattern of interaction among the disease, a set of medications, and certain symptoms. In other cases, it can lead to the *prediction* of certain outcomes, given certain temporal patterns in the patient's current longitudinal disease course. In the information-security domain, it might enable *classification* of hardware devices into infected and non-infected, by their temporal behavior.

When analyzing multivariate time series data, and in particular, when attempting to discover new, frequent temporal patterns within the data, consideration of the temporal dimension, and in particular of the *temporal relations* that hold among various concepts, is very important, and requires explicit representation [9].

However, temporal data include not only time-stamped raw data, or time *points* (e.g., hemoglobin value of 9.3 gr/100cc, at 9:05 am, on July 17th, 1998)), 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., two weeks of *moderate anemia* or of *Grade II liver dysfunction*).

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 figure 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 the medication-administration period, 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.

Thus, discovering frequent temporal patterns in multivariate temporal data, which can appear at varying frequencies, can benefit significantly from a preprocessing phase of converting the raw time-stamped data into a uniform format of a series of *interval-based concepts*, and then discovering frequent patterns of *temporal relations* amongst the intervals.

Thus, in order to analyze multivariate time series datasets having various forms of time stamped data, and specifically for purposes of temporal patterns discovery, 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 Section 2, there are several approaches to the task of converting time-stamped raw data into temporal intervals, typically at a higher level of conceptual abstraction, which we refer to as temporal abstraction; some exploit context-sensitive knowledge acquired from human experts [21], others are purely automatic [3,6,11].

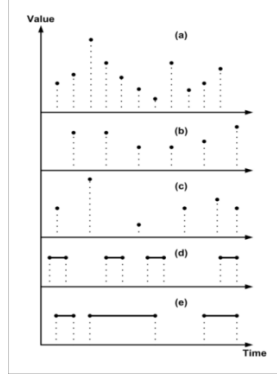


Fig 1. Time-point (a, b, c) and time-interval data (d, e).

Whatever the approach, we then need an effective method for the discovery of frequent temporal relations among the symbolic time intervals, which is the main focus of the current paper. We refer to the resulting knowledge regarding time-oriented concepts and their temporal relationships (patterns) as *temporal knowledge*.

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 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 Section 2). Note that raw-data interval-based concepts such as "10 Days of Penicilin administration" might simply remain at the same level of abstraction.

Although there is a tradeoff involved, due to the potential loss of a finer level of resolution, we believe that the mining of interval-based abstractions might be potentially more fruitful and the results more expressive (and explainable to a domain expert) than the analysis of a much larger, noisy, set of raw data.

The framework that is at the focus of this paper is presented in the general block diagram shown in figure 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 or on other computational means, and are transformed into symbolic-value time intervals (e.g., *Moderate-Anemia* from t_1 to t_2). Then, the symbolic-value time intervals are mined using the KarmaLego algorithm, which we introduce in Section 3 in detail, to discover frequently repeating temporal patterns.

As we explain in Section 3, the discovery process generates a tree of *Temporal Interval Relation Patterns (TIRP)*; for each symbol, each branch represents a temporal pattern discovered in the data with a sufficient predefined frequency, which starts with the symbolic-value concept represented at the root of the tree.

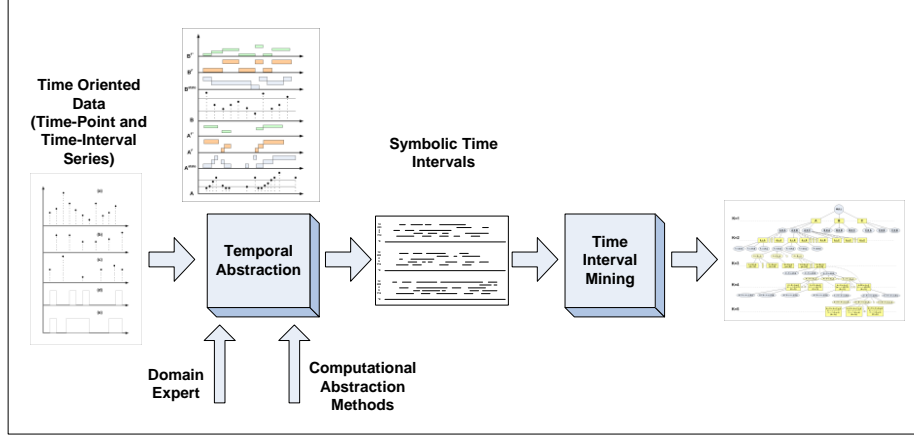


Fig 2. The raw-data time-point and time-interval series are abstracted into a uniform format of [abstract] symbolic time intervals using domain knowledge or specialized computational means. The symbolic time intervals are then mined and a tree of enumerated temporal patterns is generated.

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 automatically discovered temporal patterns for meaningful knowledge, using the KarmaLegoV visualization tool [14].
- *Temporal clustering*, in which each temporal pattern is considered as a cluster of entities (e.g., patients, mobile devices) who have similar temporal behavior (e.g., a similar disease course or malfunction history)
- *Prediction rules*, which are explicitly extracted based on the discovered temporal patterns and the transition probabilities between the components of the patterns.
- *Classification*, in which the discovered temporal patterns are used as features for a classification task, such as for implicit prediction of future outcomes using various data-mining methods [13,17]. (The prediction and classification tasks are outside of the scope of the current paper, and are discussed elsewhere).

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

1. Describing a *comprehensive architecture and process for temporal data mining* that is based on a temporal-abstraction process, and that introduces several fundamental concepts that define the output of the temporal time intervals mining process;
2. Demonstrating the use of an *extended version of Allen's [1] temporal relations*, which introduces the use of flexible, more robust, fuzzy constraints to define temporal relations.
3. Introducing *KarmaLego - an algorithm for fast TIRPs mining*, which *directly extends TIRPs* during the candidate generation phase, and *exploits fully the temporal relations transitivity properties*, to make the candidate generation process significantly more efficient;

4. Supporting the discovery of potentially more useful (for the various applications described above) TIRPs, through a *pre-clustering* of the *time intervals*, such as by duration, resulting in subsets of temporal patterns that are smaller, but more homogenous;
5. *Evaluating rigorously the resulting framework* in several different time-oriented domains.

The rest of this paper is organized as follows: We start by introducing briefly, in the Background (Section 2), the concepts of temporal data mining, temporal abstraction, temporal discretization, and temporal interval-based mining. We then present in the Methods (Section 3) a fast, time-interval mining method, called KarmaLego, and show how it can be used to discover temporal knowledge. We follow by presenting in Section 4 a detailed runtime evaluation of the KarmaLego algorithm compared to existing methods.

Finally, we discuss the implications of the KarmaLego process for a nonsupervised clustering of the mined population of entities, by different temporal pathways, and for a supervised classification of multivariate temporal data into different target concepts, such as, in the case of medical domains, meaningful clinical outcomes.

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 amongst 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 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 [19].

2.2 Temporal Abstraction

Temporal abstraction (TA) is the segmentation and/or aggregation of a time-point series into a succinct, symbolic time-interval series representation, suitable for inspection by a human decision-maker, or for data mining.

Figure 3 illustrates in a simplified fashion a process of *temporal abstraction*, in which time-point data are transformed into time-interval data. Time series B is presented. This process introduces certain complications that will be discussed below. Two types of abstractions are presented: *state* and *gradient* abstractions (other types exist, such as *rate* and *trend* [21]).

State abstraction is actually a type of a discretization process, in which the values of one or more variables that hold over the same time point are classified into a symbolic *state* value (e.g., High) according to the state abstraction's cutoff definitions (or, in general, a state-abstraction function). In the example shown in Figure 3, three symbolic state values are defined, based on the cutoffs illustrated by the lines along the time axis: Low, Medium and High. Later, adjacent time points having the same state value are concatenated into a longer time interval.

The second temporal-abstraction type illustrated here is a *gradient abstraction*, in which the first derivative of each vector of consecutive time point values is classified into the values Increasing (I), when it is sufficiently positive, Decreasing (D), when it is sufficiently negative, and otherwise Stable (S), using a *significant-change* cutoff value.

All abstractions are generated in a manner sensitive to the context, such as being a male or a female [22]. Note also that, unlike the relatively complete series of Figure 4, the abstracted time interval series might in general have several gaps. However, using domain knowledge, adjacent time intervals might or might not be concatenated into longer intervals as part of the temporal-abstraction process, assuming that the same symbolic abstraction type and value holds for both, depending on the values of the data or of the symbolic abstractions derived from it, the context, and the duration of the gap.

This *interpolation* operation and the resultant concatenation of the points or intervals into longer intervals is determined during the temporal-abstraction process, based on context-sensitive domain interpolation knowledge. In the medical domain, for example, it might consider the age, gender, underlying disorder, medication administration, etc., and might be represented explicitly within a domain-specific knowledge base [23]. (Note that interpolation is essential for initially concatenating points into intervals, to produce gradient and trend abstractions, for example).

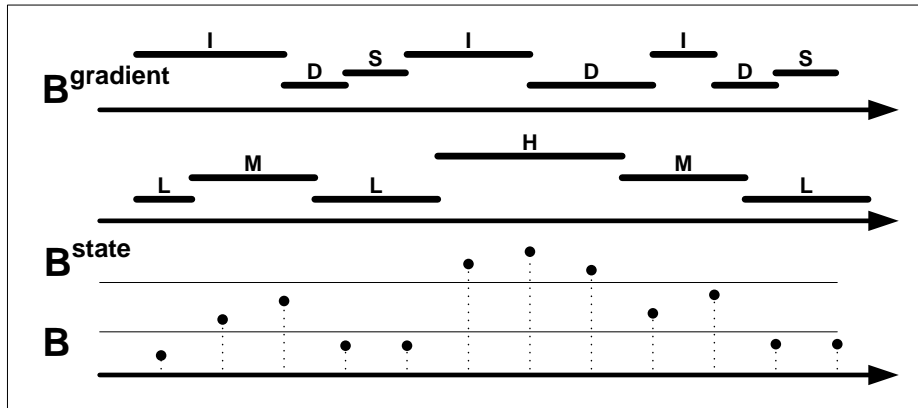


Fig 3: A series of time-stamped data (a time series) is abstracted into a state abstraction that has three discrete values, Low (L), Medium (M), and High (H); and into a gradient abstraction (the sign of the first derivative), which has the values Increasing (I), Decreasing (D), and Stable (S).

Significant efforts to solve the temporal-abstraction task have been made, such as by capitalizing on formal knowledge-based temporal-reasoning methods [21]. In the medical domain, for example, there are multiple cutoff definitions for the values of the state abstractions of variables such as blood pressure or temperature that are used on daily practice for diagnosis purposes.

However, to apply these methods, a certain amount of knowledge about the time-oriented (temporal) concepts (such as meaningful cut-off values) must first be acquired. Acquiring such knowledge from experts often encounters serious difficulties, since experts are not necessarily used to explicitly represent their knowledge in terms of complex patterns involving multivariate time-oriented data. Furthermore, the experts' suggested discretizations might be relevant mainly to the task of *interpretation* of the current data, such as for the purpose of treatment of a patient, or diagnosing the state of a potentially-infected (by malware) computer, and less relevant to the task of frequent patterns discovery, *clustering* the data or of *predicting* future outcomes from them. Thus, we need to consider also additional options for performing the TA process, such as automated discretization of time-oriented data, or *temporal discretization*.

2.3 Temporal Discretization

Although the knowledge-based TA approach is very useful for the discovery of meaningful patterns, based on a domain knowledge base, especially in intensive-knowledge domains such as medicine, it might be less effective, as explained above, for other tasks, such as classification, clustering, and prediction. Moreover, unlike gradient and rate abstractions, which can often be computed and defined mathematically, state-abstraction definitions are not always provided by a domain expert, either because a state abstraction of the variable is not often used in that domain, or because there is no standard discretization for the variable.

In the light of this difficulty, we can consider the option of an automated analysis of longitudinal records and the discovery of knowledge within them through an automated process of discretization of the concepts' values; indeed, it is the default option that we explore in the current study when insufficient domain knowledge exists, especially in nonmedical domains.

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 Hoppner [5]. 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)* [10] (which focuses on a statistical discretization of the values) and *Persist* [11] (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 [3] and eventually for the purpose of classification [15].

In the evaluations performed in the current study, we have used interval-based abstract concepts generated through the use of both knowledge-based temporal abstraction and automated temporal discretization (mostly using equal-width discretization methods).

2.4 Allen's Temporal Relations and Transition Tables

Allen formulated a finite set of 13 temporal relations between a pair of time intervals. The set includes: *before*, *meets*, *overlaps*, *starts*, *during*, *finishes*, and their corresponding inverse relations *after*, *met-by*, *overlapped-by*, *started-by*, *contains*, *finished-by*; and *equals* [1]. One might consider Allen's relations as being composed of seven basic relations, six of which have an inverse (*equals* is its own inverse).

Allen's temporal relations were introduced for the purpose of temporal reasoning, such as for planning and narrative understanding, and since then were used for many purposes and were also further extended [4]. Allen also introduced the use of temporal-relations *transition tables* for reasoning purposes, which was later generalized by Freksa by considering the option of having only semi-intervals [4]. Freksa introduced the concept of similarity among the thirteen relations, called the *conceptual neighborhood*, suggesting a more visual organization of the relations.

Figure 4 shows four examples of reasoning about temporal relations based on their transitivity property. In all the cases there are three time intervals A , B and C . Given the temporal relations among A and B - $r^{A,B}$, and among B and C - $r^{B,C}$, we would like to reason about the possible relations among A and C - $r^{A,C}$.

In the first case, (1), $r^{A,B}$ is *before* ($<$) and the $r^{B,C}$ is *before* ($<$), thus, the only optional temporal relation for $r^{A,C}$ can be *before* ($<$). In the second case (2) $r^{A,B}$ is *overlap* (o) and $r^{B,C}$ is *before* ($<$), thus $r^{A,C}$ is *before* ($<$) too. In the third case (3) $r^{A,B}$ is *meet* (m) and $r^{B,C}$ is *equal* (e) and thus the only optional temporal relation which can be for $r^{A,C}$ is *meet* (m). Thus, in this scenario $r^{A,C}$ can be *before* ($<$), as it is actually in this illustration but it could be also *meet* (m) or *overlap* (o).

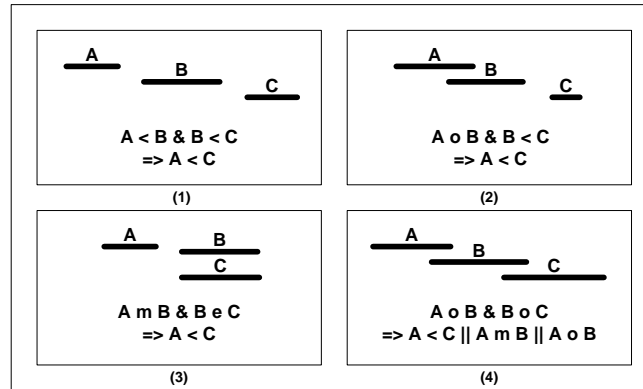


Fig 4. Four cases of temporal transitivity, illustrating how the temporal relation $r^{A,C}$ can be derived, based on $r^{A,B}$ and $r^{B,C}$.

In Allen [1] and in Freksa [4] transition tables are presented in which, given three time intervals and two temporal relations (among the first and second, and the second and third), the disjunction of the optional temporal relations among the first and third time intervals are stored. As we show in Section 3, one of KarmaLego's most novel contributions is in the use of transition tables for efficiently generating TIRP candidates.

2.5 Mining Time-Intervals

As noted in the introduction, our methodology focuses on the mining of interval-based (mostly abstract) concepts, and on the discovery of frequent temporal relationships among them, such as "three weeks of moderately high blood glucose, followed by five weeks of high-dose intermediate-acting insulin administration", rather than on raw, time-stamped data such as "blood glucose of 153 at 9:00 am on Feb 17th, 2011, and administration of NPH insulin at 4pm on Feb 19th, 2011".

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]. One of the earliest studies in the area is that of Villafane et al. [26], 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. Their language covers the concepts of coincidence, synchronicity and order based on Allen's temporal relations. Their patterns are defined by the temporal relation of the extended frequent patterns with the new symbolic time interval, called A1 patterns. However, A1 patterns are ambiguous, since the temporal relations among the components of a pattern are undefined, except for the 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 pair-wise relations within a k-intervals pattern. Using Allen's relations, Höppner [5,6] introduced a method inspired by association rules mining (making it somewhat inefficient) to mine rules in symbolic time interval sequences. The search space is restricted by using a sliding window.

In the rest of this paper, we shall refer to a conjunction of temporal relations between pairs of intervals as a *Time Interval Related Pattern (TIRP)*. The formal definition of a TIRP appears in Section 3 (Definition 5). Thus, Höppner's method essentially discovered TIRPs. Unlike Höppner's naïve mining method, Papapetrou et al. [16] propose two approaches for generating the TIRP tree, using Breath First Search (BFS), in which the enumeration is made at each level before proceeding to the next level, Depth First Search (DFS), in which a greedy approach is used and each path is enumerated up to the leaves, and a hybrid approach, which combines the BFS and DFS methods, inspired by the *Sequential Pattern Mining Algorithm (SPAM)* [2] mining method. The BFS is initially used to discover the first two sized TIRPs in the mined database. At deeper levels, the search for the pair-wise relations is made at the second level of a discovered tree (this level has 2-sized TIRPs). A hybrid approach (H-DFS) was shown to be the fastest of the three. 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. To make the temporal relations more flexible, Papapetrou introduced an *epsilon threshold*, used only for a subset of the temporal relations, including meet, matches and follows relations.

As we show in Section 3, the KarmaLego framework is inspired by the H-DFS approach. In Section 3.6, we show how Papapetrou et al.'s enumeration method can be made more efficient with respect to the specific step of extending a given TIRP. ARMADA, by Winarko and Roddick [27], is a relatively recent projection-based efficient time interval mining algorithm that uses a candidate generation and mining iterative approach. Wu et al. [28] proposed TPrefixSpan, which is a modification of the PrefixSpan sequential mining algorithm [18] for mining non-ambiguous temporal patterns from interval based events. TPrefixSpan represents the time intervals based on the start time and end time events and discovered frequent sequences of the start-time and end-time events from the projected database, although it does not prevent multiple candidates of the same patterns from being generated.

Patel [17] 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 (see Section 3.6). While several methods for non-ambiguous time intervals mining were proposed independently, not much runtime comparisons were made. Patel et al. [17] had compared their method runtime to TPrefixSpan [28], [16] and found their method to be faster. In section 4, we compare KarmaLego's runtime performance also to that of IEMiner [17], ARMADA [27], and H-DFS [16].

Mörchen [12] proposed an alternative to Allen's relations-based methods, in which time intervals are mined to discover partially ordered coinciding symbolic time intervals. Sacchi et al [20] have used abstracted time series to find temporal association rules by generalizing several of Allen's relations into a relation called PRECEDES, which mainly included the temporal relation *before*.

In the following section, we introduce the KarmaLego algorithm, designed to mine temporal intervals using all of Allen's relations, all which are defined in a robust (flexible) fashion. Mining intervals using all Allen's temporal relations is very challenging from a computational perspective. Among other means, the KarmaLego mining method extends TIRPs directly and exploits the transitivity of the temporal relations for pruning in order to efficiently generate all relevant candidate TIRPs.

3 Methods

3.1 KarmaLego – Fast Time-Intervals Mining

The discovery of *Time Interval Related Patterns (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 5 symbolic time intervals, such as in figure 5, with all possible temporal relations among them, requires in theory generating up to $7^{\frac{(5^2 - 5)}{2}} = 7^{10} = 282,475,249$ candidate TIRPs. In general, given a TIRP having k time intervals, we will have up to $7^{\frac{(k^2 - k)}{2}}$ candidate TIRPs.

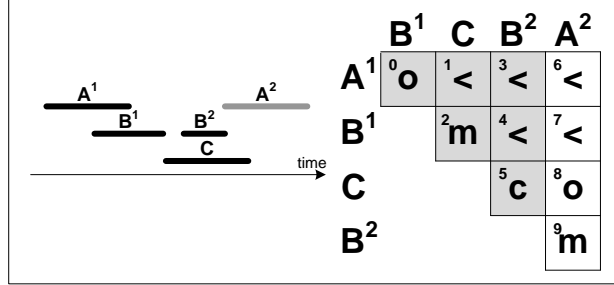


Fig 5. An example of a Time-Interval Related Pattern (TIRP), represented by a sequence of five lexicographically ordered symbolic time intervals and all of their pair-wise temporal relations.

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

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 shown in Figure 6.

Furthermore, to reduce ambiguity, we also limit the *before* temporal relation by a maximal allowed gap (see figure 6), as proposed by Winarko and Roddick [27]. (Note that in general, this gap might depend on the domain and even on the concepts involved).

To define formally the problem of mining time intervals and the solution proposed to it, we first present several basic definitions. These definitions will be used in the description of the methods.

3.2. Definitions

Definition 1. 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.

Given two time-points t_1 and t_2 :

$$t_1 =^\epsilon t_2 \text{ iff } |t_2 - t_1| \leq \epsilon$$

and

$$t_1 <^\epsilon t_2 \text{ iff } t_2 - t_1 > \epsilon.$$

Based on the two relations $=^\epsilon$ and $<^\epsilon$ and the epsilon value, a flexible version of Allen's seven relations is defined, as shown in Figure 6.

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 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.

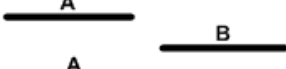

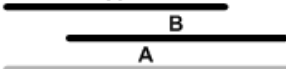
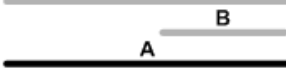
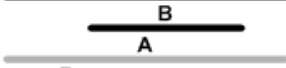
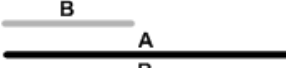
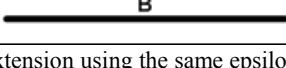
before (<)		$A.e <^{\epsilon} B.s \ \&\& \ B.s - A.e < \text{max_gap}$
meets (m)		$A.e =^{\epsilon} B.s$
overlaps (o)		$A.s <^{\epsilon} B.s \ \&\& \ B.s <^{\epsilon} A.e \ \&\& \ A.e <^{\epsilon} B.e$
finish-by (fi)		$A.s <^{\epsilon} B.s \ \&\& \ A.e =^{\epsilon} B.e$
contain (c)		$A.s <^{\epsilon} B.s \ \&\& \ B.e <^{\epsilon} A.e$
start-by (si)		$A.s =^{\epsilon} B.s \ \&\& \ B.e <^{\epsilon} A.e$
equal (=)		$A.s =^{\epsilon} B.s \ \&\& \ A.e =^{\epsilon} B.e$

Fig 6. A flexible extension using the same epsilon for all Allen's seven relations, using the same Epsilon for all the relations.

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 figure 6 keeps the set of the temporal relations mutually exclusive. This is indeed true, since the epsilon-extended temporal-relation definitions appearing in Figure 6 imply that for any two time intervals, exactly one (epsilon-extended) temporal relation applies.

Definition 2. A *symbolic time interval*, $I = \langle s, e, \text{sym} \rangle$, 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.

Definition 3. A *symbolic time interval series*, $IS = \{I_1, I_2, \dots, I_n\}$, where each I_i is a symbolic time interval, represents a series of symbolic time intervals, over each of which holds a symbolic concept.

Definition 4. A *lexicographic symbolic time-interval series* is a time-interval series, sorted in the order of the start-time, end-time using the relations $<^{\epsilon}$, $=^{\epsilon}$ and a lexicographic order of the symbols, $IS = \{I_1, I_2, \dots, I_n\}$, such that:

$$\forall I^i, I^j \in IS \ (i < j) \wedge ((I_s^i <^{\epsilon} I_s^j) \vee (I_s^i =^{\epsilon} I_s^j \wedge I_e^i <^{\epsilon} I_e^j) \vee (I_s^i =^{\epsilon} I_s^j \wedge I_e^i =^{\epsilon} I_e^j \wedge I_{\text{sym}}^i < I_{\text{sym}}^j))$$

Since in our problem definition the time intervals are ordered lexicographically, we use only the seven temporal relations shown in figure 6.

Definition 5. A non-ambiguous lexicographic *Time Intervals Relations Pattern (TIRP)* P is defined as $P = \{I, R\}$, where $I = \{I_1, I_2, \dots, I_k\}$ is a set of k symbolic time intervals ordered lexicographically and

$$R = \bigcap_{i=1}^k \bigcap_{j=i+1}^k r(I_i, I_j) = \{ r_{1,2}(I_1, I_2), r_{1,3}(I_1, I_3), \dots, r_{1,k}(I_1, I_k), r_{2,3}(I_2, I_3), \\ r_{2,4}(I_2, I_4), \dots, r_{2,k}(I_2, I_k), \dots, r_{k-1,k}(I_{k-1}, I_k) \}$$

defines all the temporal relations among each of the $(k^2-k)/2$ pairs of symbolic time intervals in I .

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 figure 5 presents all of the pair-wise temporal relations among the TIRP's symbolic time intervals, ordered lexicographically, thus 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 lexicographic ordering, which leads to a unique half matrix for each TIRP.

The half matrix representation of the temporal relations defining a TIRP will be used throughout this paper and in explaining the implementation of the KarmaLego algorithm. Actually, the half matrix is implemented as a vector. The digits at the top left corner of each cell in the half matrix represent its index in the vector. Thus, Figure 5 represents a conjunction of ten temporal relations. For example, the cell indexed as number 0 in that vector represents the relationship A^I overlaps B^I .

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 for which P holds, divided by the total number of entities (e.g., patients) $|E|$: $ver_sup(P) = |E_P| / |E|$. The vertical support is actually what is commonly used as support in association rules, itemset 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) is the number of instances of the TIRP P found in e_i - $hor_sup(P, e_i)$.

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 with horizontal support ≥ 1).

$$mean_hor_sup(P, E^P) = \frac{\sum_{i=1}^{|E^P|} hor_sup(P, e_i)}{|E^P|}$$

Definition 9. The time-intervals duration variance (TIV) of a k -sized TIRP P , having n supporting instances, in which I^{Dur} is the duration of the time interval I ($I_{l,m}$ is the l^{th} time interval (i.e., component) of the k time intervals of which instance m is composed) is:

$$TIV(P) = \frac{\sqrt{\sum_{l=1}^k \sum_{m=1}^n (I_{l,m}^{Dur} - \overline{I_l^{Dur}})^2}}{n \cdot k}$$

Where $\overline{I_l^{Dur}}$ is the mean of I_l^{Dur} , that is, the mean duration of the l^{th} component in all of the TIRP's n instances.

Definition 10. The time-intervals 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 Improving the Use of Allen's Relations during the Process of Time Intervals Pattern Mining

In order to unambiguously represent a TIRP using Allen's temporal relations, it is necessary to comply with definition 5, in which a k -sized TIRP P is defined by $(k^2 - k)/2$ temporal relations. Thus, the complexity of generating potentially frequent TIRP candidates grows exponentially with k . Several methods for candidate generation were mentioned in Section 2.

The KarmaLego algorithm was inspired by Papapetrou's method, in the sense that it adopted the concept of first indexing, using BFS, all of the 2-sized TIRPs, and then extending these TIRPs into larger k -sized TIRPs, based on the previously generated index of the two sized TIRPs. The KarmaLego algorithm was first introduced in a workshop [15] and a conference [14], but here we first describe it in detail, including the direct extension of TIRPs during the process of candidate generation and the pruning of potential candidates by exploiting the transitivity property of the temporal relations. Overall, the KarmaLego process provides a much more efficient generation of candidates for time interval relation patterns, as demonstrated also by the runtime comparison presented as part of the evaluation in Section 4.

3.4 The KarmaLego Algorithm

The KarmaLego (algorithm 1) consists of two main phases. The first phase is called

Karma¹, in which 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, a recursive process extends the frequent 2-sized TIRPs, referred to as T^2 , through efficient candidate generation, using the Lego² 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 (Figure 7).

Algorithm 1 – KarmaLego

Input:

db – A database of $|E|$ entities representing for each the symbolic time intervals of $|S|$ symbols;

min_ver_sup – the minimal vertical support threshold;

Output: T – an enumerated tree of all frequent TIRPs

1. $T \leftarrow \text{Karma}(\text{db}, \text{min_ver_sup})$
2. Foreach $t \in T^2$ // T^2 is T at the 2nd level
3. Lego($T, t, \text{min_ver_sup}$)
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.*

3.5. Karma

In algorithm 2 (Karma), the first level of the enumeration tree is constructed based on the set of the frequent symbols S , which is actually the first level of the enumeration tree, T^1 , as shown in figure 7. Then, all the 2-sized TIRPs are generated by the frequent symbols in T^1 related by the varying temporal relations in R , as described in lines 3 to 8. For each entity e , the method incrementally indexes each pair of symbolic time intervals e^s_i, e^s_j ; i and j represent their order in the symbolic time intervals series ordered lexicographically in entity e ; and s is their symbol. After determining the temporal relation among the two symbolic time intervals, based on their start-time and end-time, according to definition 1, the 2-sized TIRP instance is indexed in the proper node in the enumeration tree according to its symbols and their temporal relation. After the indexing of all of the time interval pairs is completed, each t in T^2 that is not frequent is removed (pruned) from the tree. Eventually, an enumeration tree T is returned, at this point having only two levels.

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

² Lego – A popular game, in which modular bricks are used to construct different objects.

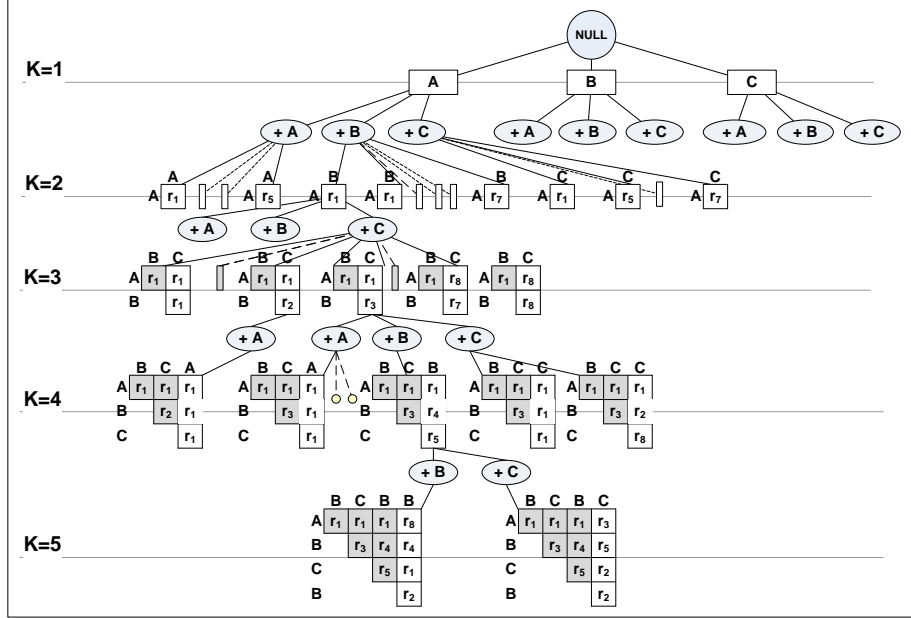


Fig 7. A KarmaLego enumeration tree, in which the direct expansion of TIRPs is performed. Each node represents a TIRP as a half matrix of the temporal relations. See text and detailed algorithms for an explanation.

The Karma procedure "grows" all of the frequent symbols represented by T^l , by generating all the 2-sized TIRPs, which are indexed in T^2 . These are all the combinations of the pairs of the frequent symbols at T^l for all of the potential temporal relations.

Thus, having n frequent symbols in $n = |T^l|$, there are n^2 pairs of symbols, which can generate a maximum of $7 \times n^2$ 2-sized frequent TIRPs (when using Allen's 7 temporal relations). Each 2-sized TIRP contains two symbols: the first, the second, and the temporal relation between them. The data structure that indexes, for all the instances of the entities (e.g., patients) that include a given 2-sized TIRP, for example $t^2 \langle s_1, r, s_2 \rangle$, is indexed using a hashing function: Entity instances are indexed based on the entity ID (e.g., patient ID) and the first time interval's start-time and end-time. This hashing function is later used during the search phase in Lego, in which interval pairs are retrieved based on the entity ID and on the first time interval's start-time and end-time.

To better present the rest of the KarmaLego method, we delve a bit deeper into the data structure involved in the process.

Each TIRP (frequent or candidate) in the extension process (during which there exist TIRPs that are longer than 2 sized) is represented by the following data structure:

TIRP.s – a vector of its ordered symbols;

TIRP.size - the size of the TIRP, defined as the number of the symbolic time intervals in the TIRP ($|TIRP.s|$);

TIRP.tr – a vector that represents the half matrix defining its conjunction of temporal

relations, as was shown in figure 5;

$TIRP.tr_size$ - the size of the half matrix defining the temporal relations in the TIRP, which is $(TIRP.size^2 - TIRP.size)/2$;

$TIRP.inst$ - a vector of the instances supporting the TIRP;

$TIRP.inst.e_id$ - contains the entity id supporting the instance;

$TIRP.inst.ti$ - a vector of pointers to the actual time intervals of the supporting instance. Note that the positions in all of the vectors of size k are indexed by the numbers $0..k-1$.

Algorithm 2 – Karma

Input:

db – A database of $|E|$ entities representing for each the symbolic time intervals of $|S|$ symbols;

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. $T^1 \leftarrow$ all the symbols above min_ver_sup
3. Foreach $e \in E$
4. Foreach $e_i^s, e_j^s \wedge i < j \wedge s \in T^1$
5. $r \leftarrow$ the temporal relation among e_i^s, e_j^s
6. Index($T^2, < e_i^s, r, e_j^s >$)
7. End Foreach
8. End Foreach
9. Foreach $t \in T^2$
10. if ver_sup(t) < min_ver_sup
11. Prune(t)
12. End Foreach
13. return T
14. End

3.6. The Direct Extension Step in KarmaLego

In the H-DFS method [16], when attempting to extend a TIRP, a similar data structure to T^2 is used, as explained regarding the Karma algorithm. In that method, the entire set of combinations of conjunctions of 2-sized TIRPs (i.e., r^k relation combinations for r temporal relations and k pairs in the conjunction) is computed as potential candidates each time, when extending a k -sized TIRP.

However, in KarmaLego, the candidate generation is made *directly*, i.e., by considering only specific frequent k -sized TIRPs and attempting to extend them. Thus, extending a k -sized TIRP into a $(k+1)$ sized TIRP is made by adding a frequent symbolic time interval and adding all the possible temporal relations among the added symbolic time interval and the existing k time intervals, according to the lexicographical order in the TIRP. (This procedure is somewhat similar to the one used by Patel et al. [17]).

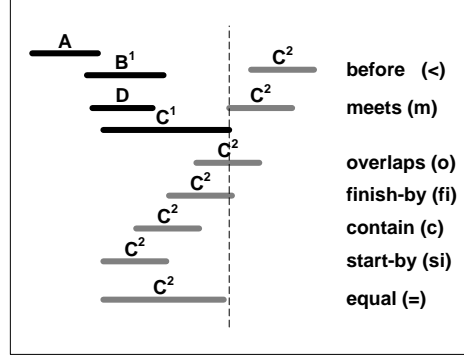


Fig 8. A 4-sized TIRP (denoted by the black intervals) is directly extended by a symbolic time interval C^2 , each time related by another temporal relation, among Allen's seven temporal relations, to the last symbolic time interval C^1 .

More specifically, as shown in Figure 8, the extension is done by generating for each frequent symbol a new symbolic time interval, and generating all of Allen's 7 relations between the new symbolic time interval and the latest, in lexicographic order (see Definition (5)), symbolic time interval (i.e., the most recent addition) within the current extended TIRP. The last step is generating all of Allen's 7 temporal relations as the temporal relations among the new symbolic time interval and the existing TIRP's time intervals.

In figure 8, a 4-sized TIRP is extended by a symbolic time interval C^2 that can be related to the TIRP's last time interval C^1 (note that C^1 is last according to the lexicographic order), having the latest start time, by any of Allen's 7 temporal relations. The next step is generating, for each of these 7 possible temporal relations between C^2 and C^1 , all of the temporal relations among C^2 and the previous time intervals: D , B^1 and A .

In this example having to set seven optional temporal relations in three temporal relations ($r(A, C^2)$, $r(B^1, C^2)$ and $r(D, C^2)$) would in theory generate up to $7^3 = 147$ candidates for each of Allen's 7 possible temporal relations between C^2 and C^1 . (Later we will show how the 147 candidates can be decreased dramatically by exploiting the transitivity property). Finally, for each candidate TIRP, we must check whether all of its component relations are frequent and whether overall it is frequent. The extension process is implemented by the Lego algorithm.

Thus, Lego (Algorithm 3) receives a k -sized TIRP and extends it by all of the frequent symbols in T^l and a set of predefined desired temporal relations R (Allen's seven relations, or other versions). First, all the frequent symbol candidates are generated and are added by relating them to the last time interval in t , using each of the temporal relations in R (lines 1 and 2). In line 3 a new TIRP is created of the extended size and a symbol s and a relation r are set to t^c in lines 4 and 5. C contains all of the candidates that were generated in line 7 by the method Generate Candidate TIRPs that will be described in the next section. Then, for each candidate c in C , if the frequency of the supporting instances that were found (line 9) is above the minimal vertical support threshold, c is added to the enumeration tree as an extension of t , and is further extended by calling Lego.

The candidate generation can be made in two ways: a naïve candidate generation method, as will be presented in Algorithm 4, which is commonly used, or a faster method, exploiting a transition table which is implemented in KarmaLego, as shown in Algorithm 5. Note that in line 9, we search for all instances supporting a $k+1$ sized candidate TIRP c . This is performed using the T^2 data structure that was described earlier, and the list of specific tuples that are already known to satisfy the k -sized TIRP. (See precise details in Algorithm 6.)

Algorithm 3 – Lego($T, t, \text{min_ver_sup}$)

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

For example, for the first pair-wise relation including the new $(k+1)$ symbolic time interval, we start by examining the first entity in which it appears, retrieve the first symbolic interval in the desired relation with it, and then search, in the T^2 table indexing the next pair-wise relation in the TIRP's conjunction, whether the same entity and the same symbolic interval with which the $k+1$ symbol is in a relation, appear, indicating an entity for which both temporal relations hold for the appropriate symbolic intervals. This process is repeated as necessary, while maintaining each time the correct entity and the relevant set of symbolic 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.

3.7. Generating Candidate TIRPs: Exploiting Transitivity in Temporal Relations

In line 3 of algorithm 2 an extended candidate TIRP t^c is created based on TIRP t , a symbolic time interval s and a temporal relation r . Figure 9 illustrates an extension

process of a 4-sized TIRP, including the symbolic time intervals: A^1 , B^1 , C , B^2 , with a symbolic time interval A^2 . The half matrix on the left presents the conjunction of the temporal relations defining the extended TIRP, in which each column contains the temporal relations among the symbolic time interval at the top and the earlier (lexicographically) time intervals in t on the left. The temporal relations, at the cells having the grey background, are the temporal relations defining t . For example, the temporal relation between B^1 and B^2 is $<$ (before). The rightmost column contains the temporal relations among the new symbolic time interval (A^2) and the time intervals in t . The temporal relations that will be set in this column will generate the entire set of candidate TIRPs C .

Algorithm 4 generates (somewhat naively, as we shall see) the entire set of candidates C by first setting all the R temporal relations to the temporal relation between the new symbolic time interval and the latest time interval in the extended TIRP, and then generating all the combinations of the temporal relations in the rest of the relations among the new symbolic time interval and the time intervals of TIRP t . For example, in figure 9, the temporal relation among the new symbolic time interval and the latest, shown in the cell having the index 9 is set to the temporal relation m (meet). There are three temporal relations to generate (set), which total will be R^3 ; so for the case of using $R=7$ temporal relations, there will be $7^3 = 343$ candidate TIRPs to be searched in T^2 (this is true with any relation set to index 9 and not specifically *meet*).

Algorithm 4 is recursive, and each time it creates for each temporal relation r in R a new candidate c^{new} , which is a copy of c (line 2), and r is set to the temporal relation between the new time interval and the $rIdx+1$ earlier time interval. When all the temporal relations were set, $rIdx$ is smaller than the size of the TIRP and the generated TIRP is ready and added to C , otherwise the function is called again with c^{new} . Eventually the function returns the set of TIRP candidates C .

Algorithm 4 brute force candidate generation process results in a large number of candidates. The number of candidates grows exponentially with the number of time intervals in the TIRP.

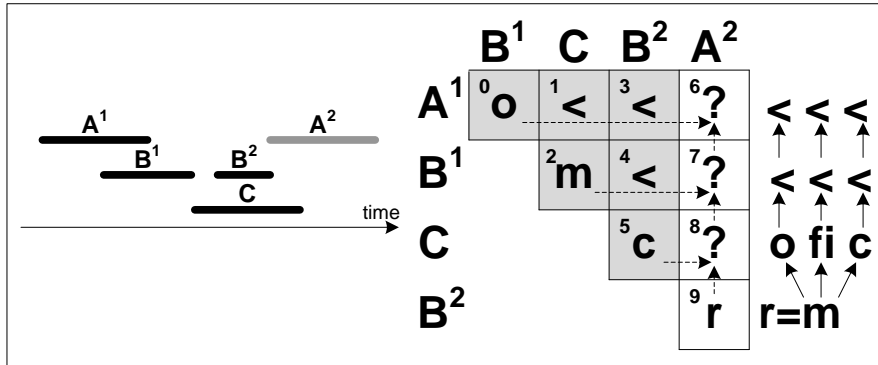


Fig 9. The transitivity property is exploited in order to generate efficiently the possible temporal relations that are the candidate TIRPs, given a temporal relation that was set between the new time interval relations of the previous time intervals of the TIRP using the temporal transition table (TTT).

Thus, for example extending a 4-sized to a 5-sized TIRP (as the example in figure 9) generates $7^4 = 2401$ candidates, and extension from a 5-sized to a 6-sized TIRP will generate $7^5 = 16,807$ candidates for each added symbol.

However, note that such naïve generation results with many combinations of temporal relations that contradict each other and create an impossible TIRP. For example, consider a candidate based on figure 9, in which $r(B^2, A^2) = \text{meets}$, and then the following [potential candidate] settings are made: $r(C, A^2) = \text{meets}$, $r(B^1, A^2) = \text{meets}$, $r(A^1, A^2) = \text{meets}$. Obviously, setting $r(B^2, A^2)$ to *meets* means that it is impossible to have $r(C, A^2) = \text{meet}$, since the temporal relation $r(C, B^2)$ is *contains*; similarly, the relations $r(B^1, A^2)$ and $r(A^1, A^2)$ contradict the existing relations.

Algorithm 4 - Gen_Candidate_TIRPs (Naively)

Input:

c – the base TIRP to generate

$rIdx$ – the temporal relation index to generate

Output: C – set of the candidate TIRPs

1. Foreach $r \in R$
2. $c^{new} \leftarrow c$
3. $c^{new}[c.tr_size-1-rIdx] \leftarrow r$
4. if($c.size - rIdx > 2$)
5. $C \leftarrow C \cup \text{Gen_Candidate_TIRPs}(c^{new}, rIdx+1)$
6. else
7. $C \leftarrow C \cup c^{new}$
8. EndForeach
9. return C
10. End

The ability to perform candidate generation without contradictions should decrease the number of candidates, and more important, should decrease the need to search for impossible candidates in T^2 . To implement this capability, we use *transition tables* [1,4], which we introduced in section 2.3. After setting the relation $r(B^2, A^2)$ to *meets*, the disjunction of the possible relations in $r(C, A^2)$ can be computed based on $r(C, B^2)$ and $r(B^2, A^2)$ from a transition table, which results in this case with a disjunction of three possible temporal relations: *overlaps* (o), or *finished-by* (fi) or *contains* (c), as shown on the right side of figure 8. Similarly, the possible relations of $r(B^1, A^2)$ can be computed based on $r(B^1, C)$ and $r(C, A^2)$. Since $r(C, A^2)$ had a disjunction of three relations, the computation will be performed for each of the three relations. Thus, when $r(C, A^2) = o$, using a transition table results in the conclusion that $r(B^1, A^2)$ is *before* ($<$), and the same type of reasoning results in the cases, in which $r(C, A^2) = fi$, and $r(C, A^2) = c$. The same process is repeated to reason about $r(A^1, A^2)$ for each of the relations, which in this example were the same – *before* and result in the same relation, *before* as well. Thus, instead of $7^3 = 343$ candidates, there are actually only 3 candidates that do not contradict the TIRP's relations. This process, which is implemented in Algorithm 5, obviously decreases dramatically the number of candidates that have to be searched in T^2 and the time that is required for that, as will be shown also empirically in the runtime evaluation results.

Algorithm 5 is a recursive process that exploits the transitivity property of temporal relations in order to generate candidates efficiently. The procedure gets the extended TIRP c and the temporal relation-index to generate. The relation-index ($rIdx$) starts as the temporal relation, that was set in Lego (Algorithm 2) among the new symbolic time interval and the latest time interval, and ends with last relation among the new time interval and the first time interval (for example, $r(A^1, A^2)$ in figure 9). Given the size of the TIRP $c.size$ and $rIdx$, the indices of the first and second relations to be looked up in a transition table are calculated (lines 1,2) and the actual disjunction of temporal relations – $trans_rels$ – is computed and returned by the transitive table at line 3. Then, for each of the relations, a new TIRP c^{new} is generated in line 5 and the relation is set in the appropriate location in the relations-half-matrix vector. If the index of the first relation, $first_rel$ is zero, then the generated set of candidate TIRPs c^{new} is added to C , otherwise $Get_Candidate_TIRPs$ is called again with c^{new} . Eventually the function returns the set of candidate TIRPs C . After the candidates C were generated, the supporting instances are searched in line 9 in Lego (Algorithm 2).

Algorithm 5 - Gen_Candidate_TIRPs (with Transitivity)

Input: c – the base TIRP to extend
 $rIdx$ – the temporal relation-index currently set
Output: c^{new} – the candidate TIRP

1. $first_rel \leftarrow (((c.size - rIdx)^2 - (c.size - rIdx)) / 2) - 1$
2. $second_rel \leftarrow c.tr_size - rIdx$
3. $trans_rels \leftarrow trans_table[c.tr[first_rel], c.tr[second_rel]]$
4. Foreach $rel \in trans_rels$
5. $c^{new} \leftarrow c$
6. $c^{new}[scnd_rel - 1] \leftarrow rel$
7. if ($first_rel > 0$)
8. $C \leftarrow C \cup Gen_Candidate_TIRPs(c^{new}, rIdx + 1)$
9. else
10. $C \leftarrow C \cup c^{new}$
11. EndForeach
12. return C
13. End

Algorithm 6 describes the method of searching for supporting instances, which counts the precise vertical support for each extended candidate TIRP. The method receives as input the candidate TIRP c and the set of the two sized TIRPs in T^2 . In line 1, the new symbol that was added (in Algorithm 3 – Lego) is set to new_s , then for each instance in the supporting instances of the extended TIRP the search is made. First the temporal relation rel between the new time interval and the latest (in the extended TIRP) is set (line 3), then the latest symbol of the extended TIRP s is set (line 4). Then $GetLastTIs$ searches in the relevant t^2 , defined by the symbols s and $last_s$ and the temporal relation rel , for the instances starting with the latest time interval in the instance $ins.ti$. $GetLastTIs$ might return several new time intervals new_tis .

For each new time interval new_ti , the existence of all the temporal relations among new_ti and the earlier time intervals of the extended TIRP have to be checked in T^2 (lines 9-15). First, $inst$ is copied into a new instance $inst^{new}$ (line 7), the new time

interval new_ti is set to its proper location (line 8), and then, using a loop, the pairs of the time intervals and the temporal relations are searched at the relevant T^2 (lines 9-16). If the Boolean method $NoInst?$, which is searching for the pair of intervals at the right T^2 , returns true, meaning that no instances of the relation were found for the same entity, the search is stopped (line 13), otherwise at the end of the search $inst^{new}$ is added to $c.inst$. This process is repeated for each of the new_tis . The searched instance $inst$ is removed from $c.inst$, since it was replaced by the $inst^{new}$ that were found frequent, and if not, it should be removed too (line 18). Finally the candidate c is returned with the supporting instances.

Algorithm 6– Search_Supporting_Instances

Input: c – the TIRP to search

T^2 – the 2-sized TIRPs to search

Output: c – with the supporting instances

```

1.  $new\_s \leftarrow c.s[c.size-1]$ 
2. Foreach  $inst \in c.inst$ 
3.    $rel \leftarrow c.tr[c.tr\_size-1]$ 
4.    $s \leftarrow c.s[c.size-2]$ 
5.    $new\_tis \leftarrow GetLastTIs(inst.ti[c.size-2], t^2 < s, rel, new\_s >)$ 
6.   Foreach  $new\_ti \in new\_tis$ 
7.      $inst^{new} \leftarrow inst$ 
8.      $inst^{new}.ti[c.size-1] \leftarrow new\_ti$ 
9.     For( $i=1$ ;  $i < c.size-1$ ;  $i++$ )
10.       $rel \leftarrow c.tr[c.tr\_size-1-i]$ 
11.       $s \leftarrow c.s[c.size-2-i]$ 
12.      if( $NoInst?(inst^{new}.e\_id, inst^{new}.ti[c.size-2-i], new\_ti, t^2 < s, rel, new\_s >)$ )
13.        break
14.   EndFor
15.    $c.inst \leftarrow inst^{new}$ 
16. End Foreach
17. remove  $inst$  from  $c.inst$ 
18. End Foreach
19. return  $c$ 
20. End

```

3.8 Reducing TIRP instances variance via pre-clustering

Discovery of TIRPs from multivariate time series that have undergone a temporal abstraction process has a great potential, as previously discussed, for clustering the database entities (i.e., patients) by TIRPs, and for potentially providing useful features for future classification of a set of input data by a predefined set of outcomes. It also can provide a visual representation of the temporal patterns to experts who wish to understand the typical temporal behaviors of the entities, such as a group of patients [14].

However, the definition of a non-ambiguous TIRP (see Definition 5) is *non-metric*: it pays no attention neither to the duration of the symbolic time intervals, nor to the duration of temporal gaps between them, or to overlaps or containment periods, as part of the definitions of their temporal relations; all of these aspects might significantly affect the meaning of a TIRP. This omission occurs since a symbol (which is, in fact, a predicate, such as "*Low blood pressure, as defined in the context of pregnancy*" or "*a Very high number of open connection sockets*") that holds over a symbolic interval means the same for the purpose of the TIRP enumeration process, regardless of its duration: it is generated by the temporal-abstraction process and has identical semantics over all intervals of the same type.

This lack of a metric definition regarding the duration of the intervals that compose the TIRP, and the precise quantitative nature of each temporal relation, might pose a problem. The instances of discovered TIRPs may in reality vary significantly in the durations of the time intervals that they are composed from, or in the nature of their internal temporal relations, while still satisfying the same TIRP constraints, as illustrated in figure 10. Figure 10 presents three very different instances of the same TIRP definition. In the real world, such as for a particular clinical domain, these quantitatively different instantiations of the same TIRP might represent quite qualitatively different scenarios.

Therefore, to make the instances satisfying the definition of the same TIRP more homogeneous, with respect to the duration of the symbolic time intervals, we propose to first cluster the time interval instances of each symbol according to their durations. Thus, having for example a database with the interval-based symbols A , B , and C , we can first cluster their instances into k clusters according to their duration prior to the mining process by using an appropriate method such as the k -means clustering algorithm. The result will be that we will have potentially more symbols, up to a maximum of k subtypes of each original symbol, and a minimum of 0 subtypes of the original symbol – in the case that none of the clusters were sufficiently frequent.

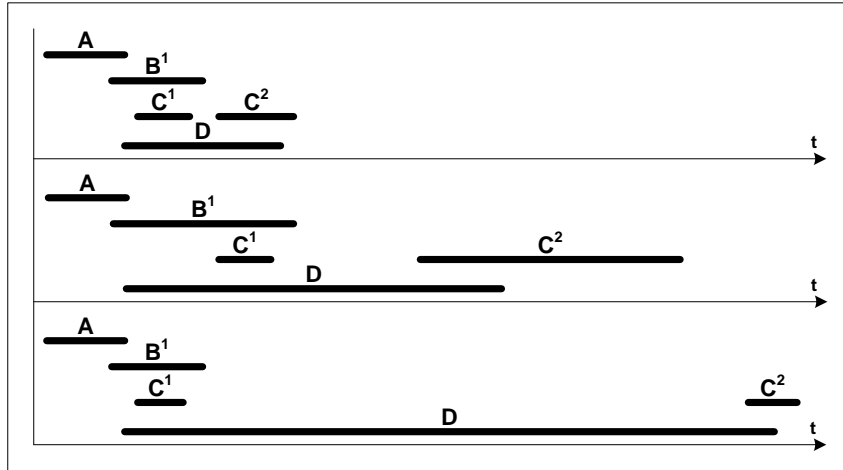


Fig 10. Various instances of TIRPs, all satisfying the same definition, demonstrating the need for having more homogenous instances. Note for example the highly different instantiations of the Before relation between C^1 and C^2 and of the duration of intervals such as C^2 and D .

As can be readily understood, there is a tradeoff involved here: clustering symbolic time intervals of a specific symbol into sub-symbols according to their duration results in more types of symbols, each of which is less frequent, which results in a potentially smaller number of frequent TIRPs; but those TIRPs that are still sufficiently frequent, will be supported by more homogenous instances. With respect to the possibility of clustering also temporal *relations*, we consider it as probably unnecessary following the clustering of the interval durations (see the discussion of this point in Section 5).

4 Evaluation

The objectives of the evaluation were twofold.

The first objective focused on a comparison of KarmaLego's performance to that of previous methods using Allen's temporal relations, on various datasets. The second objective of the evaluation focused on the effect of pre-clustering the symbolic time intervals by their duration, prior to the mining process, on the variance of the supporting instances of the TIRPs (see Definition 9) and on their vertical support.

4.1 Datasets

4.1.1 The American Signup Language Dataset

The American Signup Language (ASL) dataset was created by the National Center for Sign Language and Gesture Resources at Boston University [16]. It consists of a collection of 884 utterances, in which each utterance associates a segment of video with a detailed transcription. Every utterance contains a number of ASL gestures and grammatical fields (e.g., eyebrow raised, head tilted forward), each one occurring over a time interval. This dataset was used by Papaetrou et al [16] for the runtime evaluation.

4.1.2 The MavLab Smart House Dataset

The MavLab SmartHome dataset, provided by Jakkula et al [7] contains data from the readings of 99 sensors installed in a computerized apartment. The sensors described the activity of people and of various appliances scattered around the apartment and were sampled each single second over a period of 81 days, in which only the data for the first 3 hours from each day were used. This dataset was used for the runtime evaluation and for the clustering experiments.

4.1.3 The Diabetes Dataset

The diabetes dataset, provided as part of collaboration with Clalit Health Services (Israel's largest HMO) contains data on 2038 patients who have type II diabetes. The data was collected each month from 2002 to 2007. The dataset contains six concepts (laboratory values or interventions) recorded over time for each patient: hemoglobin-A1c values, blood glucose levels, cholesterol values, and several medications the patients purchased: diabetic (insulin-based) medications, cholesterol reducing statins, 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. The diabetes dataset was used for the runtime evaluation and for the clustering experiments.

Table 1 contains the cutoff 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. The knowledge-based, state abstraction definitions for the measurements were provided by physicians from Ben Gurion University's Soroka Medical Center. In the following tables the cutoff definitions are presented for each state for the various temporal measurement variables in the Diabetes dataset.

Table 1 Measurement data types and their cut-off (discretization) values for 3 or 4 state abstractions each, in the case of the diabetes data set.

Blood Glucose	
State	Glucose
1	< 100
2	100-125
3	126-200
4	>200

HemoglobinA1C	
State	HbA1c
1	< 7
2	7-9
3	9-10.5
4	>10.5

LDL Cholesterol	
State	LDL
1	< 100
2	100-130
3	130-160
4	> 160

HDL Cholesterol		
State	HDL-Male	HDL-Female
1	< 35	< 30
2	35 – 45	30 – 40
3	> 45	> 40

4.1.4 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 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 hours during her postoperative ICU stay, and high frequency time series, measured each minute along the first 12 hours of the ICU hospitalization.

This 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. The dataset contains 664 patients; 196 patients were mechanically ventilated for more than 24 hours.

4.2 Run-time Evaluation of the KarmaLego Algorithm on the Various Datasets

In evaluating the runtime performance, a comparison of the runtime was performed among (1) the KarmaLego method, (2) an implementation of H-DFS method [16], (3) an implementation of ARMADA [27] and (4) an implementation of IEMiner [17] method.

For the runtime evaluation, three real datasets were used: *ASL*, *Diabetes* and the *SmartHome*, which were described earlier. The datasets are characterized by four parameters in the context of the runtime evaluation: *N* - the entire number of time intervals in the dataset; *S* - the number of symbols; *E* - the number of entities (i.e., patients or subjects); and *I* - the mean number of intervals per entity, as shown in Table 2. While the ASL dataset is relatively small in all the parameters, the Diabetes dataset contains a significantly larger number of time intervals (*N*) and of entities (*E*). The SmartHome dataset has the largest mean number of time intervals per entity (*I*).

Table 2. The properties of the evaluation datasets.

Dataset	N	E	S	I
ASL	2,037	65	146	31.3
Diabetes	80,538	2,038	227	39.5
SmartHome	30,210	81	212	372.9

N - the entire number of time intervals in the dataset; *S* - the number of symbols; *E* - the number of entities (i.e., patients or subjects); and *I* - the mean number of intervals per entity

4.2.1 Experiments with the ASL Dataset

All four methods, including: H-DFS, Armada, IEMiner, and KarmaLego, were run on the ASL dataset using 5 levels of *ver_min_sup*, starting with 50% and ending with 10%. Figure 11 presents the runtime measurements in seconds for each method using a logarithmic scale for the computation time.

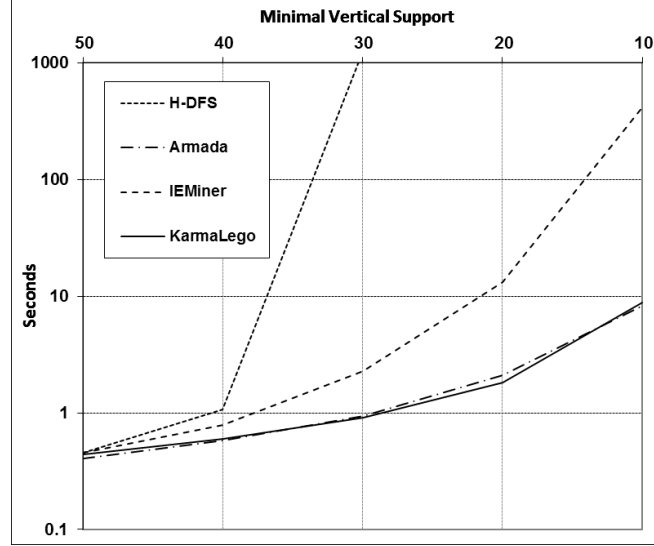


Fig 11. A performance comparison among four TIRP mining algorithms applied to the ASL data set. Computation time (in seconds) is displayed as a function of the minimal vertical support threshold (in percentages). Note that time is presented using a logarithmic scale.

In this dataset, Armada and KarmaLego behaved very similarly, while IEMiner was slower, and the H-DFS method was much slower. The similar behavior of Armada and KarmaLego on this data set can be explained by the relatively low value of I , the mean number of time intervals per entity, in this case a subject, which enables the Armada method to keep up with the KarmaLego method. As will be shown later, this is not the case when the mean number of intervals per entity is large, causing more difficulties to the Armada method.

Since the runtime of the H-DFS method is significantly slower than that of the other methods, it requires a logarithmic-scale graph for the clarity of presentation. Thus, its performance results are presented in this evaluation only once, using the relatively small ASL dataset.

4.2.2 Experiments with the Diabetes Dataset

Figure 12 shows the results of the experiments on the Diabetes dataset, in which N and E are the largest among the other datasets. While IEMiner was the slowest, KarmaLego was about twice as fast as Armada at very low levels of minimal vertical support.

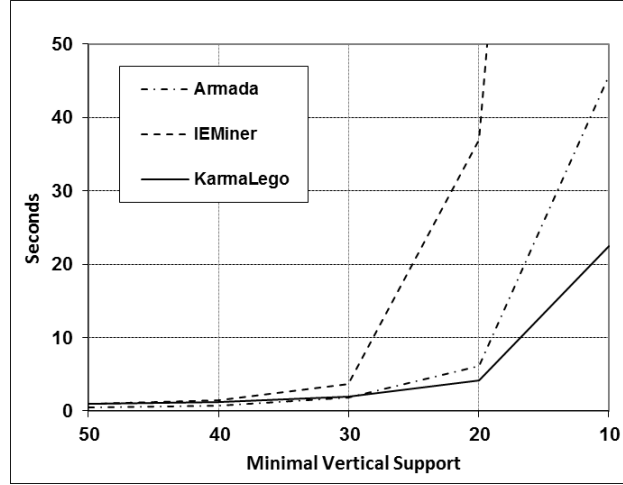


Fig 12. A performance comparison among three TIRP mining algorithms applied to the Diabetes data set. Computation time (in seconds) is displayed as a function of the minimal vertical support threshold (in percentages). The KarmaLego algorithm is faster than both others, especially for low levels of vertical support.

4.2.3 Experiments with the MavLab Smart Home Dataset

Experiments on the SmartHome dataset, which is characterized by the largest mean number of intervals per entity, I , are shown in figure 13. KarmaLego was much faster than both IEMiner and Armada. The difference was pronounced more clearly, compared to the ASL and Diabetes data sets, due to the large size of the MavLab data set.

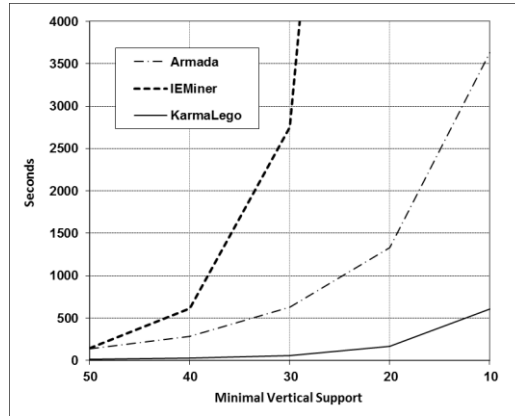


Fig 13. A performance comparison among three TIRP mining algorithms applied to the SmartHome data set. Computation time (in seconds) is displayed as a function of the minimal vertical support threshold (in percentages). KarmaLego is significantly faster than Armada, especially in the low levels of minimal vertical support.

Table 3. The evaluation datasets properties after the pre-clustering.

Dataset	N	E	I	NC	C2	C3	C4
D_3EWD	36,434	2,038	17.8	101	150	192	226
D_9EWD	73,144	2,038	35.8	234	336	410	463
SH_3EWD	42,847	88	486.9	232	366	477	573

N = No. of symbolic time intervals; E = No. of entities; I = mean No. of symbolic intervals per entity; NC = No. of symbolic-interval types with no clustering; $C2$ = No. of symbolic intervals types for $k=2$ clusters; $C3$ = No. of symbolic intervals types for $k=3$ clusters. D= Diabetes; SH = SmartHome.

4.3 Experiments in Pre-clustering Time Intervals by their Duration

Definition 5 defines a non-ambiguous TIRP from the temporal relations perspective. However, the symbolic intervals are not defined by their *duration*, which may result in varying time intervals duration in the specific instances supporting a TIRP. Thus, symbolic intervals can be pre-clustered by duration. However, as was explained earlier, there is a tradeoff between the pre-clustering and the number of discovered TIRPs. In order to learn about the influences of the pre-clustering on the variance of the TIRP instances and the number of the discovered TIRPs, an experiment was designed and evaluated according to several measures.

The assumption in this study was that pre-clustering the symbolic time interval instances in the dataset by their duration reduces not only the number of discovered TIRPs when the clustering parameter k (number of clusters) increases, but also the mean *time interval variance (TIV)* for discovered TIRPs (definition 9), due to the increased uniformity, at least in symbolic-interval duration, of the components of the discovered TIRPs.

In these experiments, which focused on TIRP counts and on interval *duration* variance measures, we used an epsilon value of zero (the Allen default); three values of k for the K-means clustering: two, three and four; and the non-clustered data for comparison. To evaluate the influence of these settings on the variance of the TIRPs time intervals instances variance, we counted the number of TIRPs discovered, C , and measured the TIV of discovered TIRPs.

The Diabetes dataset was abstracted into either 3 or 9 states using the *Equal-Width Discretization (EWD)* method (see Section 2.2), creating two datasets: Diabetes_3EWD and Diabetes_9EWD, respectively. The MavLab SmartHome dataset was abstracted into 3 states using EWD. The symbolic intervals of both datasets were pre-clustered by duration using the K-means method into $k = 2, 3$ and 4 clusters. The number of symbolic interval types in the dataset after each size of k pre-clustering is presented in Table 2. Note that the number of symbolic interval types after pre-clustering is not necessarily multiplied (approximately) by the k value, but it is bounded by that value, since the clusters may sometimes be empty.

Table 3 presents the quantitative properties of each dataset. Note that the number of symbolic time intervals after a 9EWD pre-clustering process is approximately twice than after a 3EWD state abstraction, since there are now many additional symbolic-interval types - each symbol type is fragmented into several subtypes (states), by its

value range; thus, the same is true for the mean number of symbolic intervals per entity. Furthermore, the number of symbolic interval types increases as the number of clusters grows, due to a similar fragmentation process, this time by duration.

Figures 14 and 15 display an analysis of the pre-clustering results for the Diabetes_3EWD dataset, for the 3-sized and 4-sized TIRPs respectively. In both figures, the TIV decreases when the number of clusters increases. In other words, the discovered TIRPs are more homogenous with respect to their duration. However, as expected, the *number* of discovered TIRPs also decreases, for reasons explained in Section 3.8. Note that the number of discovered TIRPs with four clusters was zero, in the case of the 4-sized TIRPs.

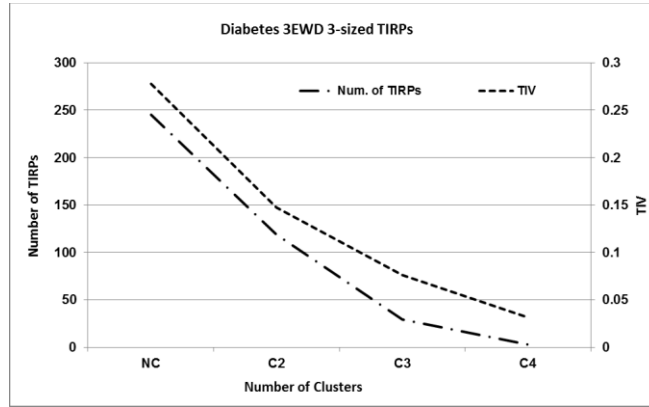


Fig 14. Pre-clustering symbolic intervals by duration in the Diabetes data set, when the discretization method is Equal-Width Discretization into 3 states (3EWD). The time-intervals-variance (TIV) decreases as the number of clusters grows. A similar observation applies to the number of discovered 3-sized frequent TIRPs.

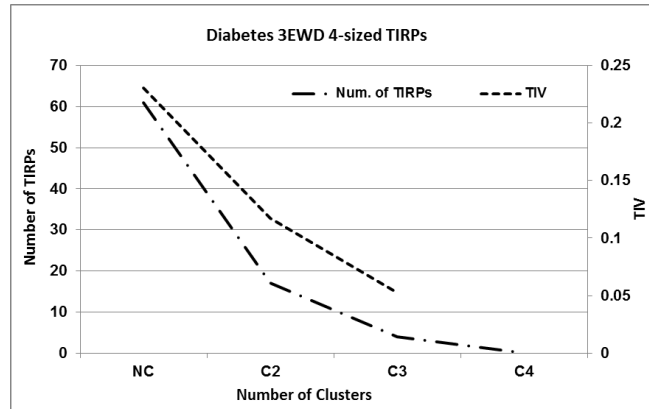


Fig 15. Pre-clustering symbolic intervals by duration in the Diabetes data set, when the discretization method is Equal-Width Discretization into 3 states (3EWD). The time-intervals-variance (TIV) decreases as the number of clusters grows. A similar observation applies to the number of discovered 4-sized frequent TIRPs.

Figures 16 and 17 display an analysis of the pre-clustering results for the Diabetes_9EWD dataset, for the 3-sized and 4-sized TIRPs respectively. In both figures, the TIV decreases when the number of clusters increases. In other words, the discovered TIRPs are more homogenous with respect to their time intervals duration. However, as expected, the *number* of discovered TIRPs also decreases here. Note that here the number of discovered TIRPs with three (and four) clusters was zero, in the case of the 4-sized TIRPs.

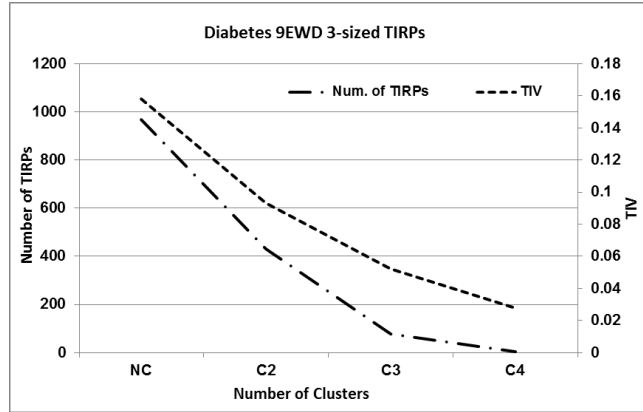


Fig 16. Pre-clustering symbolic intervals by duration in the Diabetes data set, when the discretization method is Equal-Width Discretization into 9 states (9EWD). The time-intervals-variance (TIV) decreases as the number of clusters grows. A similar observation applies to the number of discovered 3-sized frequent TIRPs.

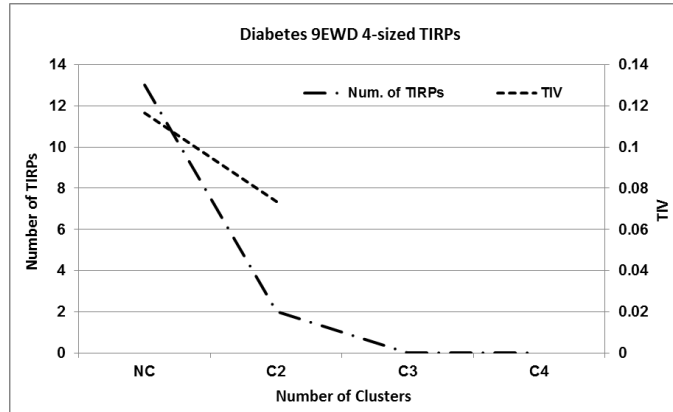


Fig 17. Pre-clustering symbolic intervals by duration in the Diabetes data set, when the discretization method is Equal-Width Discretization into 9 states (9EWD). The time-intervals-variance (TIV) decreases as the number of clusters grows. A similar observation applies to the number of discovered 4-sized frequent TIRPs.

The results of this experiment illustrate the expected differences in the discovered TIRPs for a dataset that is abstracted into a larger number of states. Abstracting into a larger number of states, as in the case of 9EWD vs 3EWD, creates a dataset with three times as many symbols; at the same time, shorter time intervals are expected as a

result of a larger number of symbols, especially if the data is changing fast. Note that in figures 14 and 16, in the case of the 3-sized TIRPs, using 3EWD, resulted in almost 300 discovered TIRPs, while using 9EWD resulted in about 1000.

Thus, the dataset with more symbolic states creates many more frequent TIRPs. Note that in the case of the 4-sized TIRPs, the number of TIRPs was zero when durations are clustered into four clusters, using a 3EWD discretization, but the number was zero already when clustering into three clusters, when using a 9EWD discretization. This can be explained by the larger number of symbols when using the 9EWD method, leading to smaller support for each symbol type. Indeed, note that even without any clustering, the number of frequent TIRPs is about a dozen when using 9EWD and few dozens when using 3EWD.

Figures 18 and 19 present the analysis results for the SmartHome_3EWD dataset, for discovery of 3-sized and 4-sized TIRPs, respectively. In both figures, the TIV decreases significantly as the number of clusters increases, while the number of frequent TIRPs decreases.

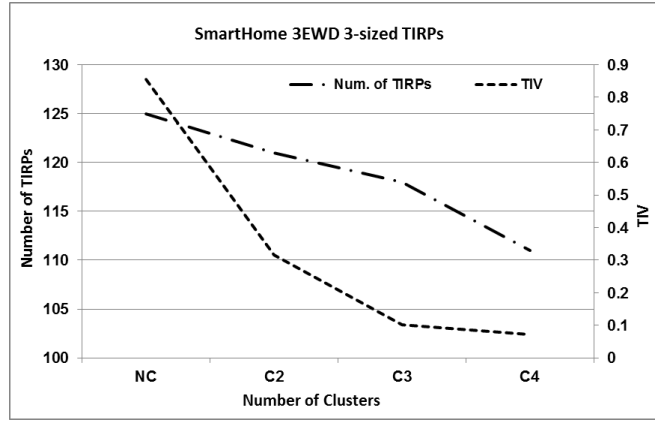


Fig 18. Pre-clustering symbolic intervals by duration in the SmartHome data set, when the discretization method is Equal-Width Discretization into 3 states (3EWD). The time-intervals-variance (TIV) decreases as the number of clusters grows. A similar observation applies to the number of discovered 3-sized frequent TIRPs.

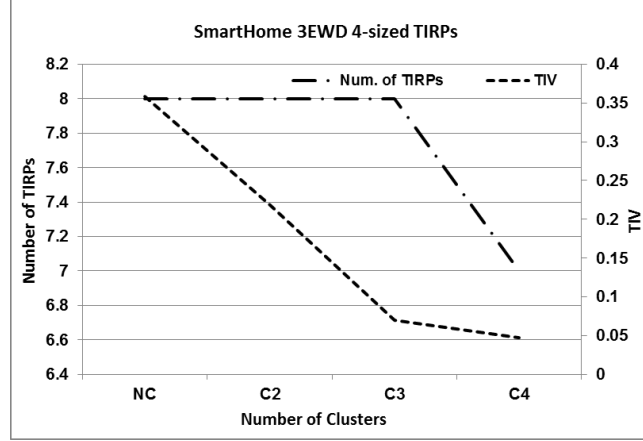


Fig 19. Pre-clustering symbolic intervals by duration in the SmartHome data set, when the discretization method is Equal-Width Discretization into 3 states (3EWD). The time-intervals-variance (TIV) decreases as the number of clusters grows. The number of discovered 4-sized frequent TIRPs decreases only when 4 clusters are created.

While the TIV decreased significantly for both 3-sized TIRPs (figure 18) and 4-sized TIRPs (figure 19), the number of discovered frequent TIRPs decreased less dramatically, especially for 4-sized TIRPs. Unlike in the case of the Diabetes datasets, the 4-sized TIRPs were frequent even with clustering by duration into three and four clusters. This can be explained by the larger number of time intervals per entity in that dataset (Table 2).

5 Discussion and Conclusions

We presented the idea of temporal knowledge discovery from multivariate temporal data via temporal abstraction, i.e., after abstracting the data into meaningful time intervals. The approach has significant potential, since temporal multivariate data often appear in varying granularities and frequencies. The effective discovery of temporal patterns in the data, referred to as TIRPs in this paper, supports temporal knowledge discovery, as well as potentially facilitating clustering of the entities by their temporal behavior, and classification of multivariate time series, including a form of prediction.

We introduced KarmaLego – a fast time-intervals mining technique which extends TIRPs directly and exploits the transitivity of the temporal relations to eliminate generated candidates. We demonstrated the use of the method on a real dataset of diabetic patients presenting a subtree of the discovered patterns.

The evaluation first compared the runtime performance of KarmaLego to other alternative methods: ARMADA, IEMiner, and H-DFS. Although the ARMADA algorithm is relatively fast, it becomes very slow, in large datasets, especially when there is a large mean of intervals per entities (I), such as in the SmartHome dataset and when using low levels of minimal vertical support. This can be explained by the

fact that the search in ARMADA is performed on the actual data, without indexing first the 2-sized TIRPs. While in both KarmaLego and the IEMiner method, the construction of TIRPs is performed on pairs of intervals that are indexed earlier (the Karma algorithm within KarmaLego), the IEMiner method is much slower than KarmaLego, since in KarmaLego the transitivity based candidate generation makes the candidate generation significantly more efficient and faster, resulting in a significantly smaller number of candidates to examine.

Each TIRP represents a cluster of patients who have similar temporal behavior along time, such as for example a similar pattern of a reaction to a particular drug dose. Each cluster, or TIRP P , can be described by its vertical support, mean horizontal support and other measures such as TIV. The extended (longer) TIRPs have equal or smaller vertical support; these are actually sub-cluster variations within a k -sized TIRP population.

Another useful way to characterize a cluster of patients by their temporal pattern is by the distribution of the static, non-temporal, variables characterizing these patients, which were *not* part of the TIRP discovery process (e.g., gender, age group). For example, during our evaluations, in the case of the diabetes data set, it seemed that certain TIRPs, such as those describing the response of the HbA_{1c} parameter (which approximates the severity of the diabetic state) to treatment by diabetes medications, are significantly more frequent in male rather than in female patients. Moreover, describing a temporal cluster by the dominant values of the static (non-temporal, such as demographic) parameters of its entities, in this case, patients, might lead to an alternative method of identifying the static properties of the subjects. For example, we might thus be able to characterize a group of patients implicitly; a pattern, say, of decreasing HbA_{1c} following a decrease in the dose of the medication might indicate, say, with high probability, females in their sixth decade.

Once the mining process is performed, we can also calculate the *transition probabilities for each temporal pattern*, namely, the probability of finding the $k+1$ time interval in the TIRP, given that the 1st to the k^{th} interval in the TIRP had been found. This definition is an extension of the concept of *confidence* (the conditional probability of finding B for the same entity when A was found, assuming an association rule $A \rightarrow B$), when mining association rules from static data. However, an analogous "temporal confidence" definition would need to consider the semantics of temporal relations (such as *overlaps* and *starts*) and the lexicographic order that we have imposed on the enumeration of components of the TIRP. The temporal confidence is especially important in the case of the *last* transition probability, i.e., when considering the case in which the k^{th} interval of the TIRP is the *last* one and is a meaningful outcome (e.g., life or death) that might be potentially predicted by the first $k-1$ intervals.

In addition to clustering intervals by *duration*, and measuring the variance in the duration of the TIRP's interval-based components, as we have done in our experiments, it is also possible in theory to cluster temporal *relations*, such as the duration of the gap between the two intervals in the case of the *Before* relation, or the duration of the overlapping segment, in the case of the *Overlaps* relation.

However, clustering time intervals by their *duration* should significantly decrease the need for clustering various aspects of the temporal relations, such as determining the

relative size of two intervals for the relations *starts* or *finishes*, due to the natural constraining of the possible relations by the duration clustering. (In fact, we would expect that following the duration-based clustering, only the clustering of gap durations within the *before* relation, and possibly the duration of the overlapped segment within the *overlap* relation, might be meaningful, since the duration-based clustering of the intervals constrains much of the rest of the options.)

Furthermore, the main downside of pre-clustering relations into sub-relations, in addition to clustering by interval durations, is that it reduces the support for each such sub-relation to a level below the minimal threshold useful for further processing.

The main contributions of the current study are the following:

1. We described *a comprehensive architecture and process for temporal data mining* that is based on a *temporal-abstraction* process, and that introduces several fundamental concepts that define the output of the time intervals mining process, such as the vertical support, the horizontal support, and the TIV; The proposed framework overcomes several problems such as high variance in low-level data values and the common phenomenon of missing data values, by first applying temporal abstraction, and then performing a process of time interval mining.
2. We demonstrated the use of *an extended version of all of Allen's [1] temporal relations*, which introduces the use of flexible, more robust, *fuzzy constraints* to define in an internally coherent fashion all temporal relations by an epsilon value. We thus introduced a mutually exclusive robust definition of all of Allen's seven basic temporal relations.
3. We have introduced *KarmaLego - an algorithm for fast time interval relation pattern mining*, which, when compared to existing TIRPs mining approaches, *directly extends* TIRPs during the candidate generation phase, and exploits fully the *temporal relations transitivity properties*, to make the candidate generation process significantly more efficient. We demonstrated the efficiency of KarmaLego by comparing its performance to that of several other TIRP mining methods. Note that the KarmaLego algorithm handles any number of temporal relations used during the enumeration process, and is oblivious to the degree of fuzziness (i.e., the epsilon value) used in their definition.
4. We considered the lack of metric specification regarding the time intervals durations in the otherwise non-ambiguous definition of TIRPs and the problem of discovering TIRPs that significantly vary with respect to the duration of their supporting instances. For that purpose, we proposed to pre-cluster the symbolic time intervals into k clusters. We demonstrated this process and its potential tradeoff using several real datasets. The use of pre-clustering indeed increases the homogeneity of the instances, but results also in the reduction of the number of discovered frequent TIRPs. A potential area for future research is to cluster also the temporal *relations*, such as the gap in the Before temporal relation, but our analysis suggests that the need for that should significantly decrease after a preprocessing step of clustering by temporal *durations*.

Acknowledgements. The authors wish to thank Panagiotis Papapetrou for sharing datasets and insightful discussions, to Christos Faloutsos, Christian Freksa, Frank Hoppner, Fabian Murchen and Dhaval Patel for insightful discussions about time intervals mining.

This work was supported in part by grants from Deutsche Telekom Laboratories, HP labs Innovation Research Program, Award No. 2008-1023, and the EU 7th Framework MobiGuide project, grant No. 287811.

References

1. J. F. Allen. Maintaining knowledge about temporal intervals, *Communications of the ACM*, 26(11): 832-843, 1983.
2. J. Ayres, J. Gehrke, T. Yiu, and J. Flannick., Sequential PAttern Mining Using Bitmaps. In *Proceedings SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, July 2002.
3. R. Azulay, R. Moskovitch, D. Stopel, M. Verduijn, E. de Jonge, and Y. Shahar ,Temporal Discretization of medical time series - A comparative study, *IDAMAP 2007*, Amsterdam, The Netherlands, July 2007.
4. C. Freksa, Temporal reasoning based on semi-intervals, *Artificial Intelligence*, vol. 54, 1, pages 199 – 227, 1992.
5. F. Höppner, Learning Temporal Rules from State Sequences, *Proceedings of WLTSD-01*, 2001.
6. F. Höppner: Time series abstraction methods -- A Survey., *Workshop on Knowledge Discovery in Databases*, Dortmund, pp. 777-786, Sept/Okt. 2002
7. V. R. Jakkula, D. J. Cook, Detecting Anomalous Sensor Events in Smart Home Data for Enhancing the Living Experience, *Artificial Intelligence and Smarter Living'11*, 1--1, 2011
8. P. S. Kam and A. W. C. Fu, Discovering temporal patterns for interval based events, In *Proceedings DaWaK-00*, 2000.
9. N. Lavrac, E. Keravnou and B. Zupan, *Intelligent Data Analysis in Medicine*, 2000
10. J. Lin, E. Keogh, S. Lonardi, and B. Chiu, A Symbolic Representation of Time Series with Implications for Streaming Algorithms, In *8th ACM SIGMOD DMKD workshop*, 2-11, 2003.
11. F. Mörchén, and A. Ultsch, Optimizing Time Series Discretization for Knowledge Discovery, In *Proceeding of KDD05*, 2005.
12. F. Mörchén, Algorithms for Time Series Knowledge Mining, *Proceedings of KDD-06*, 2006.
13. R. Moskovitch, D. Stopel, M. Verduijn, N. Peek, E. de Jonge, and Y. Shahar, Analysis of ICU Patients Using the Time Series Knowledge Mining Method, *IDAMAP 2007*, Amsterdam, The Netherlands, July 2007.
14. R. Moskovitch, Y. Shahar, Medical Temporal-Knowledge Discovery via Temporal Abstraction, *AMIA 2009*, San Francisco, USA, 2009.
15. R. Moskovitch, N. Peek, Y. Shahar, Classification of ICU Patients via Temporal Abstraction and temporal patterns mining, *IDAMAP 2009*, Verona, Italy, 2009.
16. P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos, Mining frequent arrangements of temporal intervals, *Knowledge and Information Systems*, 21 (2), 133-171, 2009.
17. D. Patel, W. Hsu, M L Lee, Mining Relationships among Interval-based Events for Classification, *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 393-404, 2008.

18. J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth," Proc. 17th Int'l Conf. Data Eng. (ICDE '01), pp. 215-224, 2001.
19. J. Roddick and M. Spiliopoulou, A survey of temporal knowledge discovery paradigms and methods. *IEEE Transactions on Knowledge and Data Engineering*, 4(14):750–767, 2002.
20. L. Sacchi, C. Larizza, C. Combi, and R. Bellazi. Data mining with temporal abstractions: learning rules from time series. *Data Mining and Knowledge Discovery*, (15):217–247, 2007.
21. Y. Shahar, A framework for knowledge-based temporal abstraction, *Artificial Intelligence*, 90(1-2):79-133, 1997.
22. Y. Shahar, (1998). Dynamic temporal interpretation contexts for temporal abstraction. *Annals of Mathematics and Artificial Intelligence* 22 (1-2) 159-192.
23. Y. Shahar, (1999). Knowledge-based temporal interpolation. *Journal of Experimental and Theoretical Artificial Intelligence* 11, 123-144.
24. Y. Shahar, H. Chen, D. Stites, L. Basso, H. Kaizer, D. Wilson, and M.A. Musen (1999). Semiautomated acquisition of clinical temporal-abstraction knowledge. *Journal of the American Medical Informatics Association* 6(6), 494-511.
25. M. Verduijn, L. Sacchi, N. Peek, R. Bellazi, E. de Jonge, B. de Mol. Temporal abstraction for feature extraction: A comparative case study in prediction from intensive care monitoring data. In *Artificial Intelligence in Medicine*, 41, 112, 2007.
26. R. Villafane, K. Hua, D. Tran, and B. Maulik, Knowledge discovery from time series of interval events, *Journal of Intelligent Information Systems*, 15(1):71-89, 2000.
27. E. Winarko and J. Roddick. Armada - an algorithm for discovering richer relative temporal association rules from interval-based data. *Data and Knowledge Engineering*, 1(63):76–90, 2007.
28. S. Wu, Y. Chen, Mining Nonambiguous Temporal Patterns for Interval-Based Events, *IEEE Transactions on Knowledge and Data Engineering*, Vol 19 (6), 742-758, 2007.