Similarity Search of Bounded TIDASETS within Large Time Interval Databases

Philipp Meisen
Inst. of Information Management in Mech. Engineering
RWTH Aachen University
Aachen, Germany
philipp.meisen@ima.rwth-aachen.de

Diane Keng
School of Engineering
Santa Clara University
Santa Clara, USA
dkeng@scu.edu

Tobias Meisen
Inst. of Information Management in Mech. Engineering
RWTH Aachen University
Aachen, Germany
tobias.meisen@ima.rwth-aachen.de

Marco Recchioni
Airport Division
Inform GmbH Aachen
Aachen, Germany
marco.recchioni@inform-software.de

Sabina Jeschke
Inst. of Information Management in Mech. Engineering
RWTH Aachen University
Aachen, Germany
sabina.jeschke@ima-zlw-ifu.rwth-aachen.de

***Abstract*—Searching for similar entities within a database is a common and a daily, billions of times, performed task. Generally, similarities are calculated using common distance measures like Manhatten, Euclidian, Levenshtein, Mahalanobis or Dynamic Time Warping (DTW). In this paper, we present a similarity measure for time interval data, which allows searching for similar sets of time interval records bounded by a time window (e.g., a day, a week, or a month). We introduce three different groups of distance measures i.e., temporal order, temporal measure, and temporal relation distances. In addition, we present bitmap-based implementations for algorithms of each of the three types. We designed our solutions to perform well on large datasets and support distributed calculations. Evaluations show the outstanding performance regarding other interval related similarity measures, i.e., ARTEMIS and IBSM.**

Keywords—Similarity Search, Time Interval Data Analysis, Time Series, k-nearest-neighbors

I. INTRODUCTION AND MOTIVATION

The utilization of similarity measures within large data sets is a common task in the field of knowledge discovery. Especially when applying categorization or clustering algorithms, a similarity measure is of great importance to determine the distance between different entities and thereby to categorize entities or to determine a cluster belonging. Over the last years, several similarity measures in various fields and for different types of entities have been proposed, e.g., text, sequences, time series, audio, or ontologies [1–5].

In this paper, we introduce a similarity measure for bounded time interval data sets (bounded TIDASETS). Thereby, a bounded TIDASET is a set of multiple labeled time intervals bounded to a specified time window, e.g. a day, a month, or any range $t_w = [t_1, t_2]$ with $t_1 < t_2$. Fig. 1 exemplifies a bounded TIDASET using a time window based on the time unit day. In addition, the figure illustrates the idea behind the utilization of a similarity measure. Given a query TIDASET of a day, the goal of the similarity search is to determine similar days stored in the underlying time interval database (i.e., 2015-12-07). This enables analyst to get answer to query like "Find the most common days to yesterday regarding the utilization of our resources". Such query, is one example for similarity related queries, that are daily formulated in several industrial areas like production, aviation, logistics, call centers or gastronomy [6, 7]. As we will show, the definition of "common" regarding time interval data is not trivial. First, it can be based on simple counts [7, 8] (e.g., five resources were used in the last hour). Second, it can be based on a measure [8] (e.g., the average income for each minute within the last hour was 52.17 $), or third it can be based on the order of the intervals [9–11] (e.g., the intervals are all following each other). Supporting these kinds of calculations, we introduce three different types of distance measures, namely temporal order, temporal measure, and temporal relation distance (cf. Fig. 1

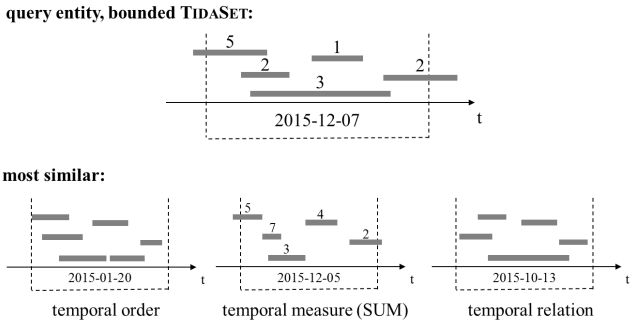


Fig. 1. A daily bounded TIDASET (top) and its most similar daily bounded TIDASETS (bottom), one for each type different distance type.

978-1-4673-9795-7/15 $31.00 © 2015 IEEE
DOI 10.1109/CSCI.2015.36

shows a most similar result for each of the mentioned types). In addition, depending on the specified range, we discuss the matching between different TIDASETS (e.g. comparing January with February). We further present possible solutions for such *matching problems*.

Summarized, the paper is organized as follows: Section II introduces the problem of finding similar bounded TIDASETS within a time interval database, accordingly. In section III we present the related work regarding time interval data and similarity. Furthermore, in section IV, we discuss the *matching problem* and present a solution utilizing mapping functions. In section V different distance measures, a weighted combination, performance improvements utilizing pruning techniques, as well as a distributed calculation for large databases of time interval data are presented. Section VI shows experimental results regarding the performance on different sized databases, different partitions, and weighted combinations. Last, the results presented in this paper are summarized in section VII.

II. PROBLEM STATEMENT

Assuming we have a time axis τ and a time interval database P (cf. [6]). The time axis τ specifies the beginning and end of time regarding the time interval database, i.e., the valid time points of the database as totally ordered set, $\tau := \{t_1, \ldots, t_n\}$. Each record ρ stored in the database (i.e., $\rho \in P$) consists of at least two fields, representing the start and the end, so that each representing a time point on τ so that start $\leq$ end. Thereby, we denote the start field of the record ρ by ρ_{start}, and respectively the end field by ρ_{end}. Additionally, each record may have one or multiple descriptive values, associating additional information to the interval, e.g., a person's name annotating the task performed during the interval, or a list of words said within the time interval. The associated information is assumed valid during the interval defined by $[\rho_{start}, \rho_{end}]$.

We define a TIDASET as a filtered subset of records stored in the database and denote it by Γ (i.e., $\Gamma \subset P$). The term filtered states that there is a logical clause specifying which records are elements of Γ, e.g., person = "Diane". Furthermore, we define a bounded TIDASET $\Gamma_{[s, e]}$, with s and e being time points on τ and $s < e$, as a filtered subset of P so that for all $\rho \in \Gamma_{[s, e]}$: $\rho_{start} \leq e$ and $p_{end} \geq s$. Further, we refer to the interval [s, e] as the time window of a bounded TIDASET.

Example: Fig. 2 exemplifies a database P with eight records. Each record has two descriptive information associated. The first one defines the type of a task (i.e., A or B) performed within the interval, whereby the second defines the amount of resources utilized to perform the task. The illustration uses the time axis

$\tau := \{2015\text{-}12\text{-}01\ 00{:}00, \ldots, 2015\text{-}12\text{-}31\ 23{:}59\}$

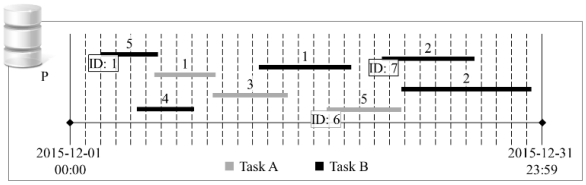


Fig. 2. A sample database P containing eight time interval records, each associated with the amount of resources utilized to perform the task and the type of task.

(i.e., the time axis contains all the minutes of December 2015). Using the depicted database, the following exemplified bounded TIDASETS are valid:

$\Gamma_{[2015\text{-}12\text{-}01\ 00:00,\ 2015\text{-}12\text{-}01\ 23:59]} = \{\}$
$\Gamma_{[2015\text{-}12\text{-}03\ 00:00,\ 2015\text{-}12\text{-}03\ 23:59]} = \{ID: 1\}$
$\Gamma_{[2015\text{-}12\text{-}21\ 00:00,\ 2015\text{-}12\text{-}21\ 23:59]} = \{ID: 6, ID: 7\}$

Based on the definitions of a time interval database P, a bounded TIDASET $\Gamma_{[s, e]}$, and a time axis τ, we define Π_τ as a partition of τ so that $[s, e] \in \Pi_\tau$ and τ is split into subsets keeping the total order, i.e., $\Pi_\tau := \{[t_1, t_x], [t_{x+1}, t_y], \ldots, [t_z, t_n]\}$. Thereby, a logical time unit (e.g., a day, a month, or a week) may define the partition.

Example: Using the time axis shown in Fig. 2 a valid partition of the time axis, using logical time units, could be:

- a daily partition:
  $\Pi_\tau := \{[2015\text{-}12\text{-}01\ 00{:}00, 2015\text{-}12\text{-}01\ 23{:}59],$
  $[2015\text{-}12\text{-}02\ 00{:}00, 2015\text{-}12\text{-}02\ 23{:}59],$
  $\ldots,$
  $[2015\text{-}12\text{-}31\ 00{:}00, 2015\text{-}12\text{-}31\ 23{:}59]\}$
- a hourly partition:
  $\Pi_\tau := \{[2015\text{-}12\text{-}01\ 00{:}00, 2015\text{-}12\text{-}01\ 00{:}59],$
  $[2015\text{-}12\text{-}01\ 01{:}00, 2015\text{-}12\text{-}01\ 01{:}59],$
  $\ldots,$
  $[2015\text{-}12\text{-}31\ 23{:}00, 2015\text{-}12\text{-}31\ 23{:}59]\}$

Given a time interval database P, a bounded TIDASET $\Gamma_{[s, e]}$, a time axis τ, and a partition Π_τ, the *problem of searching for the k most similar bounded TIDASETS* is to find k entities Γ_π with $\pi \in \Pi_\tau \setminus [s, e]$, so that the $dist(\Gamma_{[s, e]}, \Gamma_\pi)$ is minimal.

Thereby, *dist* is the distance function (i.e., a similarity measure). Regarding the utilized distance function, we introduce three different types: temporal order, temporal measure and temporal relation distance. Each of these types cover a different aspect of similarity when dealing with interval data. Fig. 3 illustrates the different types and aspects. The figure shows a similar example for each of type and gives a short explanation:

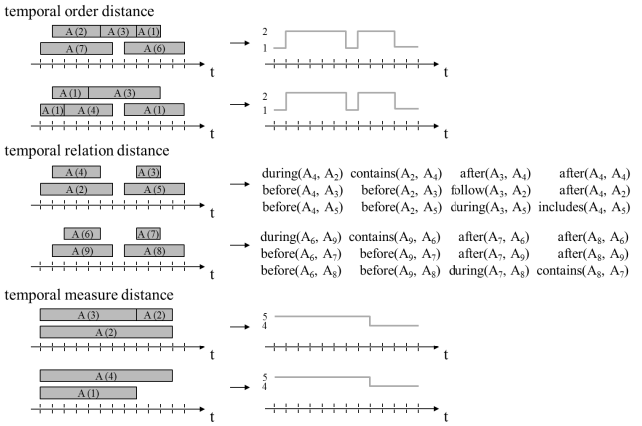


Fig. 3. Illustration of the three different types of distances. For each distance, we present an example showing equality, as well as a visual or textual explanation.

- A temporal order distance (cf. [8, 12]) is based on the amount of intervals being active at a specific time point (i.e., the record's interval includes the time point). In the example, both sets are assumed equal because the count of active intervals is equal in each time point. The line chart on the left illustrates this equality.
- A temporal relation distance (cf. [9, 11]) is based on the relations between the different intervals of a set. The example shows an implementation, which transforms each set into a feature vector (cf. section V). In the example, the sets are assumed equal, because the resulting feature vector are the same.
- A temporal measure distance (cf. [6, 8, 13]) is based on a measure (i.e., a mathematical function) which uses the fields of a record, temporal information, or aggregated dimensional values to determine a value for each time point. The distance aims to determine the difference between the measures resulting from a set of intervals. In the example, a sum measure is utilized and the resulting time series are equal, thus the sets are assumed equal.

III. Related Work

Time interval data is in the focus of research for several decades. Allen [9] specified different relationship between two intervals, which were applied, ambiguously discussed, enhanced, and modified by several researches (cf. [10] for an overview). In addition, different mining algorithms aiming to discover patterns within a (time) interval database were presented in the past [14–19].

Kostakis et al. [11] and Kotsifakos et al. [12] presented three different approaches, regarding the similarity between time interval sets, namely a DTW-based approach [20], ARTEMIS and IBSM. The presented solutions search similar e-sequences within a database. They define an e-sequence (or event-sequence) as a pre-defined set of intervals. The presented algorithms based on the assumption that the e-sequence, e.g., the set of intervals, is pre-defined and known when the sequence is added to the database. Regarding the introduced problem of searching for similar bounded TidaSets, this assumption is not valid, because the user specified the set by a query. Thus, the similarity search has to be flexible regarding the query and result sets. Nevertheless, as we will show, some of the ideas introduced by Kostakis et al. and Kotsifakos et al. can be adapted to support TidaSets.

Recently, systems specialized to analyze time interval data have been published. In [8] we introduced a bitmap-based implementation utilizing different index techniques to increase the performance when accessing and aggregating time interval data. Additionally, we presented an extendable query language to analyze time interval data in [13], which processes fastest on the bitmap-based approach of [8]. Koncilia et al. describe I-OLAP [7], which utilizes a relational database to store time interval data and provides interfaces to translate queries regarding time interval related operations into several relational queries. The system, is based on S-OLAP presented by Chui et al. [21], but does not support any similarity measures regarding sets of time intervals.

Closest to the problem of finding similar sets of bounded TidaSet instances within a time interval data base, is the problem of *full sequence matching* addressed by Kostakis et al. and Kotsifakos et al. As mentioned earlier the introduced solutions for e-sequences are not directly applicable to our problem, but provide interesting approaches. In addition, the presented bitmap-based approaches and processing algorithms are promising regarding performance and bear the capability of distributed calculation. Thus, we implemented the algorithms as extensions for our bitmap-based solution.

IV. The Matching Problem

Prior to presenting similarity measures useful to solve the problem stated in section II, we introduce the *matching problem*. The matching problem addresses the issue arising when comparing the different values associated to the time points of a partition.

Given two bounded TidaSets Γ_{π_1} and Γ_{π_2} with $\pi_1, \pi_2 \in \Pi$ and $\pi_1 \neq \pi_2$. The matching problem states how the different time points of each set π_1 and π_2 are related to each other. In general, different strategies, depending on the domain and the context, may be applied. In this paper, we discuss the following mapping strategies to solve the matching problem:

- matching functions $f: \pi_1 \rightarrow \pi_2 \cup \{\text{null}\}$,
- bilinear interpolation, and
- special techniques like dynamic time warping.

Regarding temporal data, the bilinear interpolation or the usage of special technique like DTW is typically a bad choice. The reasons lie above all in distortion across the temporal boundaries, e.g., the technique should not partly recognize a value measured on a Monday on a Thursday or warp it to another day. In this paper, we focus on matching functions and refer for strategies like bilinear interpolation [12] or DTW [11] to the related publications.

A matching function is a function $\mu: \pi_1 \rightarrow \pi_2 \cup \{\text{null}\}$. The function explicitly allows the matching of a value to null, i.e., mark the value as unmatchable. We thereby assume an unmatchable value with a distance of zero, i.e., it does not affect the distance negatively. Fig. 4 illustrates two examples of matching functions, a weekday match and an ordered match. The weekday match ensures that the first day (e.g., Thursday) of the source set is matched to the same first day of the comparing set. Thus, it may be necessary to skip several days and leave these unmatched. Contrary, the order match associates each day of the source set with the corresponding day regarding the total order of the comparing set.

In general, we assume the definition of a matching function non-strict, i.e., it is possible to match every day of the one set to the same day of the other. Depending on the context, each definable matching function may be meaningful.

V. TidaDistance: Similarity of TidaSets

A. Temporal Order, Measure, and Relation Distance

Based on the definition of a time interval database P, a time axis τ, a time axis partition Π_τ, and a matching function μ, we define three distance measures, one for each type of distance (cf.

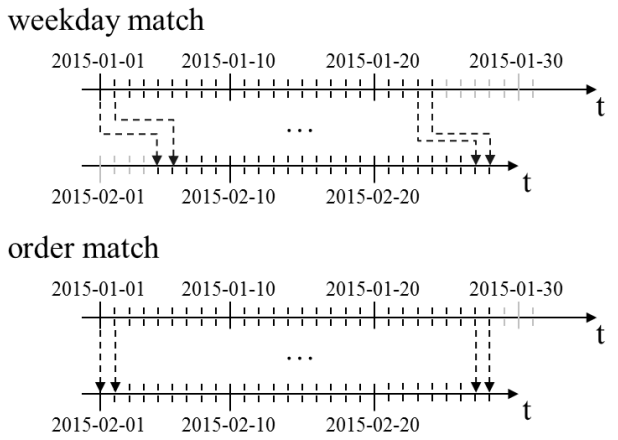


Fig. 4. Two examples of matching functions. The weekday match (top) assigns the first Thursday of the month to the first Thursday of the comparing month, whereby the order match (bottom) just matches the first day of the month to the first day of the comparing month.

Fig. 3). In addition, we define a labeling function $\lambda: P \rightarrow \{l_1, \ldots, l_n\}$, which is used to categorize a record into a group. We denote the set of all labels by $L := \{l_1, \ldots, l_n\}$.

Temporal Order Distance: Let the function

$$\text{cnt}: L \times (\{\Gamma_{\pi_1}\} \times \pi_1, \{\Gamma_{\pi_2}\} \times \pi_2 \cup \{\text{null}\}) \rightarrow \mathbb{N}^0$$

be the function used to count the intervals with a specific label at a specific time point. We define the distance TO_{Dist} between two bounded TIDASETS Γ_{π_1} and Γ_{π_2} as

$$TO_{Dist} := \sum_{l \in L,\, t \in \pi_l} to_{dist}(l, t)$$

with

$$to_{dist} := \left|\text{cnt}\left(l, (\Gamma_{\pi_1}, t)\right) - \text{cnt}\left(l, \left(\Gamma_{\pi_2}, \mu(t)\right)\right)\right|$$

Temporal Measure Distance: Let the function

$$\text{msr}: L \times (\{\Gamma_{\pi_1}\} \times \pi_1, \{\Gamma_{\pi_2}\} \times \pi_2 \cup \{\text{null}\}) \rightarrow \mathbb{R}$$

be the function used to determine the value of a measure of the intervals with a specific label at a specific time point (using a specified aggregation function). We define the distance TM_{Dist} between bounded TIDASETS Γ_{π_1} and Γ_{π_2} as

$$TM_{Dist} := \sum_{l \in L,\, t \in \pi_l} tm_{dist}(l, t)$$

with

$$tm_{dist} := \left|\text{msr}\left(l, (\Gamma_{\pi_1}, t)\right) - \text{msr}\left(l, \left(\Gamma_{\pi_2}, \mu(t)\right)\right)\right|$$

The definition of the temporal measure distance is a generalized version of the temporal count distance. However, using the count function as measure, implicitly adds several temporal aspects (e.g., temporal order) to the distance. On the other hand, the existence of a temporal measure distance allows the comparison of specific, e.g., business-related, measures (e.g., find a day with a similar production flow).

Temporal Relation Distance: The relation distance represents the idea to create a feature vector representing the relations valid at a specific time point (cf. [9]). For this purpose, it is necessary to specify which relations are valid, regarding the set of intervals, at the specific time point. TABLE I lists the time points covered by an relation.

TABLE I. OVERVIEW OF THE COVERED TIME POINTS

relation *rel*	**covered time points** *A := [a1, a2], B := [b1, b2] with A rel B*
overlaps	$[b_1, a_2]$
begins	$[b_1, b_2]$
includes	$[b_1, b_2]$
ends directly before	$[a_2, b_1]$
ends	$[b_1, b_2]$
equal	$[b_1, b_2]$
before	$[a_2 + 1, b_1 - 1]$

Let the function

$$\text{rel}: L \times (\{\Gamma_{\pi_1}\} \times \pi_1, \{\Gamma_{\pi_2}\} \times \pi_2 \cup \{\text{null}\}) \rightarrow \mathbb{N}^0$$

be the function used to count the valid relations of a specific type (i.e., overlaps, begins, includes, ends directly before, or equal) with a specific label at a specific time point. We define the distance TR_{Dist} between bounded TIDASETS Γ_{π_1} and Γ_{π_2} as

$$TR_{Dist} := \sum_{l \in L,\, t \in \tau_S} tr_{dist}(l, t)$$

with tr_{dist} defined as the function summing up the differences between the counts of the different relations

$$tr_{dist} := \sum_{rel} \left|\text{rel}\left(l, (\Gamma_{\pi_1}, t)\right) - \text{rel}\left(l, \left(\Gamma_{\pi_2}, \mu(t)\right)\right)\right|$$

B. Temporal Similarity: Combining the different distances

The three distances temporal order, measure, and relation define the temporal similarity. The impact of each distance is specified by a weighting factor: w_{to}, w_{tr}, and w_{tm}, satisfying $w_{to} + w_{tr} + w_{tm} = 1$. In addition, let max_{to}, max_{tr}, and max_{tm} be the function determining the maximal distance possible for a specific label and time point, i.e.,

$$\max_{to}(l, t) := \max\left(\text{cnt}\left(l, (\Gamma_{\pi_1}, t)\right), \text{cnt}\left(l, \left(\Gamma_{\pi_2}, \mu(t)\right)\right)\right)$$
$$\max_{tm}(l, t) := \max\left(\text{msr}\left(l, (\Gamma_{\pi_1}, t)\right), \text{msr}\left(l, \left(\Gamma_{\pi_2}, \mu(t)\right)\right)\right)$$
$$\max_{tr}(l, t) := \max\left(\sum_{rel} \text{rel}\left(l, (\Gamma_{\pi_1}, t)\right), \sum_{rel} \text{rel}\left(l, \left(\Gamma_{\pi_2}, \mu(t)\right)\right)\right)$$

Based on these maximal distances, we define the similarity as

$$\text{sim} := 1 - \frac{\sum_{l \in L,\, t \in \pi_l} w_{to} \frac{to_{dist}(l, t)}{\max_{to}(l, t)} + w_{tr} \frac{tr_{dist}(l, t)}{\max_{tr}(l, t)} + w_{tm} \frac{tm_{dist}(l, t)}{\max_{tm}(l, t)}}{\text{amount of matched time points} * \text{amount of labels}}$$

C. Bitmap-based calculation

Because of space limitations, we do not introduce the calculation of the count and measure distance in detail. For a more detailed explanation, on how the values for a time point are calculated, the interested reader is referred to [8]. Nevertheless, the calculation of the count and measure distance can be performed in O(n + m) with n being the amount of time points covered by the source set and m being the amount of time points mapped by the mapping function. Additional pruning techniques may lead to an early termination and increase performance (cf. next subsection).

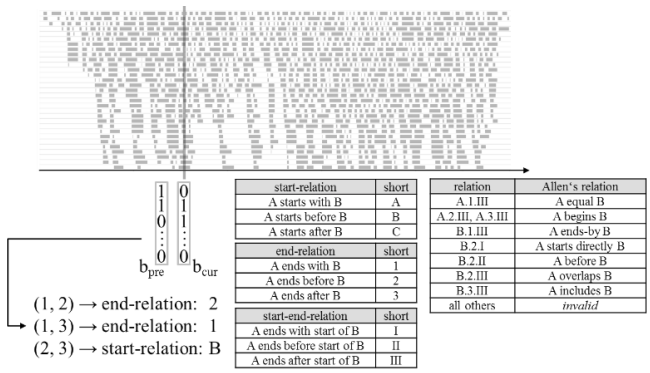
In this paper, we focus on the bitmap-based calculation of the relation distance. The algorithm iterates twice over the time points of the bounded TIDASET. In the first iteration, the algorithm determines the feature vector, which we use in the second iteration to calculate the distance. Thus, a pruning cannot be applied and the algorithm runs in linear time, i.e., O(n + m). To determine the relation between each pair of intervals of a TIDASET and assign it to a time point as specified in TABLE I, the algorithm iterates over each time point, determining the bitmap for the current label of the current time point (using the filter and grouping mechanism presented in [8]). In the next step, the determined bitmap is combined with the bitmap of the previous time point (if no previous calculation was performed, i.e., if it is the first one, the bitmaps are assumed to be empty, i.e., all values are 0). Hereby, we create three bitmaps: (1) a bitmap with all the intervals just started, (2) a bitmap with all the intervals finished, and (3) a bitmap with all the intervals still being active. Each bitmap can be easily determined by logically combining the previous b_{pre} and the current bitmap b_{cur}, i.e.

(1) $(b_{pre} \oplus b_{cur}) \wedge b_{cur}$,

(2) $(b_{pre} \oplus b_{cur}) \wedge b_{pre}$, and

(3) $b_{pre} \wedge b_{cur}$.

Within the last step, the algorithm collects new information for each current pair, i.e., start-relation, end-relation, and start-end-relation. Whenever we determined all three relations of a pair, we are capable of specifying the relation of the pair and the referred time point. Fig. 5 illustrates the process. The figure shows a Gantt-Chart of performed tasks. To determine the relations among the different tasks, the algorithm pics the bitmap of a specific time point (i.e., b_{cur}) and combines it, as mentioned, with the previous bitmap b_{pre}. Based on the resulting three bitmaps, the algorithms determines the three relations and uses the look-up tables (bottom right in the figure) to determine the relation. After determining the relation, is it straightforward to define the vectors for each time point following TABLE I and calculating the distance between two sets following the definition of TR_{Dist}.



start-relation	short
A starts with B	A
A starts before B	B
A starts after B	C

end-relation	short
A ends with B	1
A ends before B	2
A ends after B	3

start-end-relation	short
A ends with start of B	I
A ends before start of B	II
A ends after start of B	III

relation	Allen's relation
A.1.III	A equal B
A.2.III, A.3.III	A begins B
B.1.III	A ends-by B
B.2.I	A starts directly B
B.2.II	A before B
B.2.III	A overlaps B
B.3.III	A includes B
all others	*invalid*

Fig. 5. Illustration of the bitmap-based algorithm utilized to determine the relations among the different intervals.

D. Pruning techniques for k-NN

The calculation of the count and measure distance is performed linear and the distance is increasing with each time point evaluated. Thus, knowing the current smallest distance, i.e., the distance of the k-th nearest neighbor, enables the system to terminate further calculation, if the current distance value becomes larger than the smallest distance. The calculation of the temporal relation distance cannot be pruned, because the algorithm has to determine the relations among the different intervals, which is necessary to calculate the distance in a second iteration. Thus, the second iteration may be terminated, whereby the first iteration has to be performed.

E. Distributed calculation for Large Databases

When performing a k-NN search on large databases (i.e., containing billions of records), it is possible to distribute the distance calculation over several nodes. Each node of a cluster is assigned a specific time window (e.g., a day, a week, or a month). Whenever a distance calculation is triggered, the responsible node maps the bitmaps to the requested distance, which is evaluated within a reduce step. It is even possible to assign several nodes to the same time entity and utilizing a load balancer, which distributes the requested search queries among the different nodes. Last but not least, it should be mentioned, that the database keeping the different bitmaps may also be distributed, e.g., keeping the bitmap of a specific year (cf. [8]).

VI. EXPERIMENTAL RESULTS

We tested our bitmap-based implementation on an Intel Core i7-4810MQ with a CPU clock rate of 2.80 GHz, 32 GB of main memory, an SSD, and running 64-bit Windows 8.1 Pro. As Java implementation, we used a 64-bit JRE 1.6.45, with XMX 1,024 MB and XMS 512 MB. We compared the performance of the temporal order distance to the IBSM algorithm [12]. The performance of the measure and relational distance is not compared, because an implementation of the ARTEMIS algorithm was not available when requested [11]. Within the test, the *ghdatacsv* dataset [22] with 1,122,097 records and a time axis covering one year was used, searching for the 1-NN, 3-NN, 5-NN, and 10-NN. We performed the search against the whole dataset.

Fig. 6 shows the results of the tests. The figure illustrates the runtime for temporal order (i.e., Bitmap vs. IBSM), the temporal measure (i.e., using the measures SUM and a MAX-COUNT [8]), and the temporal relational distance. For the temporal distance, the bitmap-based implementation outperformed IBSM by a factor of four, retrieving the same results as IBSM. The performance of the temporal measure distance is comparable to the once of the temporal count. The calculation of the temporal relation distance is significant slower than the other two, because we have to apply a linear scan prior to determining a value. Thus, we can apply, as already mentioned, the pruning only in the second step of the algorithm.

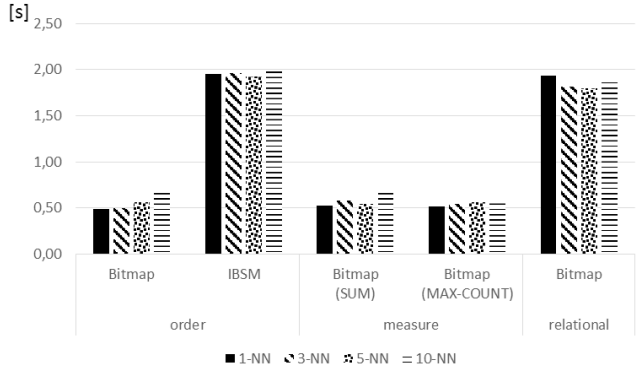


Fig. 6. Results of the performance tests.

VII. Conclusion and Summary

In this paper, we presented a first solution for the *problem of searching for the k most similar bounded TidaSets* introduced in section II. The problem extends the similarity defined for e-sequences (cf. [11, 12]). We introduced three different types of distances useful when searching for similar interval sets within a large time interval database. Based on these distances we defined a weighted similarity measure and outlined bitmap-based implementations. In addition, we shortly introduced the possibilities of performance increases when searching for the k-NN and when utilizing a distributed calculation. The experimental results show that the bitmap-based implementation outperforms the only reference implementation regarding the temporal order distance by a factor of four.

An important task for future studies is to validate the applicability of the algorithms and to enhance the algorithms presented for the introduced distances. In addition, depending on the use case, other distances may be in the focus of research. Furthermore, additional speedup techniques are interesting directions for future work.

Acknowledgment

The approaches presented in this paper are supported by the project "ELLI – Excellent Teaching and Learning in Engineering Sciences" as part of the Excellence Initiative at the RWTH Aachen University.

References

[1] R. Agrawal, C. Faloutsos, and A. N. Swami, "Efficient Similarity Search In Sequence Databases," in *Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms*, London, UK, UK: Springer-Verlag, 1993, pp. 69–84.

[2] J. H. Jensen, M. G. Christensen, D. P. W. Ellis, and S. H. Jensen, "Quantitative Analysis of a Common Audio Similarity Measure," *IEEE Trans. Audio Speech Lang. Process,* vol. 17, no. 4, pp. 693–703, 2009.

[3] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases," *Knowledge and Information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[4] G. Yu, F. Li, Y. Qin, X. Bo, Y. Wu, and S. Wang, "GOSemSim: an R package for measuring semantic similarity among GO terms and gene products," (eng), *Bioinformatics (Oxford, England)*, vol. 26, no. 7, pp. 976–978, 2010.

[5] D. Metzler, S. Dumais, and C. Meek, "Similarity Measures for Short Segments of Text," in *Lecture Notes in Computer Science, Advances in Information Retrieval*, G. Amati, C. Carpineto, and G. Romano, Eds, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 16–27.

[6] P. Meisen, M. Recchioni, T. Meisen, D. Schilberg, and S. Jeschke, "Modeling and processing of time interval data for data-driven decision support," in *2014 IEEE International Conference on Systems, Man and Cybernetics - SMC*, San Diego, California, USA, 2014, pp. 2946–2953.

[7] C. Koncilia, T. Morzy, R. Wrembel, and J. Eder, "Interval OLAP: Analyzing Interval Data," in *Lecture Notes in Computer Science, Data Warehousing and Knowledge Discovery*, L. Bellatreche and M. K. Mohania, Eds, Cham: Springer International Publishing, 2014, pp. 233–244.

[8] P. Meisen, D. Keng, T. Meisen, M. Recchioni, and S. Jeschke, "Bitmap-Based On-Line Analytical Processing of Time Interval Data," in *Proceedings of the 12th International Conference on Information Technology: New Generations (ITNG), 2015*, 2015.

[9] J. F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, http://doi.acm.org/10.1145/182.358434, 1983.

[10] F. Moerchen, "Tutorial CIDM-T Temporal pattern mining in symbolic time point and time interval data," in *Computational Intelligence and Data Mining, 2009. CIDM '09. IEEE Symposium on*, 2009, p. xiv.

[11] O. Kostakis, P. Papapetrou, and J. Hollmén, "ARTEMIS: Assessing the Similarity of Event-Interval Sequences," in *Lecture Notes in Computer Science, Machine Learning and Knowledge Discovery in Databases*, D. Gunopulos, T. Hofmann, D. Malerba, and M. Vazirgiannis, Eds.: Springer Berlin Heidelberg, 2011, pp. 229–244.

[12] A. Kotsifakos, P. Papapetrou, and V. Athitsos, "IBSM: Interval-Based Sequence Matching: 65," in *Proceedings of the 2013 SIAM International Conference on Data Mining*, 2013, pp. 596–604.

[13] P. Meisen, D. Keng, T. Meisen, and M. Recchioni, "TIDAQL: A Query Language enabling On-line Analytical Processing of Time Interval Data," in *Proceedings of 17th International Conference on Enterprise Information Systems (ICEIS2015)*, 2015.

[14] Y.-H. Hu, C. Cheng, F. Wu, and C.-I. Yang, "Mining Multi-level Time-interval Sequential Patterns in Sequence Databases," in *2nd International Conference on Software Engineering and Data Mining (SEDM 2010): Chengdu, China, 23 - 25 June 2010*, Piscataway, NJ: IEEE, 2010, pp. 416–421.

[15] R. Sadasivam and K. Duraiswamy, "Efficient approach to discover interval-based sequential patterns," vol. 9, no. 2, pp. 225–234, http://thescipub.com/abstract/10.3844/jcssp.2013.225.234, 2013.

[16] Y.-L. Chen, M.-C. Chiang, and M.-T. Ko, "Discovering time-interval sequential patterns in sequence databases," *Expert Systems with Applications*, vol. 25, no. 3, pp. 343–354, 2003.

[17] Y.-C. Chen, W.-C. Peng, and S.-Y. Lee, "CEMiner - An Efficient Algorithm for Mining Closed Patterns from Time Interval-Based Data," in *IEEE 11th International Conference on Data Mining (ICDM 2011)*, 2011, pp. 121–130.

[18] P. Papapetrou, G. Kollios, and S. Sclaroff, "Discovering frequent arrangements of temporal intervals," in *Proceedings of the 5th IEEE International Conference on Data Mining (ICDM'05)*: IEEE Press, 2005, pp. 354–361.

[19] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos, "Mining frequent arrangements of temporal intervals," (English), *Knowledge and Information Systems*, vol. 21, no. 2, pp. 133–171, http://dx.doi.org/10.1007/s10115-009-0196-0, 2009.

[20] E. Keogh and C. A. Ratanamahatana, "Exact Indexing of Dynamic Time Warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[21] C. K. Chui, B. Kao, E. Lo, and D. Cheung, "S-OLAP: An OLAP system for analyzing sequence data," in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, 2010, pp. 1131–1134.

[22] P. Meisen and M. Recchioni, *Time Interval Data (ghdatacsv)*. DOI: 10.13140/RG.2.1.4597.1687