KAPMiner: Mining ordered association rules with constraints*

Isak Karlsson, Panagiotis Papapetrou, and Lars Asker

Dept. of Computer and Systems Sciences, Stockholm University, Sweden
{isak-kar,panagiotis,asker}@dsv.su.se

Abstract. We study the problem of mining ordered association rules from event sequences. Ordered association rules differ from regular association rules in that the events occurring in the antecedent (left hand side) of the rule are temporally constrained to occur strictly before the events in the consequent (right hand side). We argue that such constraints can provide more meaningful rules in particular application domains, such as health care. The importance and interestingness of the extracted rules are quantified by adapting existing rule mining metrics. Our experimental evaluation on real data sets demonstrates the descriptive power of ordered association rules against ordinary association rules.

1 Introduction

Extracting rules from a set of transactions is a problem that has been studied extensively over the past two decades [2, 1, 28, 16]. Transactional databases typically comprise sets of items or events grouped together in transactions. Each transaction can either be treated as a "bag" of items or as an ordered set of time-stamped events, hence, introducing a sequential order within each transaction. For example, in the former case, we can have transactions of customer activity in the form of basket data containing sets of items bought together from a store [1], while in the latter case, transactions appear in the form of sequential, time-stamped data, such as financial transactions or electronic health records [2].

Hence, sequential pattern and rule mining [2, 18] extends traditional association rule mining to exploit the temporal information present in event datasets collected over time, by adding a time-stamp to each event. Such formulation might, however, fail to find interesting patterns, due to the fact that it is too restrictive by requiring a specific order of all items occurring in a sequential pattern or rule. For many real world applications, such as fault detection, or treatment recommendations in healthcare, it is not uncommon with procedures describing best practices or recommended actions that should be taken after certain preconditions are satisfied. Therefore, ordered association rules can be formed, where each rule is defined by a set of preconditions (i.e., antecedent) and a set of recommended actions (i.e., consequent). Such rules suggest that the set of recommended actions should be followed, irrespective of order, after the set of preconditions is satisfied, irrespective of order. Depending on the application

* This is an Accepted Manuscript of an article published by Springer In *Proceedings of the International Symposium on Intelligent Data Analysis*, 2017.

domain at hand, it would also be desirable to optionally include time constraints imposed on the temporal separation between each pair of items in the antecedent and consequent, respectively.

In this paper, we study the problem of mining ordered association rules with temporal constraints, and propose an efficient algorithm, called KAPMiner, to solve it. This problem has been studied in recent works in the form of temporal rule mining [5–7]. Nonetheless, KAPMiner can achieve competitive computational efficiency against state-of-the-art, while additionally supporting temporal constraints on the extracted rules. Next, we emphasize the importance of the problem at hand by providing an example from the healthcare domain.

Example. Consider a dataset containing administrative health records of patients having suffered from Heart Failure. Each patient record can be seen as a transaction, where time-stamped healthcare-related events can occur over time. Such events can be medical diagnoses, drug prescriptions, or treatment procedures. An example of an ordered rule of substantial interestingness and importance is the following:

$$\{Heart\ Failure\} \Rightarrow \{ACE\ inhibitors, \beta\text{-}blockers\}$$

This rule suggests that when a patient is diagnosed with Heart Failure (rule precondition or antecedent), a recommended action (rule consequence) is to follow a treatment including two drugs, ACE[1] inhibitors and β-blockers, which is in compliance with the guidelines issued by the Swedish National Board of Health and Welfare[17]. Since it is common that the two drugs included in the consequent part of the rule are not necessarily prescribed concurrently or in a particular order, a typical sequential rule would not be able to partially ignore their temporal order in the patient record. At the same time, a typical association rule would not be able to capture the particular temporal order between the antecedent precondition and the consequent treatment. Hence, ordered temporal association rules can be highly useful and meaningful in this domain.

Related work. Sequential pattern mining and association rule mining are very challenging tasks, since the search space is typically large [2]. Standard apriori-based algorithms employ a bottom-up search, enumerating every single frequent pattern, and then construct association rules based on the extracted patterns. The main characteristic of these approaches is that they apply the *Apriori principle* [1]. A more efficient approach, GSP [18], introduces time and window constraints into the pattern extraction process. At the same time, the notion of frequent episode mining [12, 10] has been extensively studied in the literature. Frequent episode discovery has several applications in, among others, discovering local patterns in complex processes, conformance checking based on partial orders, medical process mining. Frequent episodes refer to event patterns occurring in a single sequence, that are partially ordered based on a predefined set of temporal relations. They are orthogonal to our formulation, since we are interested in patterns occurring frequently within a set of sequential transactions and not within a single one, while we have no predefined set of relations in our mining process.

[1] angiotensin-converting-enzyme

An alternative candidate generation approach for frequent itemsets and sequential patterns employs tree-like data structures, such as an FP-tree [8] for frequent itemsets, or a set-enumeration tree [4] for sequences. The key idea is to traverse the candidate space in a depth-first search manner for enumerating all the candidate patterns. Examples of such algorithms include SPAM [3], SPADE [26], and GO-SPADE [11]. Finally, another class of sequential pattern mining algorithms includes prefix-based candidate generation approaches [15, 21, 25], Similar approaches, both tree-based and prefix-based, have been proposed for mining closed itemsets and closed sequential patterns [27, 14, 21, 25].

In parallel, several studies have been focusing on alternative interestingness measures for association rules [19], except for support and confidence, aiming at removing redundancy and limiting the number of extracted rules to the most "interesting" ones. Alternative association rule measures and techniques have been proposed [13, 9, 20] for evaluating the importance of association rules in transactional databases, while generic and interactive techniques have been proposed [22, 24] for effectively controlling the mining process and restricting the number of insignificant rules. Finally, there has been some work on constraint-based mining of frequent itemsets, where the goal is to mine the top K patterns that maximize an interestingness measure (other than the typical support threshold) and satisfy a set of constraints [23]. More recently, a novel framework for mining the top-K sequential patterns under leverage has been proposed [16], where a novel definition of the expected support of a sequential pattern is presented along with an efficient branch-and-bound approach for mining sequential patterns under the new interestingness measure. Moreover, a statistical testing approach for association rules extracted from uncertain data combines an analytic with simulative processes for correcting the statistical test for distortions caused by data uncertainty [28].

In summary, the literature on mining rules of itemsets and sequential patterns is quite extensive, and a thorough review is far from the main objectives of this paper. Nonetheless, all existing approaches focus on more general types of rules, and are hence orthogonal to the formulation of this paper. The concept of *ordered rules* defined in this paper, suggests that the antecedent and consequent of a rule are seen as two bags of events separated by a temporal constraint regulating that the bag of antecedent events should be separated by at least d time units from the bag of consequent events. To the best of our knowledge, we are the first to introduce this formulation. A similar formulation, with a looser temporal constraint (i.e., when $d = 0$, defined as Problem 1 in our paper), has been introduced along with three algorithmic solutions, i.e., RuleGrowth [7] and `ERMiner` [6], and CMDeo [5]. Still, the algorithm proposed in this paper is shown to be more efficient in terms of computational time than the competitors, while it can also solve the constrained version of the problem.

Contributions. Our main contributions in this paper include: (1) the formulation of the novel problem of mining ordered association rules with temporal constraints from transactions of time-stamped event sequences, (2) a novel and efficient algorithm to solve the problem by employing a tree-based rule enumeration process and by applying effective pruning techniques both on the antecedent and consequent sets of the rules, (3) an extensive experimental evaluation on several real datasets against state-of-the-art methods, where we demonstrate that

our approach can achieve speedups of up to two orders of magnitude, when the dataset contains very dense or large sequences, and finally (4) a case-study in the area of healthcare, where our formulation can identify meaningful rules, when the temporal contraints are applied.

2 Problem Setting

Let Σ be an alphabet of event labels. A *time-stamped event* is defined as a tuple $E = \langle l, t \rangle$, where $l \in \Sigma$ and t is the time occurrence of the event. For notation purposes, the label and time stamp of an event E are denoted as $E.l$ and $E.t$, respectively.

Identifier	**Transaction**
$\mathcal{T}_1$	$\{\langle 1,0\rangle, \langle 2,1\rangle, \langle 3,1\rangle, \langle 1,1\rangle, \langle 1,4\rangle\}$
$\mathcal{T}_2$	$\{\langle 2,0\rangle, \langle 1,1\rangle, \langle 3,3\rangle, \langle 4,3\rangle, \langle 3,4\rangle\}$
$\mathcal{T}_3$	$\{\langle 1,0\rangle, \langle 2,1\rangle, \langle 3,2\rangle\}$
$\mathcal{T}_4$	$\{\langle 1,0\rangle, \langle 1,0\rangle, \langle 3,4\rangle\}$
$\mathcal{T}_5$	$\{\langle 3,0\rangle, \langle 1,1\rangle, \langle 1,2\rangle\}$

Identifier	**Rule**	**Support**
r1	$\{1\} \Rightarrow \{3\}$	0.4
r2	$\{2\} \Rightarrow \{1\}$	0.2
r3	$\{2\} \Rightarrow \{3\}$	0.2
r4	$\{2,3\} \Rightarrow \{1\}$	0.2
r5	$\{1,2\} \Rightarrow \{3\}$	0.2

Table 1: Example of ordered rules extracted from the transaction database for $\mu = 0.2, \nu = 0.3$, and $\delta = 3$.

A *temporal transaction* $\mathcal{T}$ is a set of events ordered by their respective time stamps, and the size of a transaction is the number of time-stamped events it contains. For example, temporal transaction $\mathcal{T} = \{E_1, \ldots, E_N\}$ is of size N. A set of temporal transactions $\mathcal{D}$ constitutes a *temporal-transaction database.*

Example 1. An example of a temporal-transaction database (of 5 transactions) is shown in Table 1 (left). Each tuple represents an event consisting of a label (in the example, $\Sigma = \{1, 2, 3, 4\}$) and a time point. For instance, transaction $\mathcal{T}_1$ shows that event label 1 occurred *one* time step before $2, 3$, and 1 and *four* time steps before 1.

Definition 1 (ordered rule). *Given two subsets of event labels, i.e., $X \subseteq \Sigma$ and $Y \subseteq \Sigma$, with $\mathcal{X} \cap \mathcal{Y} = \emptyset$, we define an* ordered rule *as follows:*

$$r : \mathcal{X} \Rightarrow \mathcal{Y} \ ,$$

such that $\forall E_i \in \mathcal{X}$, $\nexists E_j \in \mathcal{Y}$ with $E_i.t - E_j.t \geq \delta$, with $\delta \in \mathbb{R}$ and $\delta \geq 0$.

One of the key tasks in our paper is to determine whether a rule occurs in a temporal transaction.

Definition 2 (temporal occurrence of a rule). *We say that an ordered rule $r : \mathcal{X} \Rightarrow \mathcal{Y}$ occurs in a temporal transaction $\mathcal{T}$, if all events in $\mathcal{X} \cup \mathcal{Y}$ occur in $\mathcal{T}$ at least once, and each and every event in $\mathcal{X}$ has at least one occurrence before each and every event in $\mathcal{Y}$. More formally, $r \in \mathcal{T}$, if*

- $\exists E \in \mathcal{T}, \forall E \in \mathcal{X} \cup \mathcal{Y}$
- $\forall (E, E')$ *with* $E \in \mathcal{X}, E' \in \mathcal{Y}$, $\nexists (E, E'), E.l, E'.l \in \mathcal{T}$, *s.t.* $E.t - E'.t \geq \delta$.

Example 2. For $\delta = 3$, the rule $\{2, 3\} \Rightarrow \{1\}$ is contained in transaction $\mathcal{T}_1$, whereas $\{1\} \Rightarrow \{3\}$ is not, because $\{3\}$ does not occur within at least 3 time-steps from $\{1\}$.

There are several ways of assessing the quality of an ordered rule. In this paper, our main objective is to emphasize the importance of ordered rules in terms of (1) the frequency of the rule in a given database, (2) the frequency of the particular temporal order for a given set of event labels compared to any other temporal order, (3) the conditional probability of the consequent set of labels given the precedent set, and (4) the degree of dependence between the occurrence probabilities of the precedent and consequent sets.

Next, we present four quality metrics for an ordered rule. Given an ordered rule $r : \mathcal{X} \Rightarrow \mathcal{Y}$ and a temporal transaction database $\mathcal{D}$, we have:

- the **support** of r in $\mathcal{D}$ is the fraction of transactions of $\mathcal{D}$ containing at least one temporal occurrence of r, i.e.,

$$sup(r, \mathcal{D}) = \frac{|r \in \mathcal{D}|}{|\mathcal{D}|}$$

- the **support ratio** of r in $\mathcal{D}$ is the fraction of transactions containing r divided by the fraction of transactions containing itemset $\mathcal{X} \cup \mathcal{Y}$, i.e.,

$$r - sup(r, \mathcal{D}) = \frac{sup(r, \mathcal{D})}{sup(\mathcal{X} \cup \mathcal{Y}, \mathcal{D})} = \frac{|r \in \mathcal{D}|}{|\mathcal{X} \cup \mathcal{Y} \in \mathcal{D}|}$$

- the **confidence** of r in $\mathcal{D}$ is the fraction of transactions containing r divided by the fraction of transactions containing all items in the consequent set of the rule, i.e.,

$$conf(r, \mathcal{D}) = \frac{sup(r, \mathcal{D})}{sup(\mathcal{Y}, \mathcal{D})} = \frac{|r \in \mathcal{D}|}{|\mathcal{Y} \in \mathcal{D}|}$$

- the **lift** of r in $\mathcal{D}$ is the frequency of r in $\mathcal{D}$ divided by the product of the frequencies of the precedent and consequent sets, i.e.,

$$lift(r, \mathcal{D}) = \frac{sup(r, \mathcal{D})}{sup(\mathcal{X}, \mathcal{D}) \cdot sup(\mathcal{Y}, \mathcal{D})} = \frac{|r \in \mathcal{D}| \cdot |\mathcal{D}|}{|\mathcal{X} \in \mathcal{D}| \cdot |\mathcal{Y} \in \mathcal{D}|}$$

Problem 1 (mining ordered association rules). Given a temporal transaction database $\mathcal{D}$, a minimum support threshold μ, and a minimum support ratio threshold ν, we want to find the set of ordered rules $\mathcal{R}$, such that for each $r \in \mathcal{R}$, it holds that $sup(r, \mathcal{D}) \geq \mu$ and $r - sup(r, \mathcal{D}) \geq \nu$.

Problem 2 (mining constrained association rules). Given a temporal transaction database $\mathcal{D}$, a minimum support threshold μ, a minimum support ratio threshold ν, and a temporal constraint δ, find the set of ordered rules $\mathcal{R}^{\delta}$, such that for each $r \in \mathcal{R}^{\delta}$, it holds that $sup(r, \mathcal{D}) \geq \mu$, $r - sup(r, \mathcal{D}) \geq \nu$ and $\forall E_i \in \mathcal{X}$, $\nexists E_j \in \mathcal{Y}$ with $E_i.t - E_j.t \geq \delta$. Note this problem reduces to Problem 1 for $\delta = 0$.

3 Mining Ordered Association Rules

We introduce KAPMiner, an algorithm for identifying ordered association rules with optional temporal constraints. The main operator of the algorithm (see Algorithm 1) is based on the concepts of antecedent and consequent matching and merging of rules, which are identified in the hierarchy of frequent itemsets.

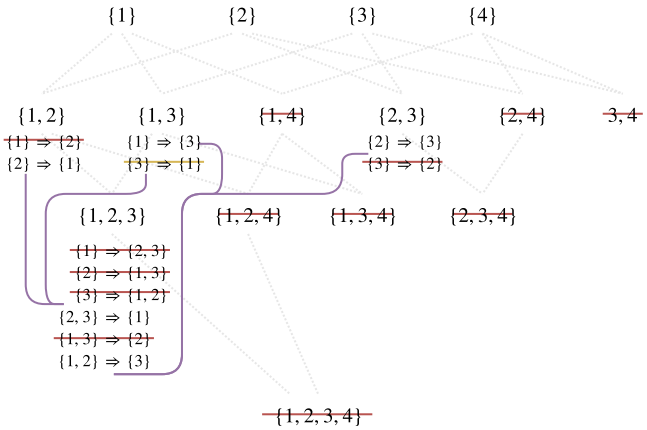

Fig. 1: Search space and pruning of the KAPMiner algorithm. For instance, the rule $\{2,3\} \Rightarrow \{1\}$ is formed by merging the consequent matching rules $\{2\} \Rightarrow \{1\}$ and $\{3\} \Rightarrow \{1\}$, whereas rule $\{1,2\} \Rightarrow \{3\}$ is formed by antecedent matching rules $\{1\} \Rightarrow \{3\}$ and $\{2\} \Rightarrow \{3\}$.

Definition 3 (Antecedent and consequent matching). *A pair of rules* $r : \{X\} \Rightarrow \{Y\}$ *and* $r' : \{X'\} \Rightarrow \{Y'\}$ *is said to be antecedent matching, if* $\{X\} \setminus \{X'\} = \emptyset$*, and said to be consequent matching, if* $\{Y\} \setminus \{Y'\} = \emptyset$*.*

The above definition can be extended to more than two rules, hence resulting in a set of antecedent or consequent matching rules, denoted as r_a and r_c, respectively. The common antecedent in r_a is denoted as $r_a.antecedent$, while the set of consequents is denoted as $r_a.consequents$.The notation is equivalent for r_c. To explore the search-space of possible frequent constrained ordered rules our algorithm merges frequent antecedent or consequent matching rules.

Definition 4 (Antecedent and consequent merging). *Given a set of antecedent or consequent matching rules* r_a *or* r_c*, respectively, the intersection of transactions where rules with matching consequent or antecedent occur, is the support of the rule with antecedents or consequents merged, respectively.*

For instance, if $\{3\} \Rightarrow \{1\} \in \mathcal{R}_c$ occurs in transaction $\{\mathcal{T}_1\}$ and $\{2\} \Rightarrow \{1\} \in \mathcal{R}_c$ occurs in $\{\mathcal{T}_1\}$, then $\{2\} \cup \{3\} \Rightarrow \{1\}$ occurs in the intersecting transactions: $\{\mathcal{T}_1\} \cap \{\mathcal{T}_1\} = \{\mathcal{T}_1\}$.

For a rule-merge to be valid for a rule consisting of elements from an itemset $\mathcal{I}$ on level k, the number of frequent rules in the antecedent or consequent matching rule set, identified from all subsets of itemsets on level $k-1$, must satisfy the condition $|r_a.antecedents| + |X| = |\mathcal{I}|$ or $|r_c.consequents| + |Y| = |\mathcal{I}|$, for antecedent and consequent merges respectively. For an antecedent or consequent merge, the *antimonotonicity property* holds since the resulting rule contains exactly one item more in either the antecedent or the consequent, the rule can only appear in the same number of transactions or less. From the antimonicity property it follows that if a rule is infrequent, it should not be used in any antecedent or consequent merges.

Algorithm 1 The KAPMiner algorithm

```
procedure KAPMiner(𝒟, minSup, minConf, δ)
    Let 𝒞 = [ℐ_1, ..., ℐ_m] be a vector of frequent event labels
    Let ℐ_i.tid be a set that identifies which transactions an itemset occurs in
    ℛ ← ∅, k ← 1
    do
        𝒩 ← ∅
        for i ← 0 until |𝒞| do
            for j ← 0 until |𝒞| such that PrefixMatch(ℐ_i, ℐ_j, k − 1) do
                ℐ′ ← MergeItemset(ℐ_i, ℐ_j)
                ℐ′.tid ← ℐ_i.tid ∩ ℐ_j.tid
                if sup(ℐ′) ≥ minSup then
                    𝒩 ← 𝒩 ∪ {ℐ′}
                    if k > 1 then
                        Let 𝒮 be the rules associated with each subset of ℐ′ in 𝒞
                        ℐ′.rules ← MergeRules(𝒮, k, minSup)
                    else
                        ℐ′.rules ← ItemRules(ℐ_i, ℐ_j, δ, minSup)
                    end if
                    ℛ ← ℛ ∪ {r | r ∈ ℐ′.rules ∧ conf(r) ≥ minConf}
                end if
            end for
        end for
        𝒞 ← 𝒩
        k ← k + 1
    while 𝒩 ≠ ∅
    return ℛ
end procedure
```

The algorithm, which expects a temporal transaction database, a minimum support threshold, a minimum confidence threshold and temporal constraint (δ), starts by scanning the temporal transaction database $\mathcal{D}$ once, to associate with each frequent item a set of transactions where the item occurs[2]. The algorithm then proceeds by identifies the frequent k-level itemsets using the $(k-1)$-level itemsets and prunes the infrequent ones. At level $k = 1$ rules are formed by the ItemRules-procedure which forms frequent ordered rules from two single item itemsets, by comparing the first and last position of each item in each transaction.

To construct rules for itemsets of size larger than two, the MergeRules-procedure is employed. This procedure expects a set of rules which are found among the $k-1$ level itemsets that are subsets of the newly formed itemset at level k. The procedure then iterates over the sets of antecedent matching rules and forms new rules by merging the consequents and taking the intersection of transaction identifiers as the support of the rule. The procedure proceeds similarly with the consequent matching rules and forms new rules by merging the

[2] While not presented in the algorithm, the scan also records the first and last position of each item in each transaction, which is used to efficiently form rules with a single item as antecedent and consequent

Algorithm 2 Constructing rules from 1-item itemsets

procedure ItemRules($\mathcal{I}_i, \mathcal{I}_j, \delta, minSup$)
  $r_1 \leftarrow \{I_i\} \Rightarrow \{I_j\}$ and $r_2 \leftarrow \{I_j\} \Rightarrow \{I_i\}$
  Let $r_1.tid$ be the set of transactions where Last($\mathcal{I}_j, \mathcal{T}$) - First($\mathcal{I}_i, \mathcal{T}$) $\geq \delta$ and let $r_2.tid$ be the set of transactions where Last($\mathcal{I}_i, \mathcal{T}$) - First($\mathcal{I}_j, \mathcal{T}$) $\geq \delta$
  return $\{r \mid r \in \{r_1, r_2\} \wedge sup(r) \geq minSup\}$
end procedure

antecedents and taking the intersection of transactions where the rules appears as the support. Finally, infrequent rules are pruned, and the size of search-space is reduced by pruning both infrequent itemsets (which are used to associate possible rule merges) and rules.

Algorithm 3 Merging the rules of n-item itemsets

procedure MergeRules($\mathcal{S}, k, minSup$)
  Let $\mathcal{S}_a$ be a list of sets consisting of rules with matching antecedent and let $\mathcal{S}_c$ be a list of sets consisting of rules with matching consequent
  $\mathcal{R} \leftarrow \emptyset$
  for each set of rules with matching antecedent $r_a \in \mathcal{S}_a$ **do**
    if $|r_c.consequents| = k + 1 - |r_a.antecedent|$ **then**
      $r' \leftarrow \{r_a.antecedent\} \Rightarrow \{$MergeItemset$(r_a.consequents)\}$
      $r'.tid \leftarrow \bigcap_{r \in r_a} r.tid$
      if $sup(r') \geq minSup$ **then**
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{r'\}$
      end if
    end if
  end for
  for each set of rules with matching antecedent $r_c \in \mathcal{S}_c$ **do**
    if $|r_c.antecedents| = k + 1 - |r_c.consequent|$ **then**
      $r' \leftarrow \{$MergeItemset$(r_c.antecedents)\} \Rightarrow \{r_c.consequent\}$
      $r'.tid \leftarrow \bigcap_{r \in r_a} r.tid$
      if $sup(r') \geq minSup$ **then**
        $\mathcal{R} \leftarrow \mathcal{R} \cup \{r'\}$ ▷ Set without duplicate rules.
      end if
    end if
  end for
  return $\mathcal{R}$
end procedure

4 Evaluation

To evaluate the performance of the proposed ordered rule mining algorithm, we compare it against the current state-of-the-art, `ERMiner` [6], in terms of ex-

ecution time for different values of minimum support threshold[3], using seven datasets with varying properties. The datasets used for comparing the algorithms are: `BIBLE` (*text sequence*, 36k sequences/13905 distinct items/average sequence length of 17.84), `BMS1` (*web logs*, 59k/497/2.42), `BMS2` (*web logs*, 77k/3340/4.62), `FIFA` (*web logs*, 20k/2990/34.74), `Kosarak25k` (*web logs*, 25k/14804/8.04), `MSNBC` (*web logs*, 33k/17/13.3) and `SIGN` (*sign language*, 800/290/52).

To evaluate the importance of mining constraint ordered rules, we evaluate the proposed algorithm practically for identifying rules from administrative claims. In the experiment we use a real temporal transaction database extracted from the VAL database, and consists of 79028 patients diagnosed with heart failure. Each sequence represent the medical history of a patient and contains items from an alphabet of 13000 distinct items (diagnoses, drugs and actions), with an average sequence length of 300 items. We identify rules with $\delta = \{1, 7, 14\}$ days between the antecedent and consequent.

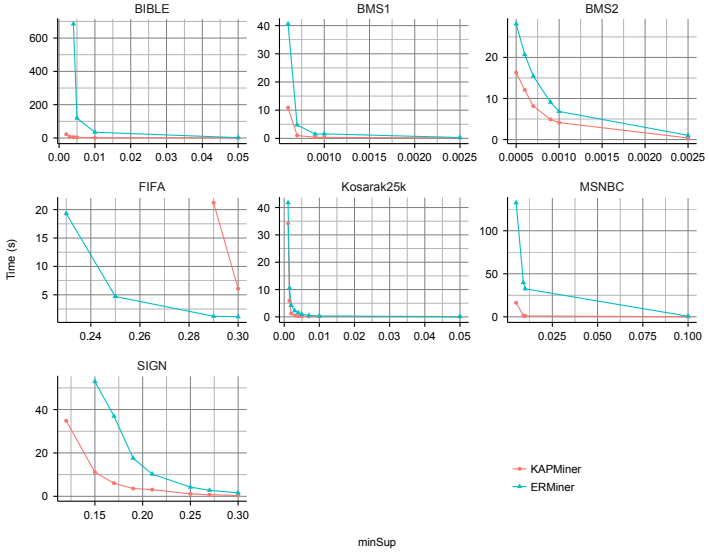

Fig. 2: Results for mining ordered rules from seven temporal transactions. Note that some points are missing, since the algorithms fail to complete within the memory constraint (e.g., `BIBLE` for `ERMiner` and `FIFA` for `KAPMiner`).

As seen in Figure 2, the results indicate that the proposed algorithm is often significantly faster than `ERMiner`, but not consistently so. For instance, it can be observed that the performance gap generally expands when the minimum support threshold is lowered, which can be explained by the fact that the proposed algorithm is able to prune more rules than the competitor. More specifically, the algorithm proposed here will prune any rule for which union of the antecedent and consequent is infrequent in an hierarchical manner, which results in excellent performance on datasets with many but infrequent items such as then `BIBLE`

[3] The experiments was performed using a 2.5GHz Intel i7 processor and 16GB of RAM. Both algorithms are implemented in Java.

dataset. But results in low performance on the `FIFA` dataset, as the algorithm fails to prune infrequent itemsets early.

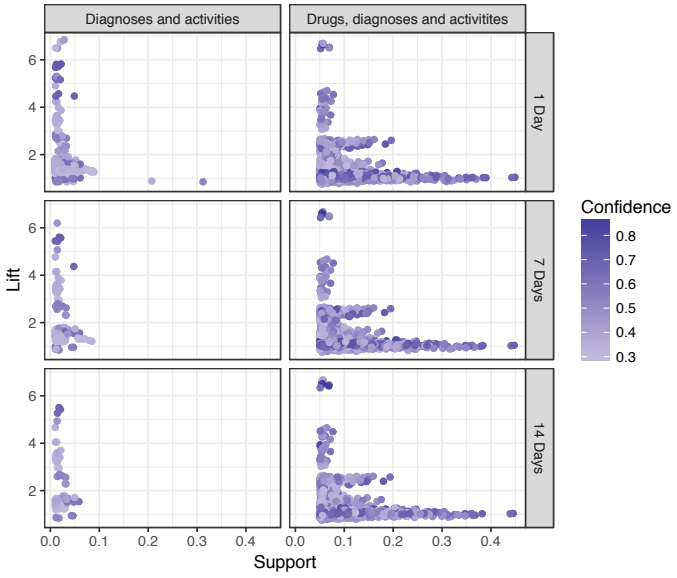

Fig. 3: Rule extraction for different values for δ with $\mu \geq 0.05$ and $\nu \geq 0.8$.

Figure 3, shows the support, lift and confidence for rules found during the case study, with a minimum support of 0.05 and a minimum support ratio of 0.8. The figure indicates that there are few high-lift rules, which has relatively low support, whereas there are a rather many high confidence rules. An example of an interesting rule that was found for $\delta = 7$, is:

$$\{hypothyroidism\} \Rightarrow \{levothyroxine, furosemide\}$$

with lift=6.6, confidence=0.69 and support ratio=0.88, which indicates that a common treatment pattern for hypothyroidism is a manufactured form of thyroid hormone and a medication to treat fluid build up.

5 Conclusions

We proposed a novel algorithm for identifying ordered rules where the consequent and antecedent are temporally separated. The proposed algorithm uses an apriori style algorithm for identifying frequent itemsets and uses temporal orderings within those itemsets to construct ordered rules using a computationally efficient transaction intersection algorithm. The experimental evaluation shows that the algorithm is up an order of magnitude faster than current state of the art when the dataset contains dense or very long sequences, but slow in cases where the dataset contains many frequent itemsets. Furthermore, in a case study for iden-

tifying interesting rules in the treatment of heart failure patients, the temporal constraint is shown to help in identifying patient trajectories.

Source code. The source code for replicating the experiments is available at the supporting website (`https://people.dsv.su.se/~isak-kar/orule/`).

Acknowledgments

This work was partly supported by grants provided by the Stockholm County Council (SU-SLL). The work of Panagiotis Papapetrou was also partly supported by the VR-2016-03372 Swedish Research Council Starting Grant.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proc. of VLDB. pp. 487–499 (1994)
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proc. of IEEE ICDE. pp. 3–14 (1995)
3. Ayres, J., Gehrke, J., Yiu, T., Flannick, J.: Sequential pattern mining using a bitmap representation. In: Proc. of ACM SIGKDD. pp. 429–435 (2002)
4. Bayardo, R.J.: Efficiently mining long patterns from databases. In: Proc. of ACM SIGMOD. pp. 85–93 (1998)
5. Fournier-Viger, P., Faghihi, U., Nkambou, R., Nguifo, E.M.: Cmrules: Mining sequential rules common to several sequences. Know.-Based Syst. 25(1), 63–76 (Feb 2012), `http://dx.doi.org/10.1016/j.knosys.2011.07.005`
6. Fournier-Viger, P., Gueniche, T., Zida, S., Tseng, V.S.: ERMiner: Sequential Rule Mining Using Equivalence Classes, pp. 108–119. Springer International Publishing, Cham (2014), `http://dx.doi.org/10.1007/978-3-319-12571-8_10`
7. Fournier-Viger, P., Nkambou, R., Tseng, V.S.M.: Rulegrowth: Mining sequential rules common to several sequences by pattern-growth. In: Proceedings of the 2011 ACM Symposium on Applied Computing. pp. 956–961. SAC '11, ACM, New York, NY, USA (2011), `http://doi.acm.org/10.1145/1982185.1982394`
8. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov. 8(1), 53–87 (2004)
9. Kamber, M., Shinghal, R.: Evaluating the interestingness of characteristic rules. In: Proc. of ACM SIGKDD. pp. 263–266 (1996)
10. Laxman, S., Sastry, P.S., Unnikrishnan, K.P.: A fast algorithm for finding frequent episodes in event streams. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2007), San Jose, USA. p. 410–419. Association for Computing Machinery, Inc. (August 2007), `https://www.microsoft.com/en-us/research/publication/a-fast-algorithm-for-finding-frequent-episodes-in-event-streams/`
11. Leleu, M., Rigotti, C., Boulicaut, J., Euvrard, G.: Go-spade: Mining sequential patterns over databases with consecutive repetitions. In: Proc. of MLDM. pp. 293–306 (2003)
12. Mannila, H., Toivonen, H., Verkamo, A.: Discovering frequent episodes in sequences. In: Proc. of ACM SIGKDD. pp. 210–215 (1995)
13. Omiecinski, E.R.: Alternative interest measures for mining associations in databases. IEEE Transactions On Knowledge and Data Engineering 15(1), 39–79 (2003)

14. Pei, J., Han, J., Mao, R.: Closet: An efficient algorithm for mining frequent closed itemsets. In: Proc. of DMKD. pp. 11–20 (2000)
15. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. of IEEE ICDE. pp. 215–224 (2001)
16. Petitjean, F., Li, T., Tatti, N., Webb, G.I.: Skopus: Mining top-k sequential patterns under leverage. Data Min. Knowl. Discov. 30(5), 1086–1111 (2016), http://dx.doi.org/10.1007/s10618-016-0467-9
17. Socialstyrelsen: Nationella riktlinjer för hjärtsjukvård (2015), http://www.socialstyrelsen.se
18. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: Proc. of EDBT. pp. 3–17 (1996)
19. Tan, P., Kumar, V.: Interestingness measures for association patterns: A perspective. Tech. Rep. TR00-036, Department of Computer Science, University of Minnesota (2000)
20. Tan, P., Kumar, V., Srivastava, J.: In: Proc. of ACM SIGKDD. pp. 183–192 (July 2002)
21. Wang, J., Han, J.: Bide: Efficient mining of frequent closed sequences. In: Proc. of IEEE ICDE. pp. 79–90 (2004)
22. Webb, G.I.: Discovering significant rules. In: Proc. of ACM SIGKDD (2006)
23. Webb, G.I., Zhang, S.: K-optimal rule discovery. Data Min. Knowl. Discov. 10(1), 39–79 (2005)
24. Xin, D., Shen, X., Mei, Q., Han, J.: Discovering interesting patterns through user's interactive feedback. In: Proc. of ACM SIGKDD (2006)
25. Yan, X., Han, J., Afshar, R.: Clospan: Mining closed sequential patterns in large databases. In: Proc. of SDM (2003)
26. Zaki, M.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning 40, 31–60 (2001)
27. Zaki, M., Hsiao, C.: Charm: An efficient algorithm for closed itemset mining. In: Proc. of SIAM. pp. 457–473 (2002)
28. Zhang, A., Shi, W., Webb, G.I.: Mining significant association rules from uncertain data. Data Min. Knowl. Discov. 30(4), 928–963 (2016)