Complete Closed Time Intervals-Related Patterns Mining

Omer David Harel, Robert Moskovitch
Software and Information Systems Engineering, Ben Gurion University of the Negev, Be'er Sheva, Israel
omerdavi@post.bgu.ac.il, robertmo@bgu.ac.il

Abstract

Using temporal abstraction, various forms of sampled multivariate temporal data can be transformed into a uniform representation of *symbolic time intervals*, from which *Time Intervals Related Patterns (TIRPs)* can be then discovered. Hence, mining TIRPs from symbolic time intervals offers a comprehensive framework for heterogeneous multivariate temporal data analysis. While the field of time intervals mining has gained a growing interest in recent decades, frequent *closed TIRPs* mining was not investigated in its full complexity. Mining frequent closed TIRPs is highly effective due to the discovery of a compact set of frequent TIRPs, which contains the complete information of all the frequent TIRPs. However, as we demonstrate in this paper, the recent advancements made in closed TIRPs discovery are incomplete, due to the discovery of only the first instances of the TIRPs within each STIs series in the database. In this paper we introduce the *TIRPClo algorithm – for complete and efficient mining of frequent closed TIRPs*. The algorithm utilizes a memory-efficient index and a novel method for data projection, due to which it is the *first* algorithm to guarantee a complete discovery of frequent closed TIRPs. In addition, a rigorous runtime comparison of TIRPClo to state-of-the-art methods is performed, demonstrating a significant speed-up on various real-world datasets.

Introduction

In the recent two decades, there has been a growing interest in discovering temporal knowledge through frequent temporal patterns, and particularly in the discovery of frequent *Time Intervals-Related Patterns (TIRPs)* from *symbolic time intervals (STIs)*. STIs may be raw, i.e., describing events having duration, such as a time period a patient is prescribed on a medication or the period of time the green light is on in a traffic light.

Alternatively, STIs can be created from time-points series after employing *temporal abstraction* methods (Shahar 1997; Lavrac, Keravnou, and Zupan 2000; Höppner 2002; Lin et al. 2003; Mörchen and Ultsch 2005; Azulay et al. 2007; Moskovitch and Shahar 2015a). Through a temporal abstraction process various forms of temporal variables, whether sampled regularly or irregularly, are transformed into a uniform representation of STIs. The STIs mostly represent periods of time that a variable is in a specific state (defined by cutoffs), or segments of an increasing or a decreasing period of the values.

Incorporating various types of temporal variables into a uniform STIs series representation, enables the discovery of frequent TIRPs from multivariate heterogeneous temporal data. The discovered frequent TIRPs have a meaningful potential usage in real-world applications, being directly used for knowledge discovery (Sacchi et al. 2007), classification (Patel, Hsu, and Lee 2008; Batal et al. 2009) or the prediction of certain outcomes (Moskovitch et al. 2015). In dynamic graphs, for example, frequent time intervals series mining can be also used for the discovery of frequent temporal patterns of edge-interactions (Kostakis and Gionis 2017). In addition, as will be explained in the Background section, TIRPs are explicitly represented by the temporal relations among their STIs. Therefore, they can be easily interpreted by domain experts, which makes time intervals mining a very attractive technique for temporal data analytics in real-life data.

The discovery of frequent *closed TIRPs*, on which we will elaborate in the Background section, in particular, has a significant potential benefit over discovering the entire set of frequent TIRPs. While a single frequent closed TIRP may contain an exponential number (in the size of the TIRP) of frequent sub-TIRPs that are not closed, their entire information is contained within the closed TIRP itself. Thus, the discovery of frequent closed TIRPs potentially produces a much more compact output of frequent TIRPs, which contains the complete information of all the frequent TIRPs.

However, the definition of the complete discovery of frequent closed TIRPs, a problem that was recently investigated for the first time in (Chen, Weng, and Hui 2016), was not addressed properly. Earlier methods have discovered only the first instance of a TIRP within each STIs series in the database. The following instances of the TIRPs, on the other hand, have been ignored, which eventually results in an incomplete discovery of frequent closed TIRPs. In this paper we demonstrate this completeness problem, both qualitatively and quantitatively, and introduce a novel algorithm which is complete. The algorithm also keeps track of *the complete set of instances* of the frequent closed TIRPs that it discovers.

The main contributions of the paper are the following:

1. *TIRPClo* – An efficient algorithm for complete mining of frequent closed TIRPs, including:
   (a) A novel method for STIs series transformation into sequential end-points representation, avoiding ambiguity.
   (b) A complete method for database projection, which overcomes the main challenge of current projection mechanisms, i.e., the detection of the reoccurring instances of the TIRPs within a single STIs series.
2. A comprehensive qualitative and quantitative analysis of the completeness problem in the current closed TIRPs discovery methods.
3. A rigorous runtime comparison of TIRPClo with recent state-of-the-art methods on various real-world datasets, demonstrating a significant speed-up.
4. The code of the TIRPClo algorithm as well as the evaluation datasets, which are all publicly available[1].

Background

Symbolic Time Intervals Mining

Temporal data include not only time-stamped raw data or time-points series, but also time intervals, which are events having a type and a non-zero duration. Such time intervals are referred to as *symbolic time intervals* (STIs).

Definition 1. (STI) A *symbolic time interval* $I = (symbol, s, f)$ is a triplet of a symbol, a start-time and a finish-time.

Definition 2. (Lexicographical STIs series) A lexicographical STIs series $(I_1, I_2, ..., I_k)$ is a sorted series of STIs, s.t. $\forall I_i, I_j$: $i < j \equiv I_i.s < I_j.s \vee (I_i.s = I_j.s \wedge I_i.f < I_j.f) \vee (I_i.s = I_j.s \wedge I_i.f = I_j.f \wedge I_i.symbol < I_j.symbol)$.

Frequent TIRPs are mined from a database of STIs series, in which each series is associated to an entity (e.g., patient) that has a unique identifier, i.e., the entity ID. The most common method to define the temporal relations among the STIs is using Allen's temporal relations (Allen 1983). Allen has formulated a finite set of 13 temporal relations between a pair of STIs. The set includes seven basic relations – six of which having an inverse relation, while *equal* is its own inverse, as shown in Figure 1. When the STIs are lexicographically ordered (definition 2), it is sufficient to use the seven relations, without their inverse relations.

Definition 3. (TIRP) A *Time Intervals-Related Pattern* (*TIRP*) is defined as $T = \{T_{Intervals}, T_{Relations}\}$ where $T_{Intervals} = (I_1, I_2, ..., I_k)$ is a lexicographically ordered set of k STIs and

$$T_{Relations} = \bigwedge_{i=1}^{k} \bigwedge_{j=i+1}^{k} AllenRel(I_i, I_j)$$

defining the conjunction of Allen's temporal relations among each of the $\binom{k}{2}$ pairs of STIs in $T_{Intervals}$.

Note that the timestamps of the STIs' end-points are not part of the TIRP definition, but only their symbols and the temporal relations among them.

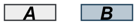
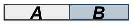
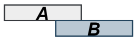
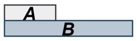
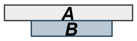
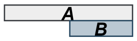
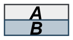
Allen Relation	*Example*	*Sequence Definition*	*Inverse Relation*
A **before** *B*		*A.s < A.f < B.s < B.f*	*B* **after** *A*
A **meets** *B*		*A.s < A.f = B.s < B.f*	*B* **met-by** *A*
A **overlaps** *B*		*A.s < B.s < A.f < B.f*	*B* **overlapped-by** *A*
A **starts** *B*		*A.s = B.s < A.f < B.f*	*B* **starts-by** *A*
A **contains** *B*		*A.s < B.s < B.f < A.f*	*B* **during** *A*
A **finished-by** *B*		*A.s < B.s < A.f = B.f*	*B* **finishes** *A*
A **equals** *B*		*A.s = B.s < A.f = B.f*	*B* **equals** *A*

Figure 1: Allen's 13 temporal relations between a pair of STIs.

Definition 4. (Vertical support) The *vertical support* of a TIRP T is the distinct number of entities $|E_T|$ in which T appears at least once, divided by the total number of entities in the database $|E|$. Therefore, $vertical\,support(T) = \frac{|E_T|}{|E|}$.

Definition 5. (Horizontal support) The *horizontal support* of a TIRP T within an entity e is the number of instances of T within e's STIs series.

Example. In Figure 2, the TIRP $T_1 =< A\,overlaps\,B >$ consists of the two STIs A and B, among which the temporal relation is $overlap$. T_1 appears twice within (a) and once within (b). Therefore, in the dataset shown in Figure 2, $vertical\ support(T_1) = \frac{|E_{T_1}|}{|E|} = 1$. In addition, $horizontal\ support(T_1, (a)) = 2$, while $horizontal\ support(T_1, (b)) = 1$.

While the methods that we review in this subsection focus on the discovery of frequent TIRPs from STIs, not all of them maintain the time intervals-based representation. Looking at the methods that were published, it is clear that two types of approaches have been developed. Some methods are time intervals-based (Winarko and Roddick 2007; Patel, Hsu, and Lee 2008; Papapetrou et al. 2009; Moskovitch and Shahar 2015b; Sharma and Patel 2018), that directly discover frequent TIRPs composed of STIs and the conjunction of Allen's relations among them (definition 3).

Others are sequence-based (Wu and Chen 2007; Chen, Weng, and Hui 2016), which transform the STIs series data into a sequential representation of the STIs' start and finish end-points. Then, typically a sequential mining-based method is applied to the database of end-points sequences in order to discover the frequent TIRPs, which are represented as frequent sequences of the end-points of the TIRPs' STIs. Note that a discovered frequent end-points sequence does not correspond to a valid TIRP if it contains only one of the two end-points (either start or finish) of some STI. In addition, Allen's temporal relations among the STIs are implicitly represented by the sequential order of their end-points. For that, a non-ambiguous sequential representation is needed. Ambiguity means either 1) having different representations for the same temporal relation, or 2) having one representation which expresses different temporal relations.

[1] https://github.com/omerh18/TIRPClo

Closed TIRPs Mining

Definition 6. (Super-pattern) In sequential events representation, given two sequential patterns $p_1 =< t_1, t_2, ..., t_k >$ and $p_2 =< t_1^{'}, t_2^{'}, ..., t_n^{'} >$, p_2 is a *super-pattern* of p_1 if and only if there exist indices $1 \leq i_1 < i_2 < ... < i_k \leq n$ s.t. $t_1 = t_{i1}^{'}, t_2 = t_{i2}^{'}, ..., t_k = t_{ik}^{'}$.

Definition 7. (Super-TIRP) Let T_1 and T_2 be two TIRPs. Then T_2 is a *super-TIRP* of T_1 if and only if 1) $T_{1Intervals} \subseteq T_{2Intervals}$ and 2) $\forall I_i, I_j \in T_{1Intervals}$: $T_{1Relations}(I_i, I_j) = T_{2Relations}(I_i, I_j)$.

Definition 8. (Closed TIRP) A TIRP T_1 is a closed TIRP if and only if it has no super-TIRP T_2 s.t. $vertical\ support(T_1)$ = $vertical\ support(T_2)$.

Example. In Figure 2, the TIRP $T_2 =< A\ overlaps\ B \wedge B\ overlaps\ C \wedge A\ before\ C >$ is a super-TIRP of the TIRP $T_1 =< A\ overlaps\ B >$. That is since 1) $T_{1Intervals} = (A, B) \subseteq (A, B, C) = T_{2Intervals}$; and 2) $T_{1Relations}(A, B) = T_{2Relations}(A, B) = overlap$. In addition, T_1 and T_2 have the same vertical support. Therefore, T_1 is not a closed TIRP, while its super-TIRP T_2 is indeed closed.

The discovery of frequent closed temporal patterns has recently gained a significant interest. That is due to potentially producing a much more compact output of frequent patterns, which contains the entire information of *all* the frequent patterns. However, prior research on closed temporal patterns mining has mainly focused on sequential data (Yan, Han, and Afshar 2003; Wang and Han 2004; Tzvetkov, Yan, and Han 2005; Huang et al. 2006; Chang et al. 2008). The CCMiner algorithm (Chen, Weng, and Hui 2016) is the first and only method proposed so far for the discovery of frequent closed TIRPs from STIs-based data.

In the next subsection we demonstrate the completeness problem in the current approach for closed TIRPs discovery, which will be also empirically investigated in the experimental section.

Completeness

Definition 9. Complete mining of frequent closed TIRPs: Given a dataset of $|E|$ entities' STIs series, the goal of the *complete frequent closed TIRPs mining task* is to discover all the frequent closed TIRPs (definition 8), given a minimum vertical support threshold.

Previous methods for closed TIRPs discovery (Chen, Weng, and Hui 2016) have intended to discover only the first TIRP instance within each entity's STIs series, and not the entire horizontally supporting instances of the TIRPs (definition 5). That is probably due to meaningful complexity and computational requirements. However, in this subsection we show that in order to correctly count the TIRPs' vertical support values, their horizontal support discovery is essential. Otherwise, TIRPs' vertical support values are potentially undercounted. Consequentially, TIRPs that are indeed frequent (i.e., their correct vertical support value is above the minimum support threshold) are wrongly considered infrequent and are thus not discovered, which results in an incomplete discovery of frequent closed TIRPs.

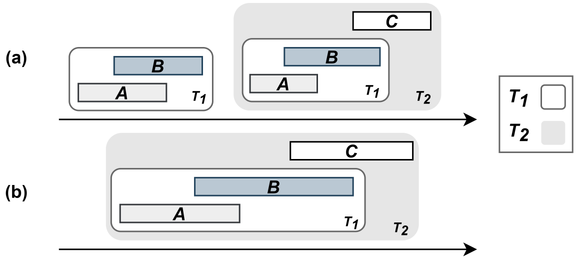


Figure 2: Two STIs series that contain the TIRPs T_1 and T_2. In series (a), the discovery of only the first instance of T_1 results in not discovering the single instance of T_2, whose vertical support is undercounted.

This is illustrated in Figure 2, which shows two STIs series (a) and (b), and two TIRPs - T_1 and T_2. The TIRP $T_1 = < A\ overlaps\ B >$ appears twice within (a) and once within (b), while the TIRP $T_2 = < A\ overlaps\ B \wedge B\ overlaps\ C \wedge A\ before\ C >$ appears only once within each of the series. In series (a), T_1's second instance appears as part of the TIRP T_2, while its first instance does not. Assume that only the first instance of a TIRP within each supporting entity is discovered, as made in (Chen, Weng, and Hui 2016). Then, in series (a), only the first instance of T_1 is discovered while its second instance is ignored, which is sufficient for T_1's vertical support counting.

However, due to ignoring the second instance of T_1 in (a), the only instance of T_2 in this series is not discovered as well. That is since typically, in order to discover an instance of a TIRP, its prefix has to be discovered first, and then extended (e.g., an instance of $< A >$ should be discovered and extended in order to discover an instance of $< A\ overlaps\ B >$). As a result, T_2's vertical support is undercounted and it might be wrongly considered infrequent. In such case, the closed TIRP T_2 is not discovered despite it is indeed frequent, which results in an incomplete discovery of frequent closed TIRPs. Therefore, the entire set of horizontally supporting instances of the TIRPs *must be discovered* in order to guarantee a complete discovery of frequent closed TIRPs, as defined in definition 9.

Methods

In this section we introduce *TIRPClo* – a complete and efficient sequence-based algorithm for frequent closed TIRPs discovery.

Definition 10. (Tiep) A *time-interval-end-point (tiep)* represents a STI's end-point through a pair of a symbol and an *end-type*, which can be either *start* or *finish*.

In this paper, the start-tiep and finish-tiep of a STI which has the symbol A are denoted by A^+ and A^- respectively. Note that in real-life data multiple STIs having the same symbol can occur within a single entity's STIs series. Thus, an index i is used in order to differentiate their tieps' instances that appear within the same entity. For example, the i^{th} instance of a STI A is represented by A_i^+ for its start-tiep and A_i^- for its finish-tiep.

Definition 11. (Complementing-tiep) Given a start-tiep $t = A^+$, its *complementing tiep* is A^- and vice versa. We denote a tiep t's complementing tiep by t^C.

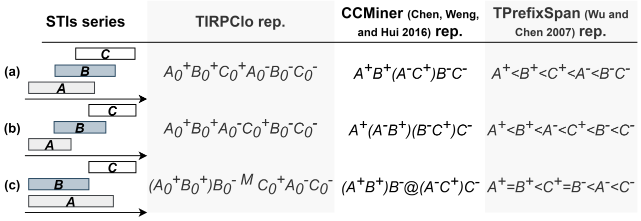
```
Algorithm 1: TIRPClo
   input : db - lexicographically ordered STIs series database
   output: $CT$ - complete set of frequent closed TIRPs
 1 $sdb \leftarrow STIs2Seq(db)$
 2 $CT \leftarrow \emptyset$
 3 for $t \in TiepsIndex$ do
 4     if $|TiepsIndex[t].E_t| < minSup$ then
 5         $TiepsIndex.remove(t)$
 6 for $t \in TiepsIndex$ do
 7     if $t.endType == START$ then
 8         $sdb_t \leftarrow ESProject(sdb, t, E_t)$
 9         if $\neg CheckPrune(t)$ then
10             $ExtendTIRP(t, sdb_t, CT, NULL)$
11 return $CT$
```

The main steps of the TIRPClo algorithm are described in Algorithm 1. As a sequence-based algorithm, TIRPClo starts with transforming the STIs series database into a novel tieps-based sequences database representation, on which we will elaborate in the next subsection (line 1). The transformation method – *STIs2Seq*, also indexes the tieps in the *tieps-index*, which enables the retrieval of the ordered instances of the tieps within their supporting entities in $O(1)$.

Then, the infrequent tieps are filtered out of the index, according to the Apriori-All principle (lines 3-5). For each frequent start-tiep t, the *Entities Spawning* projection method (Algorithm 3) is applied in order to project the initial sequences database by t (lines 6-8). In line 9 it is verified whether or not the current one-tiep pattern should be pruned, according to TIRPClo's closure-checking scheme. In TIRPClo, a pattern is pruned only in case any future extension of it will for sure not form a frequent closed TIRP. Otherwise, the *ExtendTIRP* method (Algorithm 2) is applied in order to discover all the frequent closed TIRPs that begin with t (line 10). Finally, the complete set of frequent closed TIRPs is discovered and returned by the algorithm (line 11).

```
Algorithm 2: ExtendTIRP
   input : $p$ - current pattern
           $sdb_p$ - projected sequences database of $p$
           $CT$ - set of discovered frequent closed TIRPs
           $prevSIs$ - previous support-indices
 1 if $isValidTIRP(p) \wedge isClosed(p)$ then
 2     $CT \leftarrow CT \cup \{p\}$
 3 $cndSIs \leftarrow GetSupportIndices(p, sdb_p, prevSIs)$
 4 for $< cnd_t, cnd_tSI > \in cndSIs$ do
 5     if $cnd_tSI.verticalSupport \geq minSup$ then
 6         $p^{'} = p + cnd_t$
 7         $sdb_{p^{'}} \leftarrow ESProject(sdb_p, cnd_t, cnd_tSI)$
 8         if $\neg CheckPrune(p^{'})$ then
 9             $ExtendTIRP(p^{'}, sdb_{p^{'}}, CT, cndSIs)$
```

The *ExtendTIRP* method (Algorithm 2) extends a current frequent pattern p recursively. Note that as explained in the Background section, for p to be a valid TIRP, all of its tieps must appear paired with their complementing tieps. Thus, p is added to the set of frequent closed TIRPs that are discovered by the algorithm only if it is both valid and closed, according to TIRPClo's closure-checking scheme, which is verified in lines 1-2. Then, in order to generate the candidate tieps for the extension of p, the tieps' *support-indices* are generated (line 3). A support-index is a data structure that is generated for each tiep cnd_t which is a valid candidate for the extension of p (see Candidates Generation subsection). The support-index holds cnd_t's vertical support value and points at its first instance within each supporting entity's record in the projected sequences database sdb_p.



STIs series	TIRPClo rep.	CCMiner (Chen, Weng, and Hui 2016) rep.	TPrefixSpan (Wu and Chen 2007) rep.
(a)	$A_0{}^+B_0{}^+C_0{}^+A_0{}^-B_0{}^-C_0{}^-$	$A^+B^+(A^-C^+)B^-C^-$	$A^+{<}B^+{<}C^+{<}A^-{<}B^-C^-$
(b)	$A_0{}^+B_0{}^+A_0{}^-C_0{}^+B_0{}^-C_0{}^-$	$A^+(A^-B^+)(B^-C^+)C^-$	$A^+{<}B^+{<}A^-{<}C^+{<}B^-{<}C^-$
(c)	$(A_0{}^+B_0{}^+)B_0{}^{-\ M}\ C_0{}^+A_0{}^-C_0{}^-$	$(A^+B^+)B^-@(A^-C^+)C^-$	$A^+{=}B^+{<}C^+{=}B^-{<}A^-{<}C^-$

Figure 3: Three STIs series and their sequential representations in TIRPClo, (Chen, Weng, and Hui 2016) and (Wu and Chen 2007).

For each valid candidate tiep cnd_t which is currently frequent, it is selected for the extension of p (lines 4-6). For that, the current sequences database sdb_p is first re-projected by cnd_t (line 7). Then, the *ExtendTIRP* method is repeatedly applied in order to further extend the pattern $p^{'} = p + cnd_t$, in case it should not be pruned (lines 8-9). Note that the vertical support of $p^{'}$ is known in advance and equals to cnd_t's current vertical support value, which is above the minimum support threshold by selection. Hence, all the extended patterns in TIRPClo are guaranteed to be frequent, and TIRPClo performs the least pattern extensions required for the discovery of the complete set of frequent closed TIRPs based on a single scan of the data.

TIRPClo Novel Sequence-based TIRP Representation and Tieps-Index

TIRPClo introduces a novel sequence-based representation of STIs series, which is both non-ambiguous and also clearer compared to current relevant representations (Chen, Weng, and Hui 2016; Wu and Chen 2007). Figure 3 shows TIRPClo's TIRPs' representation versus the existing representations. In TIRPClo, all of Allen's temporal relations (Figure 1) are uniquely represented due to the following properties.

First, TIRPClo always keeps the chronological order of the tieps, unlike in (Chen, Weng, and Hui 2016), in which tieps having close timestamps are considered coinciding and marked by brackets, which may change their order sometimes. In addition, TIRPClo attaches together coinciding tieps (having exactly the same timestamp) which have the same *end-type* (i.e., start or finish), ordered by an alphabetical order of their symbol, unlike in (Wu and Chen 2007). For example – the tieps $A_0{}^+$ and $B_0{}^+$ in Figure 3c.

However, in TIRPClo's representation, two tieps that do not have the same end-type are never put together in brackets, unlike in (Chen, Weng, and Hui 2016). When tieps have the same timestamp but their end-type is opposed, as happens with ${B_0}^-$ and ${C_0}^+$ in series (c), their STIs are *met* (Figure 1). Thus, the tieps are separated so that the finish-tieps are followed by the start-tieps. In addition, a M character is inserted among them to mark that they are met (Figure 3c). This character is used to differentiate the *before* and *meet* temporal relations.

During the transformation process of the STIs series data into the described tieps-based sequential representation, the tieps are also indexed in the *tieps-index*. The tieps-index maps each tiep's representation (e.g.., A^+) to a *master-tiep* using a hash map. The master-tiep indexes all of the tiep's specific instances by entity-id, ordered by their timestamp. Hence, a tiep t's i^{th} instance within an entity e, can be retrieved in a constant time through $TiepsIndex[t][e][i]$. In addition, since each STI is composed of exactly two tieps, given a database of N input STIs the index size is exactly $2N$, which makes it efficient memory-wise.

After the transformation method is applied to all the entities' STIs series in the database, the initial sequences database is created. Along the projection-based mining process, the sequences database shrinks typically, or at least does not change, corresponding to the current projected pattern. In TIRPClo, the output is designed to include the complete set of instances of the discovered TIRPs. For that, a record in a projected sequences database is a pair $< tcseq, pi >$, in which $tcseq$ is a tieps sequence of an entity from which the pattern instance pi has been projected.

The Entities Spawning Method for Database Projection

Sequences database projection methods typically search for the first instance of a tiep within an entity's record, without intending to discover the horizontal support. A projected record includes the elements appearing after the tiep's first instance. An example is shown in Figure 4(a), in which projection by the tiep B^+ refers only to its first instance, while there are another two instances which are ignored. Thus, using current projection methods for sequence-based TIRPs mining eventually results in an incomplete discovery of frequent TIRPs, as explained in the Background.

The *Entities Spawning* projection method that we introduce here addresses this very challenge of horizontal support discovery, in order to guarantee a complete discovery of frequent closed TIRPs. For that, projection of an entity's record by a tiep that appears N times within it, requires N projections – which result in multiple projected records. Figure 4(b) illustrates the multiple projections performed in TIRPClo, which refer to each of the three instances of the tiep B^+ within the original record, and result in three projected records. Furthermore, when having N instances of a tiep t within an entity's record, TIRPClo's projection of the record by the i^{th} instance of t within it, keeps the next $N - i$ instances of t within the projected record. Thus, TIRPs that contain multiple STIs which have the same symbol are discovered in TIRPClo as well.

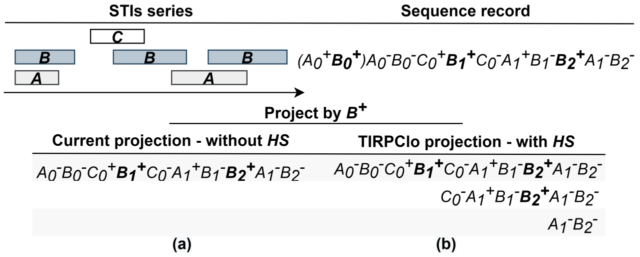

STIs series
Sequence record
C
B
B
B
A
A
(A0+B0+)A0-B0-C0+B1+C0-A1+B1-B2+A1-B2-
Project by B+
Current projection - without HS
TIRPClo projection - with HS
A0-B0-C0+B1+C0-A1+B1-B2+A1-B2-
A0-B0-C0+B1+C0-A1+B1-B2+A1-B2-
C0-A1+B1-B2+A1-B2-
A1-B2-
(a)
(b)

Figure 4: Projection without discovering the horizontal support, as currently made (a), versus TIRPClo's projection that includes horizontal support discovery (b).

In order to limit the discovery of potentially meaningless frequent TIRPs, which have very large time durations between their STIs, TIRPClo uses a *maximal gap* time constraint by which the projection is limited. The maximal gap was first introduced in (Papapetrou et al. 2009) and is defined as follows.

Definition 12. (Maximal gap) The *maximal gap* is the maximal time duration allowed between two STIs among which the temporal relation is *before*, for TIRPs discovery.

Algorithm 3: *ESProject*

```
   input : $sdb$ - current sequences database
           $t$ - tiep to project from $sdb$
           $tSI$ - $t$'s support-index
   output: $sdb_t$ - projected sequences database
1  $sdb_t \leftarrow \emptyset$
2  for $< record, firstInstance > \in tSI.instances$ do
3      $e \leftarrow record.entityID$
4      $eInstances \leftarrow TiepsIndex[t][e]$
5      for $tInstance \in eInstances[firstInstance :]$ do
6          if $tInstance.endType == START \wedge$
             $\neg maxGapHolds(record.pi, tInstance)$ then
7              break
8          $tCo \leftarrow tInstance.coincidence$
9          $projectedRecord \leftarrow$ project $tCo$ by $tInstance$
10         if $projectedRecord \neq NULL$ then
11             $pi' \leftarrow$ extend $record.pi$ with $tInstance$
12             $sdb_t.add(projectedRecord, pi')$
13 return $sdb_t$
```

The Entities Spawning projection method is described in Algorithm 3, which receives three parameters as input: a sequences database sdb for further projection, a tiep t based on which the database is projected; and t's support-index. First, using the support-index, t's instances within each record are traversed from the first instance for projection (lines 2-5). The projection is conditioned by the maximal gap, making sure that the duration till the next STI is below it (lines 6-7). Then, projection is applied to the specific coincidence in the record's sequence which contains t's current instance (lines 8-9). The projected record and the extended pattern instance are then added to the projected sequences database sdb_t (lines 10-12), which is eventually returned.

Candidates Generation

TIRPClo enables the discovery of frequent TIRPs that contain multiple STIs which have the same symbol. Hence, a tiep t may appear multiple times within a discovered pattern p. In order to represent the number of instances of t within p, the notation $\#_p t$ is introduced. Accordingly, the conditions for a tiep cnd_t which is a valid candidate in TIRPClo for the extension of p are the following:

1. If cnd_t is a start-tiep, it is always a valid candidate for p's extension.
2. If cnd_t is a finish-tiep, then it is a valid candidate if $\#_p cnd_t < \#_p {cnd_t}^C$, which means that there are more instances of cnd_t's complementing start-tiep within p, than instances of cnd_t within p.

Example. The valid candidate tieps for the extension of $p = < A^+ >$ are all the start-tieps (condition 1) and the single finish-tiep A^- (condition 2). The valid candidates for the extension of $q = < A^+B^+ >$ however, include not only the valid candidates of p, but also the finish-tiep B^-. That is since $0 = \#_q B^- < \#_q B^+ = 1$.

Closure-Checking

In order to discover the complete set of frequent closed TIRPs, TIRPClo consists of a closure-checking scheme, which is used for the detection and possibly pruning of the unclosed TIRPs early during the mining process. Since TIRPClo is a sequence-based TIRPs mining algorithm, its closure-checking scheme is inspired by the bi-directional extension approach (Wang and Han 2004), that has been primarily used for closed sequential patterns mining.

Definition 13. (Forward-extension tiep) A tiep t^* is a *forward-extension* tiep of a pattern p if $vertical\ support(p) = vertical\ support({p^F}_{t^*})$, for p's super-pattern ${p^F}_{t^*} = p + t^* = < t_1, t_2, ..., t_k, t^* >$. The set of p's sequences database's records that support ${p^F}_{t^*}$ is denoted by $F(p, t^*)$.

Definition 14. (Backward-extension tiep) A tiep t^* is a *backward-extension* tiep of a pattern p if $\exists 1 \leq i \leq k$ s.t. $vertical\ support(p) = vertical\ support({p^{B_i}}_{t^*})$, for p's super-pattern ${p^{B_i}}_{t^*} = < t_1, t_2, ..., t_{i-1}, t^*, t_i, ..., t_k >$. The set of p's sequences database's records that support ${p^{B_i}}_{t^*}$ is denoted by $B^i(p, t^*)$.

Let a current pattern $p =< t_1, t_2, ..., t_k >$. TIRPClo's closure-checking scheme is based on the following claims.

Closure-Check 1. Assume that 1) X^+ and X^- are backward-extension tieps of p for $1 \leq i \leq j \leq k$, and that 2) $vertical\ support(p) = |\{record.entityID \mid record \in B^i(p, X^+) \cap B^j(p, X^-)\}|$. Then p, as well as any future extension of it, are not closed TIRPs.

Closure-Check 2. Assume that 1) X^+ is a backward-extension tiep of p for $1 \leq i \leq k$, 2) X^- is a forward-extension tiep of p, and that 3) $vertical\ support(p) = |\{record.entityID \mid record \in B^i(p, X^+) \cap F(p, X^-)\}|$. Then p is not a closed TIRP.

Closure-Check 3. Assume that X^+ is a forward-extension tiep of p. Then p is not a closed TIRP.

If the conditions for the first claim hold for p, it cannot be extended to form any frequent closed TIRP. Therefore, p is safely pruned. Otherwise, if either the conditions for the second or third claims hold for p, it is not a closed TIRP. Therefore, p is not added to the output of the algorithm, yet, it is not pruned.

Evaluation

Datasets

For the evaluation of TIRPClo we used several real-world datasets from various domains. The main properties of the datasets are summarized in Table 1 considering five parameters: $|E|$ — the number of entities; $|STIs|$ — the total number of STIs; $\dfrac{|STIs|}{|E|}$ — the mean number of STIs per entity; MHS — the mean horizontal support of a symbol within an entity; and $|S|$ — the number of symbol types in the dataset. Datasets' extreme conditions appear in bold.

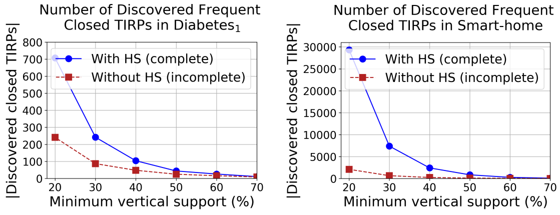
$Name$	$\|E\|$	$\|STIs\|$	$\frac{\|STIs\|}{\|E\|}$	MHS	$\|S\|$
ASL	65	2037	31.3	1.2	**146**
Diabetes$_1$	**2038**	**80538**	39.52	3.5	35
Diabetes$_2$	**2038**	**79281**	38.9	2.02	94
Smart-home	89	23213	**260.8**	**8.8**	95

Table 1: Evaluation datasets.

Experimental Results

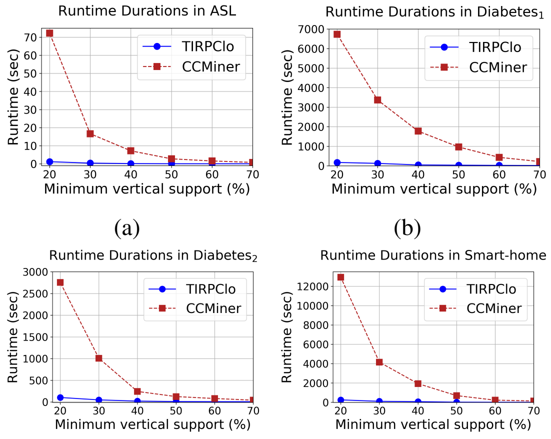
The evaluation focuses on the performance of TIRPClo in comparison to state-of-the-art closed TIRPs discovery methods (Chen, Weng, and Hui 2016), as well as its own performance with different parameters. For that, three main experiments have been designed. All methods were implemented in Visual C# and the experiments were conducted on a dell G5, having 16GB main memory, running Microsoft Windows 10. In each experiment we verified that the *same set of frequent closed TIRPs was discovered by all the compared methods.*

Experiment 1. Completeness in Closed TIRPs Mining First, in order to substantiate the need in the discovery of the horizontal support for completeness in frequent closed TIRPs mining, a quantitative analysis was designed. For that, TIRPClo was ran once without discovering the horizontally supporting instances of the TIRPs (as made in (Chen, Weng, and Hui 2016)), and then including horizontal support discovery, which is required for completeness. Figure 5 shows the number of frequent closed TIRPs that were discovered by each approach, using the smart-home and diabetes$_1$ datasets with minimum vertical support thresholds of 20%-70% and a fixed maximal gap value of 20. In the diabetes dataset, less than one third of all the frequent closed TIRPs were discovered without horizontal support discovery, already at 30% of minimum vertical support. In the smart-home dataset the differences were even larger, as less than 10% of the frequent closed TIRPs were discovered when using the previous, incomplete approach.



(a) (b)

Figure 5: Completeness in closed TIRPs mining.



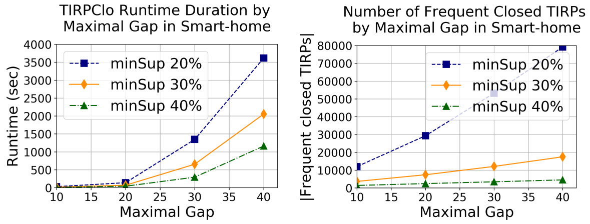
(a) (b)

(c) (d)

Figure 6: Runtime duration comparison.

Experiment 2. Runtime Duration Comparison In this experiment TIRPClo was compared to CCMiner (Chen, Weng, and Hui 2016), which is the only current method designed for closed TIRPs discovery. Note that CCMiner discovers only the first instances of the TIRPs within each supporting entity. Hence, its projection was repeatedly applied in order to discover the horizontally supporting instances of the TIRPs, which is essential for completeness. In addition, since TIRPClo uses the maximal gap constraint, it was also added to CCMiner, in order to conduct as fair comparison as possible between the methods, using minimum vertical support thresholds of 10%-70% and a fixed maximal gap value of 20.

Note that in some papers in the literature the comparisons were often at very low levels of minimum vertical support (even below 1%), which in fact does not usually show a meaningful difference. That is unlike a meaningful runtime duration difference already when having several dozens of percentages, as we show in this paper. Figure 6 summarizes the runtime duration results of the compared methods. In this figure it is clearly demonstrated that TIRPClo is much faster than CCMiner in all the datasets and already at very high levels of minimum vertical support. In fact, at minimum vertical support thresholds of 50%–60%, TIRPClo was at least ten times faster than CCMiner. At lower thresholds, e.g., 20%-30%, even larger speed-ups were reported, by at least a factor of 50 up to more than 100.



(a) (b)

Figure 7: Maximal gap analysis.

Experiment 3. Maximal Gap Analysis In this experiment, the goal was to evaluate the trade-off between TIRPClo's runtime duration and the number of discovered frequent closed TIRPs when changing the value of the maximal gap. For that, the smart-home dataset was used with maximal gap values between 10–40, having the minimum vertical support value fixed on each of the following thresholds: 20%, 30% and 40%. The results are shown in Figure 7. As expected, given a fixed minimum vertical support value, larger maximal gap values led to longer runtime durations. However, a 25% reduction in the maximal gap typically resulted not only in a 60%-70% shorter runtime duration, but also in a significant reduction of approximately 50% in the total number of discovered frequent closed TIRPs. Therefore, the computational aspect may play an important role when determining the desirable value of the maximal gap.

Discussion

In this work, we introduced TIRPClo – an efficient algorithm which is the first to completely discover the frequent closed TIRPs. First, we demonstrated the completeness problem in the current approach for frequent closed TIRPs discovery. Then, we introduced the TIRPClo algorithm, including its novel non-ambiguous sequential representation and the complete projection method. Finally, a comprehensive experimental plan was designed, which demonstrated the crucial need in the discovery of the TIRPs' horizontal support for completeness in frequent closed TIRPs mining. In addition, TIRPClo's significant runtime speed-ups compared to CCMiner were shown and the effects of changing the maximal gap parameter were empirically evaluated for the first time. For future work, we would like to extend TIRPClo to completely discover the entire set of frequent TIRPs, forming a robust framework for both fundamental time intervals mining tasks.

Acknowledgements

The authors wish to thank Prof. Panagiotis Papapetrou and Prof. Diane J Cook for providing the datasets for the evaluation. This research was partially funded by a grant of the Israeli Ministry of Science and Technology 8760521. In addition, Omer Harel was funded by the Darom-Lachish scholarship of Kreitman School of Advanced Graduate Studies at Ben Gurion University. We also want to thank the anonymous reviewers for the insightful and supportive comments.

References

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

Azulay, R.; Moskovitch, R.; Stopel, D.; Verduijn, M.; De Jonge, E.; and Shahar, Y. 2007. Temporal Discretization of medical time series-A comparative study. In *IDAMAP 2007 workshop*.

Batal, I.; Sacchi, L.; Bellazzi, R.; and Hauskrecht, M. 2009. A temporal abstraction framework for classifying clinical temporal data. In *AMIA Annual Symposium Proceedings*, volume 2009, 29. American Medical Informatics Association.

Chang, L.; Wang, T.; Yang, D.; and Luan, H. 2008. Seqstream: Mining closed sequential patterns over stream sliding windows. In *2008 Eighth IEEE International Conference on Data Mining*, 83–92. IEEE.

Chen, Y.-C.; Weng, J. T.-Y.; and Hui, L. 2016. A novel algorithm for mining closed temporal patterns from interval-based data. *Knowledge and Information Systems* 46(1): 151–183.

Höppner, F. 2002. Time series abstraction methods–a survey. *Informatik bewegt: Informatik 2002-32. Jahrestagung der Gesellschaft für Informatik ev (GI)* .

Huang, K.-Y.; Chang, C.-H.; Tung, J.-H.; and Ho, C.-T. 2006. COBRA: closed sequential pattern mining using bi-phase reduction approach. In *International Conference on Data Warehousing and Knowledge Discovery*, 280–291. Springer.

Kostakis, O.; and Gionis, A. 2017. On mining temporal patterns in dynamic graphs, and other unrelated problems. In *International Conference on Complex Networks and their Applications*, 516–527. Springer.

Lavrac, N.; Keravnou, E.; and Zupan, B. 2000. Intelligent data analysis in medicine. *Encyclopedia of computer science and technology* 42(9): 113–157.

Lin, J.; Keogh, E.; Lonardi, S.; and Chiu, B. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, 2–11.

Mörchen, F.; and Ultsch, A. 2005. Optimizing time series discretization for knowledge discovery. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, 660–665.

Moskovitch, R.; and Shahar, Y. 2015a. Classification-driven temporal discretization of multivariate time series. *Data Mining and Knowledge Discovery* 29(4): 871–913.

Moskovitch, R.; and Shahar, Y. 2015b. Fast time intervals mining using the transitivity of temporal relations. *Knowledge and Information Systems* 42(1): 21–48.

Moskovitch, R.; Walsh, C.; Wang, F.; Hripcsak, G.; and Tatonetti, N. 2015. Outcomes prediction via time intervals related patterns. In *2015 IEEE international conference on data mining*, 919–924. IEEE.

Papapetrou, P.; Kollios, G.; Sclaroff, S.; and Gunopulos, D. 2009. Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems* 21(2): 133.

Patel, D.; Hsu, W.; and Lee, M. L. 2008. Mining relationships among interval-based events for classification. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, 393–404.

Sacchi, L.; Larizza, C.; Combi, C.; and Bellazzi, R. 2007. Data mining with temporal abstractions: learning rules from time series. *Data Mining and Knowledge Discovery* 15(2): 217–247.

Shahar, Y. 1997. A framework for knowledge-based temporal abstraction. *Artificial intelligence* 90(1): 79–133.

Sharma, A. K.; and Patel, D. 2018. STIPA: A Memory Efficient Technique for Interval Pattern Discovery. In *2018 IEEE International Conference on Big Data (Big Data)*, 1767–1776. IEEE.

Tzvetkov, P.; Yan, X.; and Han, J. 2005. TSP: Mining top-k closed sequential patterns. *Knowledge and Information Systems* 7(4): 438–457.

Wang, J.; and Han, J. 2004. BIDE: Efficient mining of frequent closed sequences. In *Proceedings. 20th international conference on data engineering*, 79–90. IEEE.

Winarko, E.; and Roddick, J. F. 2007. ARMADA–An algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering* 63(1): 76–90.

Wu, S.-Y.; and Chen, Y.-L. 2007. Mining nonambiguous temporal patterns for interval-based events. *IEEE transactions on knowledge and data engineering* 19(6): 742–758.

Yan, X.; Han, J.; and Afshar, R. 2003. CloSpan: Mining: Closed sequential patterns in large datasets. In *Proceedings of the 2003 SIAM international conference on data mining*, 166–177. SIAM.