KarmaLego - Fast Time Intervals Mining

Robert Moskovitch* Yuval Shahar†

Abstract

As part of the task of knowledge discovery from multivariate time-oriented data, the variety of temporal data types, such as time-point series having varying frequencies and time intervals having varying durations, presents a challenging problem in terms of the integration of the data. Recently the use of temporal abstraction, in which time series are abstracted into symbolic time intervals representation and the time intervals are then mined to find frequent symbolic Time Intervals Related Patterns (TIRPs), was proposed. Time interval mining is a relatively young research field developed in the present decade. Most of the existing methods use candidate generation methods to enumerate all the possible patterns in the database. A great challenge is the scaling of the methods because of the large number of potential candidates. We formalize the time intervals mining task from multivariate time series via temporal abstraction, and then introduce KarmaLego, a novel time intervals mining algorithm, which expands TIRPs directly through candidate generation using a more flexible Allen's temporal logic; we then present a significantly faster version, KarmaLegoF, which generates only relevant candidates by exploiting the transitivity of time intervals. The performance of the KarmaLego algorithms was evaluated rigorously on various real life datasets in comparison to other existing state-of-the-art methods, and was found to be faster, especially in large datasets and at low minimal support levels. Finally we demonstrate the usefulness of the method in several potential applications.

1 Introduction

The reduction in data storage costs and the increasing use of sensor loggers presents an exceptional opportunity to discover temporal knowledge [14] in many domains, such as medicine [15], interpretation of data from smart houses [7], and more. Multivariate time-oriented data are available in many domains and in varying formats (e.g., time points at fixed or variable frequencies, or varying durations of time intervals). To integrate these varying types of time-oriented variables to make temporal knowledge discovery possible presents a great challenge. One of the methods used to facilitate the discovery is the conversion of time-stamped series into a time-interval series representation. Conversion of a time series to a time interval series has several advantages, such as a reduction in the amount of data to process and a compact representation of the data. Furthermore, to enhance comprehensibility and stability of the intervals, and facilitate meaningful summaries of the data, an abstraction of the raw data into higher level concepts (e.g., levels of anemia instead of Hemoglobin values) is often performed. The latter task is often referred to as *temporal abstraction* [16].

Previous studies suggested converting numerical time series to symbolic time intervals by discretization [5, 9, 10] or performance of temporal abstraction using domain-specific knowledge [16]. Mining time intervals has been attracting growing attention over the recent 10 years [6, 8, 11, 12, 13, 19, 20], inspired often by sequential mining [3] and the a priori principle [1], on which we will elaborate later. However, while sequential mining, in which instantaneous events are mined to discover sequences of events was substantially researched [3], it is actually a private case of time intervals mining, in which time intervals are instantaneous and the only relation allowed among them is *before*.

The relatively large amount of possible relations among time intervals (unlike in instant events) creates a great challenge in scaling these mining methods. Our long-term goal is to discover temporal knowledge from time-oriented data via temporal abstraction; however, in this paper we focus on the efficient and fast mining of time intervals. Our main contributions in this paper are:

- A formal definition of the problem of discovery of temporal knowledge from varying types of time-oriented data via temporal abstraction
- karmaLego - A novel time intervals mining algorithm that expands TIRPs directly
- KarmaLegoT - An efficient algorithm which does NOT generate wrong candidates, based on the transitivity property
- A novel measure of the temporal patterns, called horizontal support, which refers to the number of its occurrences in each entity
- Demonstration of the usefulness and application of the use of temporal patterns for temporal knowledge discovery, hierarchical temporal clustering and classification

*Department of Information Systems Engineering, Ben Gurion University, Beer Sheva, Israel

†Department of Information Systems Engineering, Ben Gurion University, Beer Sheva, Israel

We start by providing the relevant background to our paper, including temporal knowledge representation. In section 3 we formalize the problem of temporal knowledge discovery using temporal abstraction and time interval mining. In section 4 we present our method KarmaLego in detail. In sections 5 we present our experiments and results, in 6 the usefulness of the method and in 7 a survey of previous studies in time intervals mining. Finally in section 8 we discuss our results and future work.

2 Temporal Knowledge Representation

Temporal knowledge representation is an essential tool in the task of temporal data mining, which influences the entire mining process. Time intervals are often defined as a triple $ti =< ti_{start}, ti_{end}, ti_{symbol} >$, having a start-time, end-time, and a symbolic value. In order to represent temporal knowledge consisting of time intervals, Allen's thirteen temporal relations [2] are often used. In Allen's logic, all the possible relations between two time intervals are defined, including *before, meets, overlaps, starts, during, finishes*, their inverse and *equal*, as shown in Figure 3.

However, in the context of time interval patterns mining, these relations suffer from several disadvantages, as pointed out by Moerchen et al. [12]: Robustness: they are not robust, since some of them require the precise equality (or lack thereof) of two or more interval endpoints. For example, the relations *overlap, during* and *finishes* can describe a very similar situation, as shown in examples 1.a,b,c in Figure 1, which illustrate the relations between two time intervals A and B. Ambiguity: the same Allen relation can represent intuitively very different situations. Examples 2.a,b,c in Figure 1 illustrates the different situations that are all defined as overlap, while the different overlaps may have different meanings.

To overcome the robustness criticism, we use an epsilon value to define more flexible relations, as proposed in [13]. However, we apply this approach to all of Allen's basic seven temporal relations as shown in Figure 3. For example, when epsilon > 0 the relation meet includes all scenarios from a slight gap (before) to a slight overlap between the intervals. The definitions in Figure 3 are constructed such that they imply mutually exclusive temporal relations. That is, each scenario can only be classified into a single temporal relation. Note that the same epsilon is used for all relations. We added an eighth relation, *starts* (the inverse of started-by), required only when epsilon > 0, since in this case the second interval (B) can be after the first (A), to disambiguate the lexicographic ordering of the intervals, as we later define in Section 3 (Definition 8). For the ambiguity limitation we suggest to perform preclustering of the time intervals as we discuss later, but do not demonstrate here.

In order to represent a time intervals pattern without ambiguity all the pairwise relations of each pair of intervals in the pattern have to be defined [6]. Thus, given a *k* sized temporal pattern (having *k* intervals) we need k·(k-1)/2 relations. To clarify the justification for defining all the relations we will use example (b) in Figure 2. Defining it only by A o B & B o C is ambiguous since it may fit to several options of relations between A and C. Thus, in order to define a pattern with no ambiguity we have to define a conjunction of all the pairwise relations, as we will define formally later, which in the case of Figure 2.b is A o B & B o C & A $<$ C.

Since there are often several types of relations (e.g., *before*, *meet*,..) the amount of potential generated TIRPs grows exponentialy with k. To be more specific, given $\Re$ the set of the temporal relations the amount of potential candidates is $\|\Re\|^{k\cdot(k-1)/2}$, which creates a great challenge in scaling the mining algorithms. As we shall see the computational implications which we aim to solve in the algorithm proposed in this paper. Suffice it to say at this point that exploiting transitivity can often save considerable computation (see Fig 2).

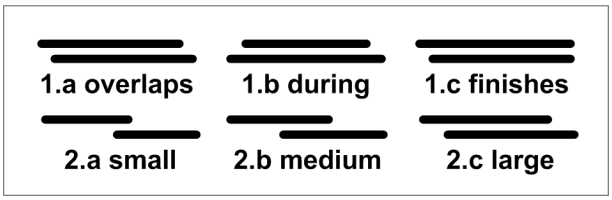

Figure 1: Examples 1.a,b,c illustrate the lack of robustness in Allen's relations, in which overlap, during and finishes represent a very similar situation. Examples 2.a,b,c demosttrate the ambiguity in Allen's temporal relations. (Modified from [12]).

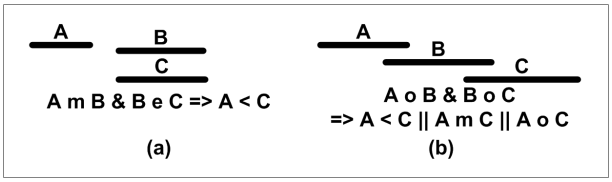

Figure 2: Demonstration of the ambiguity results of defining only the relations among adjacent intervals. Using the transitive property within temporal relations: in the case of (a), the transitive property can be used to infer that A$<$C, while in (b) several options exist between A and C: before, meet or overlap.

3 Definitions and Problem Formulation

We start by defining the temporal framework we are using and the problem of mining time intervals. Definitions 1 to 4 and 7 refer to the first stage of abstracting time points data into time intervals, while the rest refers to the task of time

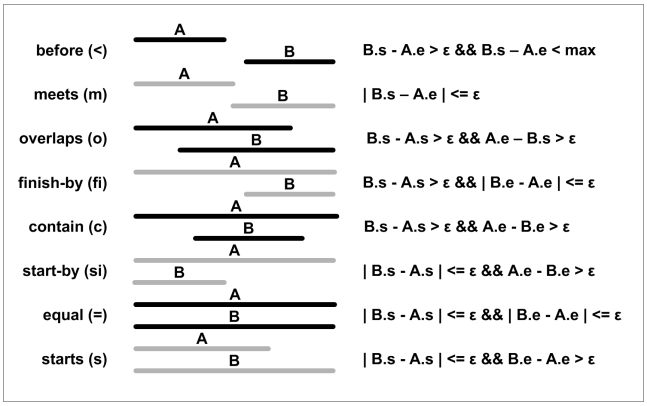

Figure 3: A flexible extension of Allen's seven relations using the same Epsilon value for all relations. The [eighth] starts relation is required when epsilon > 0.

intervals mining.

DEFINITION 1. *Time stamps, $\tau_i \in T$, comprise the basic primitives of time.*

DEFINITION 2. *A time interval, I, is an ordered pair of time stamps that denotes the endpoints, (I_{start}, I_{end}), of the interval.*

DEFINITION 3. *Given a list of entities E = $\{e_1, e_2, e_3,..,e_n\}$ (e.g., a list of patients), a variable $\pi \in \Pi$ is either a measurable (raw) V or a derived (abstract) value s ∈ S of an entity $e_i \in E$ to which the data are referring (e.g., a patient). Variables have various properties, such as a domain Vπ of possible numeric or symbolic values and measurement units.*

DEFINITION 4. *A time series TS = $\{ts_1, ts_2, .., ts_n\}$, which represents a series of time stamped data referring to the relevant entity, where ts = $< e_i, \pi, v, \tau >$, in which e_i is the relevant entity, Π is the dependent variable of the time series, π is the time stamp at which it is measured and v is its value at that time stamp. The values can be mapped to time stamps in varying gap durations, such that: $\forall i(\tau_i < \tau_{i+1}) \wedge ((\tau_{i+1} - \tau_i = \tau_i - \tau_{i-1}) \vee (\tau_{i+1} - \tau_i < \tau_i - \tau_{i-1}) \vee (\tau_{i+1} - \tau_i > \tau_i - \tau_{i-1}))$*
A private case is the uniform sampled time series, in which the gaps duration are fixed, such that $\forall i(\tau_i < \tau_{i+1}) \wedge (\tau_{i+1} - \tau_i = \tau_i - \tau_{i-1})$

DEFINITION 5. *A symbolic time interval SI = $< e_i, \pi, I, s >$ represents the predicate that the symbolic value s ∈ S (e.g., High or Low) of variable π holds during interval I for entity e_i.*

DEFINITION 6. *A symbolic time interval series, TIS = $\{SI_1, SI_2, .., SI_n\}$, where each SI is a symbolic time interval, represents a series of symbolic intervals. Typically (i.e., in a database) each symbolic time interval can refer to a different entity.*

DEFINITION 7. *An abstraction function $\theta \in \Theta$ is a function that takes as input a time series TS and returns a symbolic time interval series TIS = θ(TS) = $\{sI_1, , sI_n\}$, which is a non overlapping symbolic time interval series.*

DEFINITION 8. *A lexicographic time interval series is a time interval series, sorted in the order of the start time, end time, and a lexicographical order of the labels, TIS = $\{sI_1, sI_2, .., sI_n\}$, where $sI_i = < \pi_i, s_i, I_i >$ and $\forall sI_i, sI_j \in TIS(i < j) \wedge ((sI_{i.start} < sI_{j.start}) \vee (sI_{i.start} = sI_{j.start} \wedge sI_{i.end} < sI_{j.end}) \vee (sI_{i.start} = sI_{j.start} \wedge sI_{i.end} = sI_{j.end} \wedge sI_{i.symbol} = sI_{j.symbol}))$*

DEFINITION 9. *A non ambiguous lexicographic Time Intervals Relations Pattern (TIRP) P is defined as P = $\{\overline{I}, \overline{R}\}$, where $\overline{I}$ = $\{sI_1, sI_2, .., sI_k\}$ is a set of k symbolic time intervals ordered lexicographically and $\overline{R}$ = $\prod_{i=1}^{k} \prod_{j=i+1}^{k} r(sI_i, sI_j)$ = $\{r_{1,2}(sI_1, sI_2), r_{1,3}(sI_1, sI_3), .., r_{1,k}(sI_1, sI_k), r_{2,3}(sI_2, sI_3), r_{2,4}(sI_2, sI_4), .., r_{2,k}(sI_2, sI_k),.., r_{k-1,k}(sI_{k-1}, sI_k)\}$ defining all the temporal relations among each of the (k^2-k)/2 couples of symbolic time intervals in $\overline{I}$. (A convenient graphical presentation of a TIRP is attained by using half matrix, as shown in figure 5 and throughout the paper).*

DEFINITION 10. *Given a symbolic time interval series referring to a list of $\|E\|$ entities (e.g., a number of patients) described by symbolic time intervals series, the vertical support of a TIRP P is denoted by the cardinality $\|E^P\|$ of the set E^P of distinct entities in which P is found at least once, divided by the number of the entities: ver_sup(P) = $\|E^P\|$ / $\|E\|$*

DEFINITION 11. *Given a symbolic time interval series TIS_i referring only to one entity e_i (e.g., a patient's record), the horizontal support of a TIRP P in entity e_i, hor_sup(P, e_i), is the number of instances of TIRP P found within e_i's time intervals series TIS_i.*

DEFINITION 12. *The mean global horizontal support of a TIRP P is the average of the horizontal support of P for all the entities E: mean_global_hor_sup = $\frac{\sum_{i=1}^{|E|} hor\_sup(P,e_i)}{|E|}$. When the set E is restricted to the set E^P, we refer to the mean local horizontal support of P, MLH_Sup(P), since the horizontal support is computed only for entities in which P was found at least once.*

3.1 Problem Definition Given a set of entities *E* described by varying symbolic time intervals series *TIS*, and a *minimum vertical support* threshold *min_ver_sup*, the goal is to find all the TIRPs above the *min_ver_sup* threshold.

3.2 An Example of Mining Time Intervals In order to introduce the problem of time intervals mining, its challenges, and the flavor of our solutions, we provide here an example of a database and of a mining process. Table 1 presents an example database of four entities represented by a list of time intervals ordered lexicographically (def 8), in which, for simplicity's sake, we represent all of the possible combinations of variable and symbolic values as three symbols: A, B and C. We are describing here a mining process, given a 90% minimum vertical support. During the mining process, we generate longer and longer sequences of symbols, in which each pair of symbols is related by one or more of the possible temporal relations, referring to supporting instances. If the support is above the pre-defined minimal vertical support, the sequences are expanded accordingly. The

Table 1: A database example.

e_id	Time-interval-series
1	A[1,4], B[2,5] C[6,8], B[8,10]
2	A[6,8], B[7,9], A[12,14], C[10,11]
3	C[1,4], A[5,7], B[6,8], B[9,11], C[12,13]
4	C[1,2], A[2,4], B[3,7], A[8,10], B[11,13]

first step is to count the vertical support of each symbol, representing 1-sized TIRPs. As shown in Figure 4, at the top, all the symbols appear in all the entities, thus having 100% vertical support. The next step is to expand the sequence of symbols and relations. For each symbol we add all the symbols (including itself) and check all of the optional relations among them. For example, A-A (the relation between two instances of A) can be before, meet, etc. In this example, we found instances only of A $<$ (before) A, with only a 50% vertical support. When adding B to A, we found only instances of A $<$ B and A m (meet) B among the other eight optional relations.

Figure 4 illustrates the instances supporting all the existing pairs and their vertical support in parentheses. Once we have all of the 2-sized TIRPs, we try to expand them to 3-sized TIRPs. Since the only TIRP above the minimal vertical support is A o B, we will generate all of its possible expansions. However, according to def 9, we must generate TIRPs that are non-ambiguous, since a TIRP is defined by a conjunction of all of its pairwise relations between all of its corresponding time intervals.

Figure 5(a) presents the half matrix representation of the conjunction of the pairwise relations in the TIRP A o B. Figure 5(b) presents the half matrix defining the conjunction of relations for the extended TIRP in the case of adding an interval of type C. In this case, two additional relations should be defined: the relation B $r_{2,3}$ C and the relation A $r_{1,3}$ C, which are actually the relations between the previous intervals (A, B) and the new interval (C). This will generate 8^2=64 potential expanded TIRPs, in our case of eight possible relations.

While this method of expanding directly TIRPs by joining the last interval with the corresponding 2-sized TIRP seems strait forward and intuitive it was never proposed before and is crucial for scaling purposes, as will be shown in the evaluation section. This method of expanding the sequence by directly referring to the instances of the prefix of the sequence is one of the contributions of this paper, which enables us to expand TIRPs directly and search only on expansions to previous supporting instances. As we shall see, there is a much more efficient way of implementing this method, using the transitivity of the temporal relations, which is another contribution of this paper. We will elaborate on this in the description of KarmaLego and KarmaLegoT. In Figure 5 we found the definition of the expanded TIRP which was found in the example data, which is defined by AoB & B$<$C & A$<$C. To discover the supporting instances we start with the supporting instances of A o B and we search for the instances in B $<$ C, in which the first interval is the same as the last in A o B. Now we only need to verify that A $<$ C contains the fitting instances.

Figure 5 presents the half matrix graphical representation of a TIRP, which illustrates our Incremental Direct Expansion, which enables us to expand directly TIRPs consisting of the concept that an expansion of a TIRP requires the definition of the relations between the new time interval and the previous intervals.

4 KarmaLego

Having provided an example of the mining problem and generally our approach and a brief introduction to the details of an example of a class of methods that will be compared to it [13, 20], we now present KarmaLego, a new algorithm for fast mining of time interval related patterns. Our method consists of an enumeration tree, which is efficient and faster than other mining methods, because it first enumerates the frequent 2-sized TIRPs and expands them directly. Figure 12 illustrates the TIRPs tree which is created through the mining process in KarmaLego. Eventually each TIRP is described by its *vertical* and *mean horizontal* support.

We present here a pseudo code of the KarmaLego algorithm corresponding to the Incremental Direct Expansion principle we described earlier. The input to the algorithm is a database db of *n* entities E = $< e_1, e_2, .., e_n >$. Each $e_i = < I_1, I_2, .., I_k >$ entity is described by a list of time intervals ordered lexicographically (def 8). Additionally a TIRP is a structure, including a sequence of symbolic time intervals I, a halfmatrix R defining the relations in the TIRP, and a list of the supporting instances E^{TIRP}.

Algorithm 4.1 describes the main routine of KarmaLego. First, the symbols having above minimal vertical support are selected from the db constructing the first level of the tree T^1, through a single scan of the db. For each entity *e* in the db, all the pairs of intervals, are indexed according to the symbols

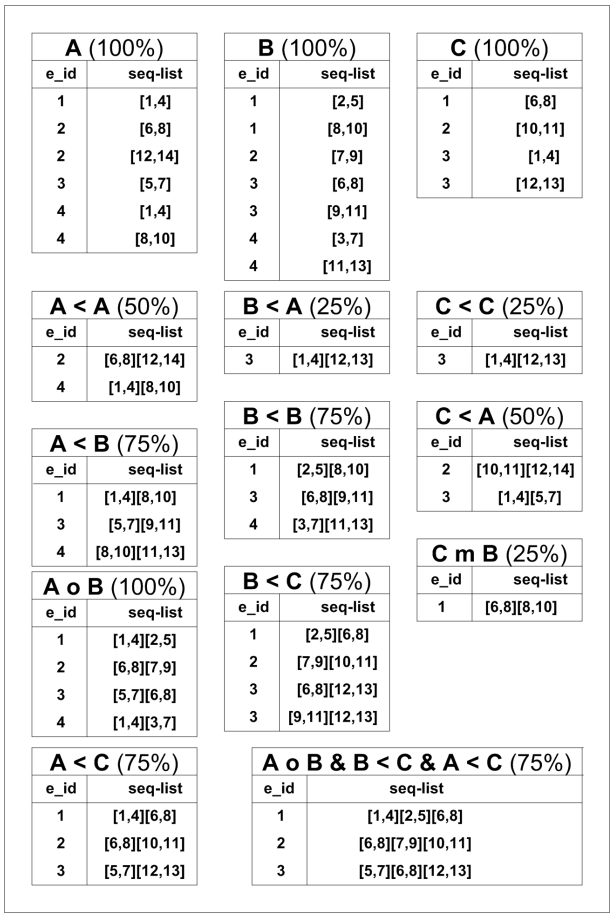

Figure 4: Enumeration of the time interval database in Table 1, starting with the symbols and extending to the 2 time intervals TIRPs and 3 time intervals TIRPs.

and the temporal relation to the second level of the tree T^2. Finally, each 2-sized TIRP t^2 is expanded by Expand_TIRP, which receives a reference to T^2.

ALGORITHM 4.1.
KarmaLego(db, mver_sup, e)
1 $T^1 \leftarrow$ obtain symbols above mver_sup
2 foreach e $\in$ db
3 for i=0 to $\|e.I\|$
4 for j=i to $\|e.I\|$
5 r = find_relation(e.I*i*,e.I*j*)
6 $T^2 \leftarrow < e.Ii, r, e.Ij >$
7 foreach $t^2 \in T^2$
8 Expand_TIRP(T^2, 2, t^2)

Algorithm 4.2 presents the procedure that expands a TIRP directly. This procedure expands a frequent TIRP from a previous level. Thus, given a k-sized TIRP, it expends it to a k+1-sized optional TIRPs by adding the new symbolic

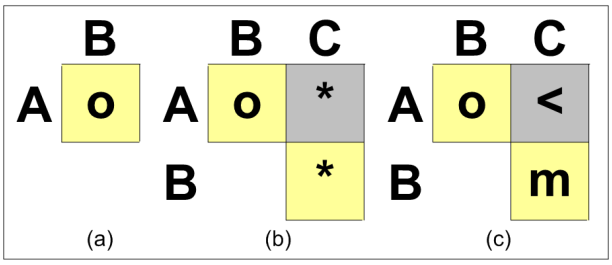

Figure 5: A half matrix representation of the relations in a TIRP A o B and its potential extensions.

interval and generating all the corresponding TIRPs based on all the optional temporal relations. However, only the relation *r* among the new symbolic (k+1) interval and the previous (k) interval is set. Later the additional relations among the new symbolic (k+1) interval and the previous intervals (1 to k-1) will be generated in the procedure MineCombinations. The reason why the first r is set and the rest of the relations are later is for the later efficient method presented in KarmaLegoT.

ALGORITHM 4.2.
Expand_TIRP(T^2, k, TIRP)
1 foreach s $\in$ S
2 I $\leftarrow$ s
3 foreach r $\in \Re$
4 R[k,k] $\leftarrow$ r
5 Mine_Combinations(T^2, k+1, TIRP)

Algorithm 4.3 describes the procedure which generates the relations corresponding to the expanded TIRP generated by Expand_TIRP. First, all the combinations (*combs*) of R corresponding to the new symbol are generated using the procedure GenerateCombs, which is $\Re^k$, where R is the halfmatrix and in our case $\|\Re\|$ is 8 when epsilon $>$ 0 (for the last column of the R half-matrix). Then, for each combination the existence of the relation between the new symbolic interval and the previous one is searched at the second level T^2, based on the symbolic intervals and the relation r (line 3). For each instance supporting the expanded TIRP the instances in t^2, which are actually TIRPs themselvs, are searched. Note that there might be more than one corresponding instance which will discover multiple instances of the same TIRP for an entity, which we measure with the horizontal_support (line 5). Then the relations to the previous intervals in the TIRP are checked as to whether it fits the new symbolic interval; if not it is removed from the supporting instances list (line 8). Finally, if above ver_min_sup amount of (distinct) entities supporting instances are found, the TIRP is further expanded.

ALGORITHM 4.3.
MineCombinations(T^2, k, TIRP)
1 combs = GenerateCombinations(R, k)
2 foreach comb $\in$ combs
3 $t^2 \leftarrow T^2$ <I[k-1],r,I[k]>
4 foreach ins $\in$ TIRP.E
5 foreach (ins.id == t^2.E.id)
5 ins.I[k-1] $\leftarrow$ t^2.E.secondSymbol
6 if (any relation of ins.I[k] NOT fit R)
7 Remove ins from TIRP.E
8 if ($\|$TIRP.E$\| >$ mver_sup)
9 Expand_TIRP(T^2, k+1, TIRP)

4.1 KarmaLegoT We described the KarmaLego algorithm, consisting of candidate generation, in which the TIRPs are expanded directly. However, while KarmaLego presents an efficient direct expansion, there is still much room for improvement. Note that when generating all the optional combinations of relations for the expanded candidate TIRPs in KarmaLego, some of the combinations might be logically inconsistent.
For example, expanding the TIRP 'A before B' to the sequence A, B, C, we have to generate all the optional combinations of the relations: $r_{1,3}$(A, C) and $r_{2,3}$(B, C), which will amount to $\|\Re\|^2$ options, which in our case is 8^2 (for epsilon > 0). Setting $r_{1,3}$(A, C) to *before* contradicts the setting, for example, the relations *meet*, *overlap*, *starts*, *finishes* and the rest, out of *before* at $r_{2,3}$(B, C). Generating these candidates causes an unneeded search for patterns which cannot exist in the data. Obviously having the ability to *not* generate these combinations will reduce the wasted effort and exhaustive search of TIRPs significantly.

4.1.1 Using the Transitivity of Temporal Relations We propose to exploit the transitivity property of temporal relations among time intervals to avoid generating combinations which are logically wrong. Given the previous example, knowing the relations among $r_{1,2}$(A, B) and $r_{2,3}$(B, C), we can reason on the optional relations in $r_{1,3}$(A, C). Figure 2 presents two corresponding examples to reason the relation $r_{1,3}$(A, C), given $r_{1,2}$(A, B) and $r_{2,3}$(B, C). In Figure 2.a only a single optional relation can be set to $r_{1,3}$(A, C), which is *before*, but in Figure 2.b there is a disjunction of three optional relations: *before*, *equal* or *overlap*. We are employing the transitive property to generate only true combinations by reasoning. When expanding a TIRP of size k there are k optional relations which generate naively $\Re^k$ candidate expanded TIRPs. However, by setting first the last relation (Algorithm 4.2, line 4) to relation r, the relation between the k+1 interval and the previous intervals is reasoned based on the transitive property. In order to implement this reasoning mechanism, we need a function that receives two temporal relations: the relation between the first two intervals (e.g., A and B) and the relation between the second and the third (e.g., B and C), which will return a disjunction of the optional temporal relations between the first and third intervals (e.g., A and C).

4.1.2 Transition Table We use Allen's [2] transition tables, which were later extended by Freksa [4] for temporal reasoning. Freksa introduces a graphical presentation of Allen's relations based on a conceptual neighborhood criterion. In our implementation, we loaded the transition table to a two dimensional array, which was used to look for the optional disjunction of relations given two relations. To explain better the reduced candidate generation using the transitive property we refer to the example in Figure 6, in which a 4 sized TIRP including the intervals A_1,B_1,C,B_2 and its relations defined in the half matrix (on right) in the shaded part are presented. The TIRP's expansions are generated by first

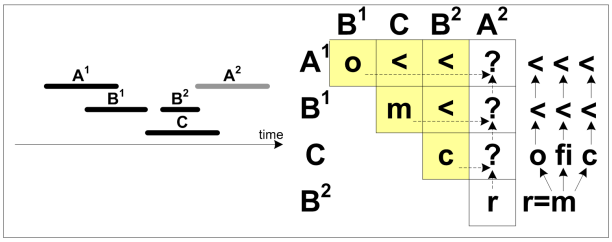

Figure 6: An illustration of the reduced candidate generation using the transitive property.

setting the additional interval A_2 related to the last interval B_2 by *r*, which in the example is set to *meet*. Then, based on the relation $r_{3,4}$(C,B_2)=c and $r_{4,5}$(B_2,A_2) it is reasoned that $r_{3,5}$(C,A_2) can be o, f_i or c. Similarly, we proceed and reason according to the arrows, relying on the previously reasoned relation and the first relation on the left (in the matrix) which in these cases is always the relation $<$. Thus, instead of having 8^4=4,096 generated candidates (after setting *r* relation to *meet*), we get only 3 real candidates.

5 Experiments

To evaluate the effectiveness and efficiency of KarmaLego in comparison with the H-DFS and Armada methods, we executed an extensive performance study of three real datasets with various kinds of sizes and data distribution. All experiments were conducted on a 3 Ghz Intel Xeon server, having 3 Gb main memory, running Microsoft WindowsServer2003. Since the methods were not available, we implemented the three methods using Microsoft Visual C++ 6.0: H-DFS according to [13], ARMADA [20] and KarmaLego in the naive candidate generation and with the transitive property exploiting candidate generation called KarmaLegoT. The methods' output was compared to make sure it was identical. In addi-

tion to the runtime period given a min_ver_sup level evaluation, we also provide an experiment on the ASL data which demonstrates the influence of the epsilon variable on the amount of discovered TIRPs and the distribution of the 2-sized TIRPs.

5.1 Datasets We used three real datasets for the evaluation, which we characterize by four aspects: the entire amount of intervals in the dataset *N*, the number of symbols *S*, the amount of entities *E*, and the average amount of intervals per entity *I*, as shown in Table 2.

5.1.1 American Sign Language The ASL database created by the National Center for Sign Language and Gesture Resources at Boston University [13], consisting of a collection of 884 utterances, where each utterance associates a segment of video with a detailed transcription. Every utterance contains a number of ASL gestural and grammatical fields (e.g., eye-brow raise, head tilt forward, wh-question), each one occurring over a time interval.

5.1.2 Diabetes The diabetes dataset, provided by our Soroka academic hospital (can be provided by the authors), contains data on diabetic patients collected over five years containing six measurements and records of a dozen types of medication taken by the patients every month. The measurements were abstracted into three states, which were given by the physicians, while the medications were abstracted into ten states using Equal Width Discretization (EWD), which divides equally the value ranges of each medication into ten discrete states.

5.1.3 MavLab Smart House The MavLab SmartHome dataset, provided by [7], contains data describing the activity of 99 sensors installed in a computerized apartment, and their readings. The sensors described the activity of various appliances scattered around the apartment and were sampled each 1-second a period of 81 days, where only the first 3 hours data of each day were used. Since these variables are given as time series, we abstracted them into three states, using EWD. When referring to abstraction we mean that the time series were discretized and adjacent time points having the same symbol were concatenated.

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**

5.2 Experiments and Results We start with a runtime evaluation and proceed with experiences with various levels of min_ver_sup and epsilon. Note that each dataset presents different challange. The ASL is relatively small, the Diabetes is has the largest amount of intervals (N) and entities (E) and the SmartHome has the largest mean of intervals per entity (I).

5.2.1 Runtime Experiments Since the runtime of the H-DFS method is significantly slower than that of the other methods, which requires a logarithmic graph, for the clarity of presentation we present it only once using the ASL dataset, which is relatively small.

ASL: We ran all four methods on the ASL dataset on 10 levels of ver_min_sup. Figure 7 presents the ASL results in a logarithmic presentation, in which Armada and KarmaLegoT are behaving very similarly, while KarmaLego is slower and H-DFS is significantly slower.

Diabetes: Experiments on the Diabetes dataset (Figure 8),

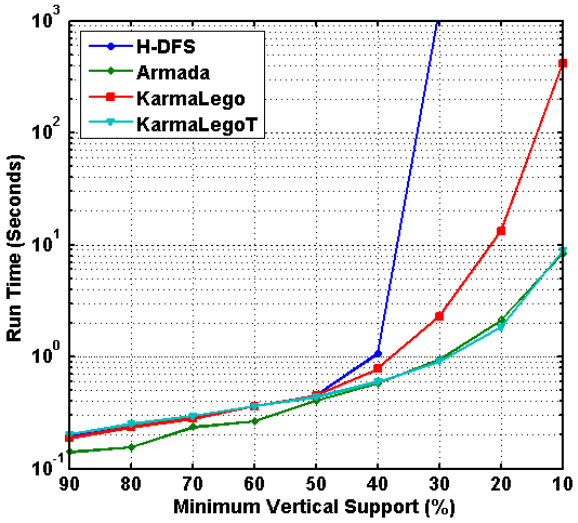

Figure 7: On the relatively small ASL dataset, the KarmaLegoT and Armada are very similar, followed by KarmaLego and the H-DFS which is significantly slower. Note that this is a logarithmic presentation.

in which N and E are the largest among the other datasets, show that KarmaLegoT is twice as fast as Armada at low levels of ver_min_sup. Note that the low levels of suport are the meaningful, since many times at high levels there are no TIRPs and sometimes we are acutally interested in the low levels to discover TIRPs which are not common to the entire population of the mined database.

MavLab Smart Home: Experiments on the SmartHome dataset (Figure 9), which is characterized by the largest I, show that KarmaLegoT is significantly faster than Armada. This can be explained by the advantage KarmaLegoT has in that it first indexes the pairs of intervals, in the second level

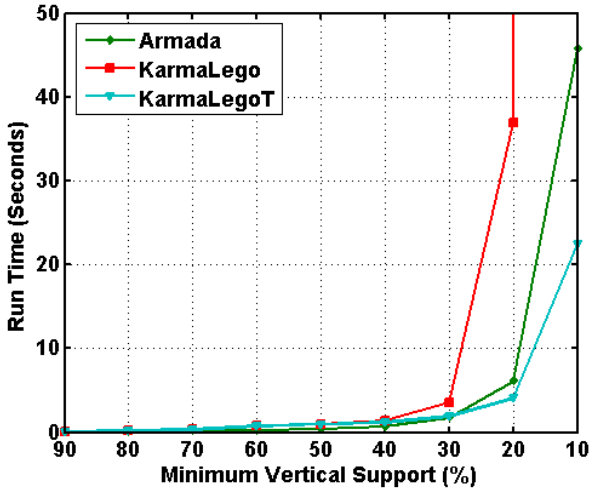

Figure 8: On the Diabetes dataset, having the largest N and E, the KarmaLegoT is significantly faster than Armada, followed by KarmaLego.

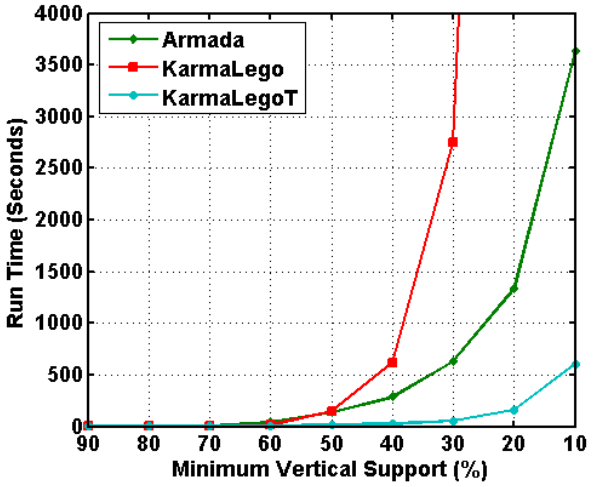

Figure 9: KarmaLegoT is significantly faster than Armada, especially in low levels of support on the SmartHome datasets, in which entities have many intervals.

of the enumeration tree, and does not search for the patterns repeatedly on the raw data as Armada does, which becomes critical here since the entities have many intervals. Note that in this case we used only 3 hours of each day in the data. Obviously with larger dataset, which we actually use, the difference would increase.

5.2.2 Experiments with the Epsilon To demonstrate the influence of the epsilon value on the discovered TIRPs, we ran the KarmaLegoT method using several settings for epsilon from 0 to 5 and for several levels of ver_min_sup and counted the amount of TIRPs discovered for each setting. Figure 10 shows that increasing the epsilon value decreases the amount of TIRPs discovered. This happens because the distribution of the temporal relations among the pairs of time intervals is changing based on the value of epsilon, as will be shown in the next figure.

Figure 11 presents the amount of 2 sized TIRPs distribution by one of the eight temporal relations, based on the varying values of epsilon and a fixed level of 10% minimum vertical support. Increasing in the value of epsilon reduces significantly the amount of *before*, *contain* and *overlap* relations, which mainly transforms to *equal*, *meet*, *finish-by* and the *start* relation. The reduction in the amount of TIRPs is not bad since the discovered TIRPs become more general and having higher level of vertical support.

6 Related Work

Mining time intervals is a relatively young research field. One of the earliest works in the area is that of Villafane et al. [19], who searches for containments of intervals in mult-symbolic interval series. Kam and Fu [8] were the first to use all of Allen's relations [2] to compose hierarchically represented interval patterns, in which the patterns are restricted to right concatenation of intervals to existing extended patterns, called *A1* patterns.

Hoppner [6] was the first to define a non ambiguous representation of Allen's-based time intervals patterns by a k^2 matrix in which all the pairwise relations within a k-intervals pattern are defined. An naive Apriori based algorithm is proposed, using also the transitivity property for pruning. Papapetrou et al. [13] presented a mining method consisting of the SPAM sequential mining algorithm [3], called H-DFS, which results in an enumeration tree that spans all of the discovered patterns. However, this method does not scale since it generates first the sequences of symbols and then all the potential relations combinations at each level separately from the begining, which does not enable to make direct expansion of the patterns possible.

ARMADA, presented recently by Winarko and Roddick [20], uses a candidate generation and mining iterations approach. First, the algorithm reads the database into memory and counts the support of each state and generates frequent 1-patterns. Then an index set of the supporting intervals for each frequent 1-pattern is constructed and later expanded for longer patterns. Finally, using a recursive find-then-index strategy, the algorithm discovers all the temporal patterns from the in-memory database. However, a limitation of this method is that it repeats searching on the raw data for each expansion step.

Moerchen [12] proposed an alternative to Allen's relations-

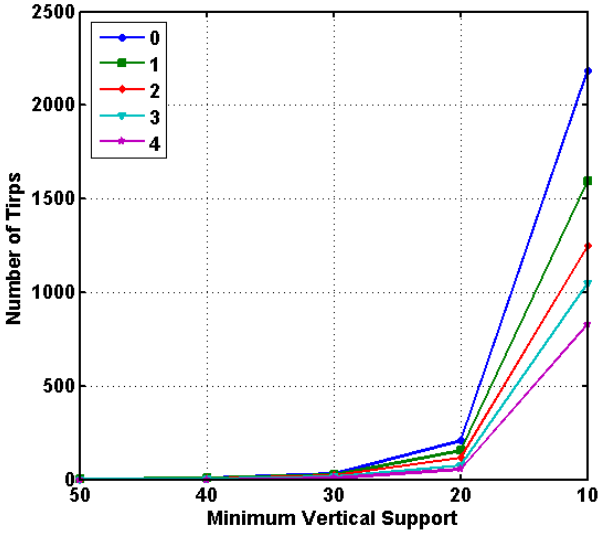

Figure 10: The number of TIRPs is increasing as the value of epsilon is decreasing

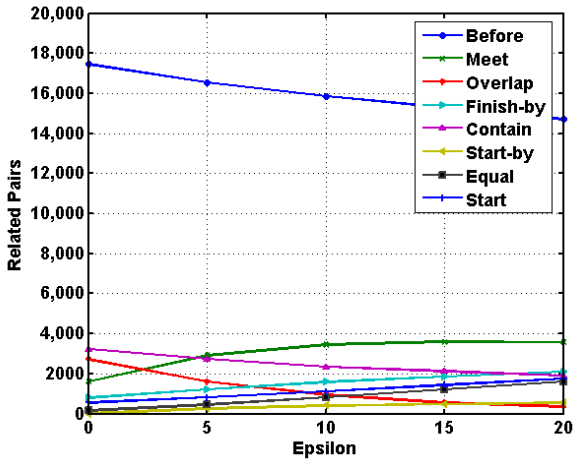

Figure 11: The increase in the value of epsilon reduces significantly the amount of *before*, *contain* and *overlap*, which are often converted into *meet*, *equal* and *start* relations.

based methods, in which time intervals are mined to discover coinciding symbolic time intervals, called Chords, and the repeating partially ordered chords called Phrases. In [17] a mining method based on mining the end points of the time intervals is proposed. In [15] abstracted time series are used to find temporal association rules by generalizing Allen's rules to a relation called PRECEDES and [7] mines time intervals to discover anomalies in Smart Home data.

7 Discussion and Conclusions

The long-term focus of our research program is knowledge discovery from multivariate time series via temporal abstraction. Our research is best viewed within the growing field of time intervals mining. Our contributions include a clear formalization of the task of temporal knowledge discovery from multivariate time oriented data via temporal abstraction, a new time-interval-mining method, called KarmaLego, which performs a direct candidate expansion, and a significantly faster and more efficient - KarmaLegoT, which exploits the transitivity of temporal relations, thus generating only relevant candidates.

We compared KarmaLego and KarmaLegoT to Papapetrou's H-DFS method and Winarko and Roddick's ARMADA algorithms. The ARMADA algorithm is relatively fast; nevertheless, in large datasets, especially those having large amount of intervals per entities such as the SmartHome dataset, and at low levels of minimal vertical support it becomes very slow. This can be explained by the fact that the search in ARMADA is performed on the actual data, while in KarmaLego and KarmaLegoT the search is performed on pairs of intervals that are indexed in the first (BFS) phase of the algorithm at the second level of the enumeration tree.

We also evaluated the influence of varying epsilon values, which control the flexibility of Allen's extended temporal relations, on the runtime performance of KarmaLegoT. A high value of epsilon is very useful, especially with the dense interval data that often result from abstracted time series, since, if a small epsilon value is used, the resulting high variability in specific TIRPs is not always an advantage. The probelm of ambiguity in the use of Allen's realtions in the task of time intervals mining is actually wider when referring to larger than 2-sized TIRPs. Note that the instances supporting a TIRP may vary much by the duration of the time intervals and the temproal relations, as shown in Figure 1. In order to overcome this we suggest to precluster the duration of each symblic time intervals of the entire dataset. Such a procedure has an advantage of discovering more specific TIRPs, in which the supporting instances are more similar and overcome the said problem. However, the tradeoff is in larger amount of symbols, potetntially having lower support, and as a result less discovered TIRPs. We are currently investigating the use of this procedure in the varying applications of temporal hierarchical clustering and classification of multivariate time series.

8 ACKNOWLEDGMENTS

We would like to thank Dr F. Morchen, P. Papapetrou, Prof J. Ullman Prof A. Porat and Prof D. Cook for insightfull discussions and for providing the datasets.

References

[1] R. Agrawal and R. Srikant. Fast algorithms for mining associations rules. In *Proceedings of VLDB*, 1994.

[2] J. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 11(26):832–843, 1983.

[3] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using a bitmap representation. In *Proceedings of SIGKDD02*, 2002.

[4] C. Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 1(54):199–227, 1992.

[5] F. Hoppner. Time series abstraction methods - a survey. In *Workshop on Knowledge Discovery in Databases*, 2002.

[6] F. Hoppner and F. Klawonn. Finding informative rules in interval sequences. *Intelligent Data Analysis*, 3(6):237–256, 2002.

[7] V. Jakkula and D. Cook. Anomaly detection using temporal data mining in smart home environment. *Methods of Information in Medicine*, (47):70–75, 2008.

[8] P. Kam and A. Fu. Discovering temporal patterns for interval based events. 2000.

[9] J. Lin, E. Keogh, L. Wei, and S. Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Mining and Knowledge Discovery*, 15(15):107–144, 2007.

[10] F. Moerchen. Optimizing time series discretization for knowledge discovery. In *Proceedings of SIG-KDD05*, 2005.

[11] F. Moerchen. Unsupervised pattern mining from symbolic temporal data. *ACM SIGKDD Explorations Newsletter archive*, 1(9), 2007.

[12] F. Moerchen and A. Ultsch. Efficient mining of understandable patterns from multivariate interval time series. *Data Mining and Knowledge Discovery*, 2(15), 2007.

[13] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Discovering frequent arrangements of temporal intervals. In *Proceedings of IEEE ICDM-05*, 2005.

[14] 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.

[15] 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.

[16] Y. Shahar. A framework for knowledge-based temporal abstraction. *Artificial Intelligence*, 1-2(90):79–133, 1997.

[17] W. Shin-Yi and C. Yen-Liang. Mining nonambiguous temporal patterns for interval-based events. *IEEE Transactions on Knowledge and Data Engineering*, 19(6):742–758, 2007.

[18] 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, 1-12*, 2007.

[19] R. Villafane, K. Hua, D. Tran, and B. Maulik. Knowledge discovery from time series of interval events. *Journal of Intelligent Information Systems*, 1(15):71–89, 2000.

[20] 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.

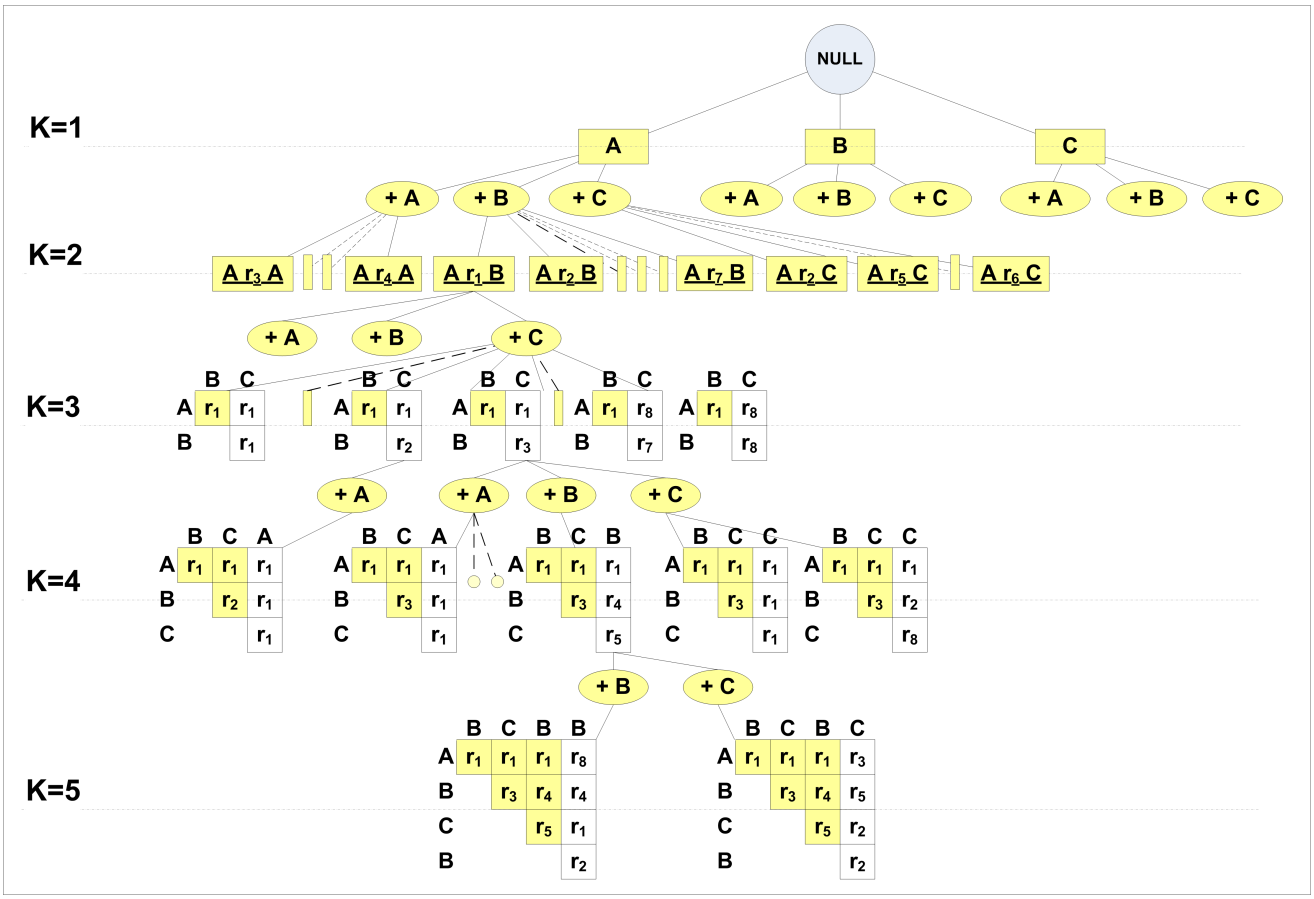

Figure 12: The enumeration tree created by KarmaLego, in which TIRP nodes are expanded directly based on the Incremental Direct Extension concept, where their support is above the minimal vertical support. The ellipse nodes describe the added interval symbol and the half matrices define the corresponding relations generated between the new time interval and the existing intervals shown in the last column.