Knowledge-Based Systems 161 (2018) 313–328

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

LTARM: A novel temporal association rule mining method to understand toxicities in a routine cancer treatment

Dang Nguyen[a], Wei Luo[a,*], Dinh Phung[b], Svetha Venkatesh[a]

[a] Center for Pattern Recognition and Data Analytics, School of Information Technology, Deakin University, Geelong, Australia
[b] Faculty of Information Technology, Monash University, Clayton Campus, VIC 3800, Australia

ARTICLE INFO

Article history:
Received 5 January 2018
Revised 18 July 2018
Accepted 23 July 2018
Available online 27 August 2018

Keywords:
Cancer treatment
Toxicity
Pairwise association analysis
Data mining
Temporal association rules

ABSTRACT

Cancer is a worldwide problem and one of the leading causes of death. Increasing prevalence of cancer, particularly in developing countries, demands better understandings of the effectiveness and adverse consequences of different cancer treatment regimes in real patient populations. Current understandings of cancer treatment toxicities are often derived from either "clean" patient cohorts or coarse population statistics. Thus, it is difficult to get up-to-date and local assessments of treatment toxicities for specific cancer centers. To address these problems, we propose a novel and efficient method for discovering toxicity progression patterns in the form of temporal association rules (TARs). A temporal association rule is defined as a rule where the diagnosis codes in the right hand side (e.g., a combination of toxicities/complications) are temporally occurred after the diagnosis codes in the left hand side (e.g., a particular type of cancer treatment). Our method develops a lattice structure to efficiently discover TARs. More specifically, the lattice structure is first constructed to store all frequent diagnosis codes in the dataset. It is then traversed using the paternity relations among nodes to generate TARs. Our extensive experiments show the effectiveness of the proposed method in discovering major toxicity patterns in comparison with the temporal comorbidity analysis. In addition, our method significantly outperforms existing methods for mining TARs in terms of runtime.

1. Introduction

Cancer remains a major challenge in modern medicine and it affects more and more people [1]. Cancer treatments, in particular radical treatments, cause painful toxicities, from mild hair loss and ulcers in mouth, to more permanent damage to the body [2] and even life-threatening conditions [3]. Acknowledging the relatively low cure rate of cancer, whether to receive radical treatments, is a valid question for a patient to ask. Even though the types of toxicity effects of cancer treatments are well-known [4–7], their exact realization at different patient groups is often not well understood. With growing number of cancer cases, increasing variability of diagnoses and treatment options, there is a need to describe the temporal progression of toxicities among the real patient populations.

In a typical clinical setting, the data collection on the toxicities is often poor. It is challenging to classify and measure the wide gamut of cancer toxicities and complications. Although a number of toxicity scales exist [4,6,7], cancer centers often do not have resources to ensure completeness and accuracy of data collection. The good news is that many of the treatment toxicities and complications have already been documented, mainly for billing purposes, in the hospital medical coding. Can we utilize these codes to better understand treatment toxicities in the *true* patient population? Or even better, can we use them to measure toxicities, to predict how they will unfold as the treatment progress, and in the end to gauge an *optimal* treatment plan for each individual patient with different cancer diagnoses in different stages? This paper presents a step toward such a quest. We ask: With the medical coding, can we discover the most common progressive patterns of toxicities/complications among patients who receive radiation treatments?

To answer this question, one solution is to apply a recently proposed method for temporal comorbidity analysis [8,9]. In this approach, we first assess the pairwise associations between cancer treatments and diagnosis codes to determine their dependence or independence. We then perform Binomial tests to determine the temporal directions of these associations. This approach, however, is relatively insensitive to the temporal gap constraints between

* Corresponding author.
E-mail addresses: d.nguyen@deakin.edu.au (D. Nguyen), wei.luo@deakin.edu.au (W. Luo), dinh.phung@monash.edu (D. Phung), svetha.venkatesh@deakin.edu.au (S. Venkatesh).

https://doi.org/10.1016/j.knosys.2018.07.031
0950-7051/

cancer treatments and diagnosis codes, and hence is ineffective in revealing the temporal progression of toxicities and complications of cancer treatments. Moreover, it cannot uncover the co-occurrence of toxicities. In this paper, our goal is to overcome these weaknesses by developing a novel method based on the discovery of temporal association rules (TARs). Each temporal association rule is a rule where the rule antecedent is a particular type of cancer treatments and the rule consequent is a set of co-occurring toxicities. Because a cancer treatment or a toxicity is associated with a time-stamp, a temporal association rule can capture the temporal progression between them. For example, a rule "*1526900: Radiation treatment* $\rightarrow$ *(30d) R11: Nausea and E86: Volume depletion*" is interpreted as "after receiving a radiation treatment 30 days, patients suffer from both nausea and volume depletion".

Mining TARs is a challenging task since it requires high computational costs when dealing with large datasets. Existing methods [10–12] often include two phases: (1) mining frequent itemsets[1] and (2) generating TARs from discovered frequent itemsets. Both of two phases are time-consuming. To speed up the process of mining frequent itemsets in the phase 1, existing methods can use different algorithms including Apriori [14], FPGrowth [15], Eclat [16], and so on. The second phase, however, is often slow as these methods have to find rules from a large number of frequent itemsets. For example, given a frequent itemset Z, recent methods [10,12] have to find all proper subsets $X \subset Z$ (assume Z is a set of k items, they have to consider $2^k - 2$ subsets of Z) to generate rules in the form of $X \rightarrow Z \backslash X$. To overcome this limitation, we propose an efficient algorithm, named **LTARM** (*Lattice-based Temporal Association Rule Mining*), to discover a complete set of TARs from the dataset. Because **LTARM** constructs a lattice structure to encode the paternity relations among frequent itemsets, it only traverses the lattice to generate TARs instead of enumerating all possible subsets of each frequent itemset.

To summarize, we make the following contributions:

1. We propose to use TARs to discover the toxicity progression of cancer treatments. Different from the temporal comorbidity analysis method, our proposed method not only captures the temporal relations between a cancer treatment and toxicities but also uncovers the co-occurrence of toxicities.
2. We develop an efficient method for mining TARs with a lattice structure and two theorems which help to prune invalid rules quickly, thus reducing the number of candidate rules in the search space. Compared to existing methods, our method significantly accelerates the execution time (achieving 5–8 times faster than the baselines).
3. We visualize discovered rules using interactive network graphs which draw a detailed view of temporal associations between cancer treatments and toxicities.

The remaining of paper is organized as follows. In Section 2, related works on data mining methods for cancer data and methods for temporal association mining are briefly reviewed. Section 3 describes the clinical dataset used in our study and the problem statement. The main contributions are presented in Section 4, in which a lattice-based algorithm for mining TARs is described. Experimental results are discussed in Sections 5 while conclusions and future work are represented in Section 6.

[1] The terms "item" and "itemset" are used in association rule mining [13]. In this paper, an "item" is equivalent to a "diagnosis code" and an "itemset" is equivalent to a "codeset". We use these terms interchangeably.

2. Related work

In this section, we briefly review recent studies related to our method, namely data mining methods for cancer data and methods for temporal association rule mining.

2.1. Data mining methods for cancer data

Data mining methods applied to cancer diagnosis and treatment have become increasingly important recently. Yang and Chen [17] combined decision tree and association rule mining to find the correlation between the clinical information and the pathology report to support the lung cancer pathologic staging diagnosis. Alwidian et al. [18] developed a method adapted from class association rule mining to predict breast cancer in women. With predefined weights for risk factors, their approach discovered more accurate classification rules than standard class association rule mining approaches. Data mining techniques were also applied to patient medical records to improve the quality of oral cancer care [19,20].

In cancer comorbidity study, Chen and Xu applied association rule mining to analyze the comorbidities in patients with colorectal cancer [21]. Three data mining techniques, namely artificial neural network, logistic regression, and random forest were used to predict the overall survivability in comorbidity of cancers [22]. To discover the temporal associations between cancer treatments and toxicities, one way is to apply the temporal comorbidity analysis method introduced in [8,9]. Although this approach is simple and easily implemented to draw an overview picture of the temporal associations between cancer treatments and toxicities, it has two significant drawbacks. First, it only detects the association between a pair of diagnosis codes (not between two sets of diagnosis codes); thus, it cannot discover the co-occurrence of toxicities. Moreover, since it does not summarize relevant diagnosis codes, the number of rules generated is often very large, with redundant information. Second, it does not utilize the temporal gap between two diagnosis codes well; hence, it is ineffective in revealing the temporal progression of toxicities of cancer treatments.

2.2. Temporal association rule mining

Mining patterns/rules from temporal datasets is one of topics which has been extensively studied in the data mining community. A temporal dataset can be defined in different ways, which is categorized into one of three main groups: (1) each transaction in the dataset is associated with a time-stamp [23,24]; (2) each item in a transaction is associated with a time-interval [25,26]; and (3) each item in a transaction is associated with a time-stamp (note that the same item in two different transactions can be associated with two different time-stamps) [10–12].

The methods which find patterns/rules from temporal datasets in the group 1 always assume that there is a *time-stamp* associated with each transaction. In [23], the authors used calendar schemas to discover calendar-based patterns, where a calendar schema has the form of (year, month, day). From extracted patterns, they generated TARs using an Apriori-based algorithm. In [27], Lin and colleagues presented two tree-based algorithms for mining frequent temporal patterns, which consider not only the supports of patterns but also their weights. Recently, Ghorbani and Abessi [24] adapted the Apriori algorithm to find temporal patterns and TARs. They first introduced the concept of time cubes, which considers time hierarchies in the mining process. They then discovered patterns and rules using two thresholds support and density.

In the group 2, the methods focus on temporal datasets, where each item is associated with a *time-interval* (i.e., it has a *start time* and an *end time*). For example, Winarko and Roddick [28] proposed

the ARMADA algorithm to discover frequent temporal patterns and generate interval-based TARs. In addition, they also enforced the maximum gap constraint to prune insignificant patterns and reduce the number of generated rules. In [29], Patel et al. introduced an Apriori-based algorithm to discover frequent temporal patterns based on a lossless representation. They then used these temporal patterns to construct a classifier. Other works by Moskovitch et al. [26] and Sacchi et al. [25] extracted interval-related patterns represented with the Allen's interval algebra.

This paper focuses on temporal datasets belonging to the group 3, i.e., our approach deals with *time-stamped* events (or items) which are common in a routine cancer data collection. Several methods have been proposed to discover rules from time-stamped items. In [10], the authors developed a method based on Apriori [14] to extract the temporal dependencies between gene expressions. Their approach detected various sizes of the time delay between associated genes and sets of co-regulators for the target genes. With the same topic, Alves et al. [30] reviewed relevant methods of frequent itemset mining and temporal rule mining in gene association analysis. The work by Concaro and colleagues [11] covered rules involving time-stamped events, but they considered only rules with a single consequent (i.e., the right hand side has only one event). In other words, they did not take into account the comorbidities reflected in coded medical records. In [12], the authors applied an Apriori-based method for discovering TARs in a cancer dataset, which had to scan the dataset multiple times, hence leading to a high computational cost. The common process in all such methods includes two phases: (1) mining frequent itemsets from the dataset and (2) generating TARs from discovered frequent itemsets. While the first phase can be achieved using different algorithms such as Apriori [14] or FPGrowth [15], the second phase is always performed using the algorithm introduced in [14], which requires the finding of all proper subsets of each frequent itemset. In case of a large number of frequent itemsets discovered from the dataset, the second phase consumes time, causing the whole mining process of existing methods remarkably inefficient.

Related to time-stamped events, there were several methods which applied sequential pattern mining algorithms to find TARs [31,32]. However, these methods are different from our method in at least two folds. First, these methods deal with *a single long sequence of events* while our method deals with *multiple sequences of events*. Second, these methods require end-users to specify a target event, and it discovers rules whose the right-hand side is this target event (a singleton event). In contrast, our method does not require a specific target event, and it finds general temporal rules in the form of $X \rightarrow (\triangle)Y$, where X and Y are two sets of events and $\triangle$ is a time gap constraint between X and Y. Thus, the problem solved by our method has more computational complexity than finding rules with a target event.

3. Background

In this section, we first introduce the clinical dataset used in our study and then define our problem (i.e., the problem statement).

3.1. Clinical dataset

This study was conducted in a regional cancer center in an Australia tertiary hospital[2]. The cancer center covers a population over 200,000 and provides cancer treatment/palliative care to over 6,000 patients each year. The study cohort consists of 717 patients who received radical radiation treatments between January 2010 and April 2015. The clinical dataset includes diagnosis codes (in the form of International Classification of Diseases (ICD)-10 codes [33]) for each admitted hospital episode from the start of radiation treatment to the end of study period. The procedure codes (in the form of Australian Classification of Health Interventions (ACHI) codes [34]) for radical radiation treatments are also included. We remove the episodes for dialysis and same day chemo treatments since these episodes contain mostly repeated treatment information and reflect little information on toxicities and complications. We also remove diagnosis codes which appear in less than 30 patients. After cleaning data, we obtain 3,364 hospital episodes following the first radiation treatment (on average 4.7 episodes per patient).

From these episodes, we convert the data into a transaction dataset format [35], where each transaction is a patient and items are diagnosis codes. Each diagnosis code in a patient is associated with a time-stamp and the same diagnosis code in two different patients can be associated with two different time-stamps. To illustrate the transaction dataset, we use an example dataset as shown in Table 1.

The characteristics of the transaction dataset is summarized as follows: the number of transactions (or patients) is 717, the number of distinct items (or diagnosis codes) is 1,835, and the minimum, average, and maximum numbers of items in a transaction are 1, 16.21, and 66, respectively. We also show top 10 diagnosis codes w.r.t their frequencies in Table 2.

3.2. Problem statement

We aim to find TARs in the form of $X \rightarrow (\triangle)Y$, where X (left hand side) is a type of cancer treatment and Y (right hand side) is a set of diagnosis codes. Y and X are expected to follow an temporal order, where the occurrence of Y follows the occurrence of X after a time gap $\triangle$ for most patients. Note that although our study is interested in temporal rules where X is a particular type of cancer treatment (i.e., a single diagnosis code), our proposed method is still applicable to discover general temporal rules where both X and Y are two sets of diagnosis codes. Hereafter, we will describe the problem and our method focusing on general temporal rules.

We formally define a temporal association rule and related concepts as follows. Given a set of diagnosis codes (or *codes* for short) $\mathcal{C} = \{c_1, c_2, \ldots, c_M\}$, a *transaction dataset* $\mathcal{D} = \{P_1, P_2, \ldots, P_N\}$ is a set of patients where each patient P_i is a set of distinct codes (i.e., $P_i \subseteq \mathcal{C}$). Each code in a patient is associated with a time-stamp and the same code in two different patients can be associated with two different time-stamps. A *codeset* X is a set of distinct codes (i.e., $X \subseteq \mathcal{C}$).

Definition 1 (Pidset of a code)**.** The *Pidset* of a code c is defined as $Pidset(c) = \{(P_i, t_i) \mid P_i \in \mathcal{D} \wedge c \in P_i\}$, i.e., a list of patients along with time-stamps, who have the code c. For notation purposes, we use $c.P$ to access the list of patients who have c and use $c.P[P_j \in c.P]$ to access the time-stamp associated with c in the patient P_j.

Definition 2 (Pidset of a codeset)**.** The *Pidset* of a codeset X is defined as $Pidset(X) = \{P_i \mid P_i \in \mathcal{D} \wedge X \subseteq P_i\}$, i.e., a list of patients who have the codeset X.

Definition 3 (Relative support [35])**.** The *relative support* of a code c, defined as $\sup(c) = \frac{|Pidset(c)|}{|\mathcal{D}|}$, is the fraction of patients in $\mathcal{D}$, who contain c. Similarly, the *relative support* of a codeset X, defined as $\sup(X) = \frac{|Pidset(X)|}{|\mathcal{D}|}$, is the fraction of patients in $\mathcal{D}$, who contain X.

For the brevity of exposition, we will refer to *relative support* imply as *support* in the rest of this paper.

[2] Ethics approval was obtained from the Hospital and Research Ethics Committee at Barwon Health (number 12/83). The Deakin University has reciprocal ethics authorization with Barwon Health.

Example 1. Consider the example dataset shown in Table 1(a). The code a is contained in three patients P_1, P_2, and P_3. Thus, $Pidset(a) = \{(P_1, t_1), (P_2, t_4), (P_3, t_7)\}$, $a.P = \{P_1, P_2, P_3\}$, and $\sup(a) = \frac{|Pidset(a)|}{|\mathcal{D}|} = 3/3 = 1$ (remind that $|\mathcal{D}|$ is the number of patients in the dataset). To access the time-stamp associated with a in the patient P_2, we use $a.P[P_2] = t_4$. The codeset $X = \{d, e\}$ is contained in two patients P_1 and P_3. Thus, $Pidset(X) = \{P_1, P_3\}$ and $\sup(X) = 2/3 = 0.67$.

Since the support of a code or a codeset is defined in the same way, the concepts provided in the following sections are applicable to both code and codeset.

Definition 4 (Frequent codeset [35])**.** Given a minimum support threshold $\delta \in [0, 1]$, a codeset X is called a *frequent codeset* if $\sup(X) \geq \delta$.

Theorem 1. *(**Apriori property** [14]). Given two codesets $X, Y \subseteq \mathcal{C}$, if $X \subseteq Y$, then $\sup(X) \geq \sup(Y)$. In other words, if Y is frequent (i.e., $\sup(Y) \geq \delta$), then any subset X of Y is also frequent since $\sup(X) \geq \sup(Y) \geq \delta$.*

Definition 5 (Temporal association rule)**.** A *temporal association rule* R has the form of $X \rightarrow (\triangle)Y$, where X and Y are two frequent codesets ($X \cap Y = \emptyset$) and $\triangle$ is a time gap constraint between X and Y. The rule R indicates the presence of the codeset Y likely follows the presence of the codeset X after a time frame $\triangle$ with a certain probability.

To evaluate the quality of temporal rules, we propose two measures, namely the *direction support* and the *direction confidence*, which are presented next.

Definition 6 (Direction support)**.** The *direction support* of a temporal association rule R: $X \rightarrow (\triangle)Y$ is defined as:

$$\text{dirsup}(R) = \frac{|\{P_i \mid P_i \in Pidset(X \cup Y) \wedge \min(Y, P_i) - \max(X, P_i) \geq \triangle\}|}{|\mathcal{D}|}, \quad (1)$$

where P_i is a patient who has both X and Y, $\min(Y, P_i) = \min_{\forall c \in Y}\{c.P[P_i]\}$, and $\max(X, P_i) = \max_{\forall c \in X}(c.P[P_i])$.

Different from the *support* of an association rule [13], which only shows the co-occurrence between X and Y, the direction support furthermore captures the temporal dependence between X and Y. In the context of our problem, the direction support can capture the comorbidities which are temporal dependent while appearing at different stages of a cancer treatment process.

Definition 7 (Direction confidence)**.** The *direction confidence* of a temporal association rule R: $X \rightarrow (\triangle)Y$ is defined as:

$$\text{dirsup}(R) = \frac{\text{dirsup}(R)}{\sup(X)} \quad (2)$$

Definition 8 (Valid temporal association rule)**.** Given a minimum direction confidence threshold $\lambda \in [0, 1]$, a rule R: $X \rightarrow (\triangle)Y$ is called a *valid temporal association rule* if $\text{dirconf}(R) \geq \lambda$.

Example 2. Consider the sample dataset shown in Table 1(a). Let $\triangle = 30$days. The direction support and the direction confidence of the rule R: $a \rightarrow \{d, e\}$ are computed as follows. There are two patients who contain both a and $\{d, e\}$, namely P_1 and P_3. Given P_1, $\min(\{d, e\}, P_1) - \max(a, P_1) = \min\{d.P[P_1], e.P[P_1]\} - \max\{a.P[P_1]\} = \min\{t_3, t_2\} - \max\{t_1\} = t_2 - t_1 \geq \triangle$. Given P_3, $\min(\{d, e\}, P_3) - \max(a, P_3) = \min\{d.P[P_3], e.P[P_3]\} - \max\{a.P[P_3]\} = \min\{t_6, t_8\} - \max\{t_7\} = t_6 - t_7 < \triangle$. As a result, $\text{dirsup}(R) = 1/3 = 0.33$ and $\text{dirconf}(R) = \frac{\text{dirsup}(R)}{\sup(a)} = 0.33/1 = 0.33$ since $\sup(a) = 1$ as computed in Example 1.

Problem statement. Given a transaction dataset $\mathcal{D}$, a minimum support threshold $\delta \in [0, 1]$, a minimum direction confidence threshold $\lambda \in [0, 1]$, and a time frame $\triangle$ (in days), our goal is to find a set of frequent codesets $\mathcal{F}$ from $\mathcal{D}$, i.e., $\mathcal{F} = \{X \subseteq \mathcal{C} \mid \sup(X) \geq \delta\}$ and then find a set of valid TARs $\mathcal{R}$ from $\mathcal{F}$ such that $\mathcal{R} = \{R : X \rightarrow (\triangle)Y \mid X, Y \in \mathcal{F} \wedge X \cap Y = \emptyset \wedge \text{dirconf}(R) \geq \lambda\}$.

4. Framework

In this section, we introduce our proposed method for mining TARs, named **LTARM**, which stands for *Lattice-based Temporal Association Rule Mining*. **LTARM** has two main phases: (1) building the lattice structure of frequent codesets and (2) mining TARs from the lattice, which are presented next.

4.1. Building the lattice structure of frequent codesets

We first define the lattice structure used for storing frequent codesets and encoding the parent-child relationships among them. We then develop an algorithm to build this lattice structure.

4.1.1. Lattice structure

Definition 9 (Node)**.** A node X in the lattice is a tuple in the form:

$$X = (codeset, traverse, children),$$

where *codeset* is a frequent codeset (see Definition 4), *traverse* is a flag to indicate whether this node has already generated a rule, and *children* is a list of *child* nodes. For notation purposes, the frequent codeset, the flag, and the list of child nodes of a node X are denoted as *X.codeset, X.traverse*, and *X.children*, respectively.

Definition 10 (Child node)**.** Given two nodes X and Y, Y is a *child* node of X if the frequent codeset contained in X is an immediate subset of the frequent codeset contained in Y, that is, $X.codeset \subset Y.codeset$ and $|X.codeset| = |Y.codeset| - 1$.

Definition 11 (Lattice structure)**.** The *lattice structure* is a directed graph where each node is defined in Definition 9 and an edge connects from a node X to a node Y if Y is a child node of X. In the lattice, there is a special node, called the *root* node. The root node contains an empty frequent codeset which is considered as an immediate subset of a frequent 1-codeset (i.e., a frequent codeset contains only one single code).

Example 3. In the lattice as shown in Fig. 1, *root* is the root node which has four child nodes *a, b, d*, and *e* (we name a node by the frequent codeset contained in that node). Similarly, the node *b* has two child nodes *ab* and *bd*.

4.1.2. Algorithm

Our algorithm, named **BUILD-LATTICE**, for constructing the lattice structure is shown in Algorithm 1 . It first creates the root node *root* whose child nodes are frequent 1-codesets (lines 2–6). It then calls the procedure EXTEND-LATTICE to build the lattice structure (line 7). This procedure combines each node $L_x \in root.children$ with another node $L_y \in root.children$ to generate a candidate child node O (line 14). It then computes two elements *codeset* and *Pidset* for O (lines 15–16). Based on the information of *Pidset*, it computes the support of O and checks whether this support satisfies δ (line 17). If true, O is a frequent codeset and its parent-child relationships with its parent nodes and its child nodes are updated (lines 18–19). O is then added to $\mathcal{N}_x$ and the lattice $\mathcal{L}$ (lines 20–21). The procedure EXTEND-LATTICE is recursively called with the new input parameter $\mathcal{N}_x$ (line 24).

Algorithm 1: BUILD-LATTICE($\mathcal{D}$, δ) algorithm.

Input: $\mathcal{D} = \{P_1, \ldots, P_N\}$: a transaction dataset, δ: a minimum support threshold
Output: $\mathcal{L}$: the lattice structure containing all frequentcodesets in $\mathcal{D}$

```
1  begin
2      scan D to find the set of all frequent 1-codesets, F1;
3      let root = (∅, false, {}) be the root node;
4      foreach codeset ∈ F1 do
5          add a node X = (codeset, false, {}) to root.children;
6      end
7      EXTEND-LATTICE(root.children, δ);
8  end
9  EXTEND-LATTICE(N, δ)
10 begin
11     foreach Lx ∈ N do
12         Nx = {};
13         foreach Ly>x ∈ N do
14             create a node O = (codeset, false, {});
15             O.codeset = Lx.codeset ∪ Ly.codeset;
16             Pidset(O.codeset) =
               Pidset(Lx.codeset) ∩ Pidset(Ly.codeset);
17             if sup(O) = |Pidset(O.codeset)|/|D| ≥ δ then // use
               Theorem]1
18                 add O to Lx.children and Ly.children;
19                 UPDATE-LATTICE(Lx, O);
20                 Nx = Nx ∪ O;
21                 add O to the lattice L;
22             end
23         end
24         EXTEND-LATTICE(Nx, δ);
25     end
26 end
27 UPDATE-LATTICE(Lx, O)
28 begin
29     foreach X ∈ Lx.children do
30         foreach Y ∈ X.children do
31             if O ⊂ Y then add Y to O.children;
32         end
33     end
34 end
```

4.1.3. Complexity analysis

Our algorithm **BUILD-LATTICE** requires one scan over the dataset $\mathcal{D}$ to find all frequent singleton codesets (line 2). This task has $\mathcal{O}(|\mathcal{D}|)$. Let $|\mathcal{F}_1| = |root.children| = f$. Frequent codesets are discovered by the procedure EXTEND-LATTICE, requiring $\mathcal{O}(f^2)$. The procedure UPDATE-LATTICE consists of two nested "for" loops (lines 29–30), which requires a quadratic complexity $\mathcal{O}(|L_x.children|) \times |X.children|)$. Since the number of patients $|\mathcal{D}|$ and the number of frequent 1-codesets f are often much larger than $|L_x.children|$ and $|X.children|$, the complexity of our **BUILD-LATTICE** is determined by $\mathcal{O}(|\mathcal{D}| + f^2)$. It will be $\mathcal{O}(|\mathcal{D}|)$ (when $f \ll |\mathcal{D}|$) in the average case and $\mathcal{O}(|\mathcal{D}|^2)$ (when $f \gg |\mathcal{D}|$) in the worst case.

4.1.4. Illustrative example

To demonstrate the basic steps of **BUILD-LATTICE**, we apply it to the sample dataset in Table 1 with $\delta = 0.6$. The root node (*root*) has four child nodes which are frequent 1-codesets *a, b, d,* and *e*. Since code *c* only appears in P_1 ($\text{sup}(c) = 1/3 < \delta$), it is not a child node of *root*. In the procedure EXTEND-LATTICE, each node in *root.children* is combined with another node to generate a new candidate child node. Let consider node *a* as an example. It combines with node *b* to generate node *ab*. Since $\text{sup}(ab) = 2/3 > \delta$, this node is added to the lattice as a child node of *a* and *b*. Similarly, node *ad* is a child node of *a* and *d* while node *ae* is a child node of *a* and *e*. Because node *a* has been already combined with all other nodes in *root.children*, EXTEND-LATTICE is called with the new input parameter $\mathcal{N}_x = \{ab, ad, ae\}$. Following the same process, node *ab* is combined with node *ad* to generate node *abd*. Since $\text{sup}(abd) = 2/3 > \delta$, this node is added to the lattice as a child node of *ab* and *ad*. Node *ab* is then combined with node *ae* to generate node *abe*. However, $\text{sup}(abe) = 1/3 < \delta$, this node is not frequent and is not added to the lattice. After the algorithm **BUILD-LATTICE** finishes, we obtain the lattice as shown in Fig. 1.

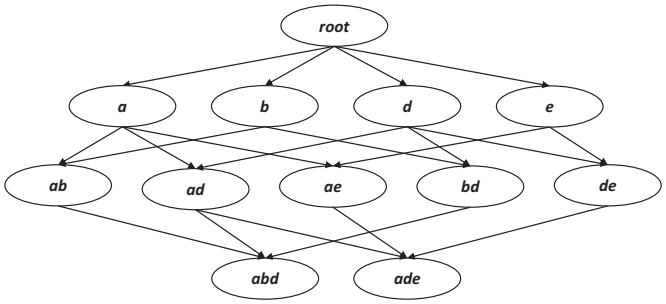


Fig. 1. Lattice structure built from the dataset in Table 1(a). The name of a node is also the frequent codeset contained in that node. For example, the nodes *a, ab,* and *abd* contain the frequent codesets {*a*}, {*a, b*}, and {*a, b, d*}, respectively.

4.2. Mining TARs from the lattice structure

We first propose two theorems which help to prune invalid rules quickly. We then develop an algorithm to discover TARs from the lattice of frequent codesets.

4.2.1. Theorems

Theorem 2. *Given a rule* R: $X \rightarrow (\triangle)Y$, *let* $\text{sup}(R) = \text{sup}(X \cup Y)$ *be the support of* R *and* $conf(R) = \frac{\text{sup}(R)}{\text{sup}(X)}$ *be the confidence of* R *[35]. If* $conf(R) < \lambda$, *then* $dirconf(R) < \lambda$.

Proof. Regarding Definition 6, $\text{dirsup}(R) = \frac{|\{P_i | P_i \in Pidset(X \cup Y) \wedge \min(Y, P_i) - \max(X, P_i) \geq \triangle\}|}{|\mathcal{D}|}$. Regarding Definition 3, $\text{sup}(X \cup Y) = \frac{|Pidset(X \cup Y)|}{|\mathcal{D}|} = \text{sup}(R)$. We can conclude that $\text{dirsup}(R) \leq \text{sup}(R)$. This implies that $\text{dirconf}(R) = \frac{\text{dirsup}(R)}{\text{sup}(X)} \leq \text{conf}(R) = \frac{\text{sup}(R)}{\text{sup}(X)}$. Thus, if $\text{conf}(R) < \lambda$, then $\text{dirconf}(R) < \lambda$. □

Theorem 2 helps us to prune invalid rules quickly without computing their direction supports and direction confidences. Recall that the way to compute the direction support of a rule R: $X \rightarrow (\triangle)Y$ requires two steps: (1) finding the set of patients who have $X \cup Y$ (i.e., $Pideset(X \cup Y)$), and (2) for each patient $P_i \in Pidset(X \cup Y)$, checking whether the condition $\min(Y, P_i) - \max(X, P_i) \geq \triangle$ satisfies. The first step to compute $Pidset(X \cup Y)$ is very fast whereas the second step consumes time since it depends on the number of patients in $Pidset(X \cup Y)$ and the number of singleton codes in X and Y. From $Pidset(X \cup Y)$, we compute $\text{sup}(R)$ and $\text{conf}(R)$, and if $\text{conf}(R) < \lambda$, then we do not need to compute $\text{dirsup}(R)$ and $\text{dirconf}(R)$. For example, consider the rule R: $a \rightarrow (\triangle)\{d, e\}$ with $\triangle = 30$ days, as mentioned in Example 2. There are two patients who have {*a, d, e*} and three patients who have *a*. Hence, we can easily compute $\text{sup}(R) = \text{sup}(\{a, d, e\}) = \frac{|Pidset(\{a,d,e\})|}{|\mathcal{D}|} = 2/3 = 0.67$ and $\text{conf}(R) = \frac{\text{sup}(R)}{\text{sup}(a)} = 0.67/1 = 0.67$ since $\text{sup}(a) = \frac{|Pidset(a)|}{|\mathcal{D}|} = 3/3 = 1$. Assume

Table 1
An illustration of the clinical dataset. Table (a) shows three patients along with their diagnosis codes. Each diagnosis code in a patient is a type of cancer treatment or a toxicity and it is associated with a time-stamp. Tables (b) and (c) describe the meaning of diagnosis codes and time-stamps, respectively.

(a) Patients with diagnosis codes.

Patient	Diagnosis codes
P_1	$\{a(t_1), b(t_1), c(t_1), e(t_2), d(t_3)\}$
P_2	$\{a(t_4), b(t_4), d(t_5)\}$
P_3	$\{d(t_6), a(t_7), e(t_8)\}$

(b) Diagnosis codes.

Code	Description
a	1526900: Radiation treatment
b	R11: Nausea
c	E86: Volume depletion
d	K590: Constipation
e	E876: Hypokalaemia

(c) Time stamps.

Time stamp	Date
t_1	2010/08/20
t_2	2013/07/17
t_3	2015/03/12
t_4	2013/06/21
t_5	2014/04/07
t_6	2012/01/12
t_7	2012/11/14
t_8	2015/02/27

Table 2
Top 10 diagnosis codes w.r.t their frequencies (i.e., the number of occurrences).

No.	Code	Description	Frequency
1	1526900	Radiation treatment megavoltage $\geq$ 2 fields	187
2	M80703	Squamous cell carcinoma NOS	177
3	E86	Volume depletion	175
4	M81403	Adenocarcinoma NOS	175
5	R11	Nausea and vomiting	161
6	K590	Constipation	138
7	E876	Hypokalaemia	135
8	D649	Anaemia unspecified	129
9	N179	Acute kidney failure unspecified	124
10	E834	Disorders of magnesium metabolism	124

$\lambda = 0.7$. Since $\text{conf}(R) < \lambda$, we conclude that $\text{dirconf}(R) < \lambda$ (indeed, $\text{dirconf}(R) = 0.33 < \lambda$) and R is not a valid rule.

Theorem 3. *Given two rules* $R1$: $X \rightarrow (\triangle)Y$ *and* $R2$: $X \rightarrow (\triangle)Z$, *and* $Y \subset Z$. *If* $\text{dirconf}(R1) < \lambda$, *then* $\text{dirconf}(R2) < \lambda$.

Proof. Regarding Definition 7, $\text{dirconf}(R1) = \frac{\text{dirsup}(R1)}{\text{sup}(X)}$ and $\text{dirconf}(R2) = \frac{\text{dirsup}(R2)}{\text{sup}(X)}$. To prove Theorem 3, we need to prove $\text{dirconf}(R2) \leq \text{dirconf}(R1)$, in other words, $\text{dirsup}(R2) \leq \text{dirsup}(R1)$.

To prove $\text{dirsup}(R2) \leq \text{dirsup}(R1)$, we need to prove this inequality: $|\{P_i \mid P_i \in Pidset(X \cup Z) \wedge \min(Z, P_i) - \max(X, P_i) \geq \triangle\}| \leq |\{P_i \mid P_i \in Pidset(X \cup Y) \wedge \min(Y, P_i) - \max(X, P_i) \geq \triangle\}|$ (see Definition 6). Since $Y \subset Z$, we have $Pidset(X \cup Z) \subseteq Pidset(X \cup Y)$. To prove the inequality, we need to prove $\forall P_i \in Pidset(X \cup Z)$, if $\min(Z, P_i) - \max(X, P_i) \geq \triangle$, then $\min(Y, P_i) - \max(X, P_i) \geq \triangle$.

Assume that $\exists P_* \in Pidset(X \cup Z)$ such that $\min(Z, P_*) - \max(X, P_*) \geq \triangle$ but $\min(Y, P_*) - \max(X, P_*) < \triangle$. Without loss of generality, let $X = \{a\}$, $Y = \{d\}$, and $Z = \{de\}$. Regarding Definition 6, we have:

$$\begin{aligned}\min(Z, P_*) - \max(X, P_*) &= \min\{d.P[P_*], e.P[P_*]\} - \max\{a.P[P_*]\} \\ &= \min\{d.P[P_*], e.P[P_*]\} - a.P[P_*] \\ \min(Y, P_*) - \max(X, P_*) &= \min\{d.P[P_*]\} - \max\{a.P[P_*]\} \\ &= d.P[P_*] - a.P[P_*]\end{aligned}$$

There are two cases:

(1) $d.P[P_*] \leq e.P[P_*]$: in this case, $\min(Z, P_*) - \max(X, P_*) = d.P[P_*] - a.P[P_*] = \min(Y, P_*) - \max(X, P_*) \geq \triangle$. This conflicts with the assumption.

(2) $d.P[P_*] > e.P[P_*]$: in this case, $\min(Z, P_*) - \max(X, P_*) = e.P[P_*] - a.P[P_*] \geq \triangle$. Since $d.P[P_*] > e.P[P_*]$, $d.P[P_*] - a.P[P_*] = \min(Y, P_*) - \max(X, P_*) \geq \triangle$. This conflicts with the assumption.

Consequently, we has proved that $\forall P_i \in Pidset(X \cup Z)$, if $\min(Z, P_i) - \max(X, P_i) \geq \triangle$, then $\min(Y, P_i) - \max(X, P_i) \geq \triangle$. In other words, Theorem 3 has been proved. □

Theorem 3 implies that if the direction confidence of a rule is less than λ, then no items should be appended to its consequence to generate candidate rules since these rules are not valid. For example, let $\lambda = 0.7$, $\triangle = 30$ days, and consider the rule $R1$: $a \rightarrow (\triangle)d$ generated from Table 1. There are three patients (P_1, P_2, and P_3) who have $\{a, d\}$, in which only two patients (P_1 and P_2) who satisfy $\min(d, P_i) - \max(a, P_i) \geq \triangle$. Thus, $\text{dirconf}(R1) = 0.67$. Since $\text{dirconf}(R1) < \triangle$, we do need to consider the rule $R2$: $a \rightarrow (\triangle)\{d, e\}$ which has $\text{dirconf}(R2) = 0.33 < \triangle$.

4.2.2. Algorithm

Our algorithm, named **TRAVERSE-LATTICE**, for discovering valid TARs from the lattice structure is shown in Algorithm 2 . It first traverses all child nodes of the root node (line 2). For each child node, it checks whether this node has generated a rule (line 3). If not, a rule will be generated by the procedure GENERATE-RULE (line 4). The goal of GENERATE-RULE is that given a frequent codeset X, it generates candidate rules in the form of R: $X \rightarrow (\triangle)Z \backslash X$, where $X \subset Z$ (i.e., node Z is a child node of node X). Since the lattice structure has already encoded the parent-child relationship between X and Z, we can traverse the lattice to generate these candidate rules very quickly. In addition, GENERATE-RULE employs two Theorems 2 and 3 (lines 20 and 23) to prune invalid candidate rules. As a result, our mining process of TARs is very efficient. In contrast, existing algorithms for mining TARs have to find all proper subsets of Z, which can be explosive up to $2^k - 2$ if Z has k codes, to generate candidate rules. For each candidate rule, they always compute the direction support and the direction confidence, which is a high computational task.

4.2.3. Complexity analysis

Our algorithm **TRAVERSE-LATTICE** traverses all child nodes of the root node (line 2), which requires $\mathcal{O}(f)$ (recall $|root.children| = f$). The procedure GENERATE-RULE consists of two separate loops (lines 13–15 and lines 16–30), requiring $\mathcal{O}(|L_c.children| + |\mathcal{Q}|)$. The complexity of our **TRAVERSE-LATTICE** is $\mathcal{O}(f + |L_c.children| + |\mathcal{Q}|)$. Let g be the average number of child nodes of every nodes. The complexity can be simplified to $\mathcal{O}(f \times g)$. It will be $\mathcal{O}(f)$ (when $g \ll f$) in the best case and $\mathcal{O}(g^2)$ (when $f \ll g$) in the average case.

4.2.4. Illustrative example

To demonstrate the process of **TRAVERSE-LATTICE**, we apply it to the lattice structure as shown in Fig. 1, with $\lambda = 0.7$ and $\triangle = 30$ days. The algorithm first scans all child nodes (a, b, d, and e) of the root node. Let consider node a as an example. The procedure GENERATE-RULE will find valid TARs where their antecedent is a. First, the child nodes of a, which are ab, ad, and ae, are added to the queue $\mathcal{Q}$. Then, the first candidate rule $R1$: $a \rightarrow (\triangle)b$ is generated with $\text{conf}(R1) = 0.67$. Since $\text{conf}(R1) < \lambda$, we do not need to compute its direction confidence regarding Theorem 2 and $R1$ is an invalid rule. The second candidate rule $R2$: $a \rightarrow (\triangle)d$ is generated with $\text{conf}(R2) = 1 \geq \lambda$. We compute $\text{dirsup}(R2) = 0.67$ and $\text{dirconf}(R2) = 0.67$. Since $\text{dirconf}(R2) < \lambda$, $R2$ is an invalid rule and

Algorithm 2: TRAVERSE-LATTICE(*root*, λ, △) algorithm.

Input: *root*: the root node of the lattice, λ: a minimumdirection confidence threshold, △: a time frame
Output: $\mathcal{R}$: the set of valid temporal association rules

```
1  begin
2      foreach L_c ∈ root.children do
3          if L_c.traverse = false then
4              GENERATE-RULE(L_c, λ, △);
5              L_c.traverse = true;
6              TRAVERSE_LATTICE(L_c, λ, △);
7          end
8      end
9  end
10 GENERATE-RULE(L_c, λ, △)
11 begin
12     Queue Q = ∅;
13     foreach L_s ∈ L_c.children do
14         add L_s to Q;
15     end
16     while Q ≠ ∅ do
17         get a node L from Q;
18         set X = L_c.codeset; Y = L.codeset \ L_c.codeset;
           X ∪ Y = L.codeset;
19         create a rule R : X → (△)Y;
20         if conf(R) = sup(X∪Y)/sup(X) ≥ λ then // use Theorem 2
21             compute dirsup(R) =
               |{P_i|P_i∈Pidset(X∪Y)∧min(Y,P_i)−max(X,P_i)≥△}| / |D|;
22             compute dirconf(R) = dirsup(R)/sup(X);
23             if dirconf(R) ≥ λ then // use Theorem 3
24                 add R to R;
25                 foreach L_k ∈ L.children do
26                     if L_k ∉ Q then add L_k to Q;
27                 end
28             end
29         end
30     end
31 end
```

Table 3
Valid TARs extracted from the lattice structure in Fig. 1, with λ = 0.7 and △ = 30 days.

Rule	Direction support	Direction confidence
$R1$: $\{a, b\} \rightarrow (\triangle)d$	0.67	1.00
$R2$: $b \rightarrow (\triangle)d$	0.67	1.00

we do not need to consider other candidate rules like $a \rightarrow (\triangle)\{b, d\}$ and $a \rightarrow (\triangle)\{d, e\}$ because they are also invalid rules regarding Theorem 3. Similarly, the third candidate rule $R3$: $a \rightarrow (\triangle)e$ with $\text{conf}(R3) = 0.67 < \lambda$ and this rule is invalid.

After processing node a to discover valid TARs with the rule antecedent a, the procedure **TRAVERSE-LATTICE** is recursively applied to the child nodes (ab, ad, and ae) of node a to find valid TARs in the form of R: $\{a, b\} \rightarrow (\triangle)$xxx, R: $\{a, d\} \rightarrow (\triangle)$xxx, and R: $\{a, e\} \rightarrow (\triangle)$xxx, respectively. Finally, the algorithm continues with the remaining child nodes of the root node (i.e., nodes b, d, and e) to search for other valid TARs. All valid TARs mined from the lattice structure shown in Figure 1, are reported in Table 3.

5. Experiments

In this section, we conduct comprehensive experiments on the clinical dataset (see Section 3.1) to qualitatively and quantitatively evaluate the performance of our proposed method – **LTARM**. Our goals are:

1. To demonstrate our method's capability of capturing the temporal progression and the co-occurrence information of toxicity patterns, which is superior than the temporal comorbidity analysis method.
2. To demonstrate the efficiency of our method in terms of runtime, which is better than two existing methods for temporal association rule mining.

5.1. Discovery of temporal toxicity patterns

The first experiment addresses the task of discovering the temporal relationships between cancer treatments and toxicities, wherein we visualize and compare the discovered toxicity patterns of our method with those of the temporal comorbidity analysis method.

5.1.1. Baseline

The main goal of the temporal comorbidity analysis method [8] is to find the frequently temporal relations among comorbidities in longitudinal hospital discharge records. The comorbidities take the form of pairs of diagnosis codes in which one code follows the other temporally in different episodes of a patient. To find such pairs, the method takes two steps. In the first step, all frequent pairs are discovered, ignoring the temporal order between the first code and the second code. This is achieved through the Chi-square test between each pair of codes. There are $(\frac{M^2-M}{2})$ possible pairs among codes, where M is the number of distinct codes. For each pair of two codes a and b, we compute an 2×2 contingency table and a Chi-square value. The contingency table includes: (1) The number of patients who have both code a and code b; (2) The number of patients who have code a but not code b; (3) The number of patients who have code b but not code a; and (4) The number of patients who have neither code a nor code b. The Chi-square value χ^2 is calculated from the contingency table, as follows:

$$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{2}\frac{O_{ij} - E_{ij}}{E_{ij}},$$

where O_{ij} is an observed value at the intersection of the row i and the column j, and $E_{ij} = \frac{O_{i.} \times O_{.j}}{N}$, where $O_{i.}$ is the total of the values in the row i, O_j is the total of the values in the column j, and $N = |\mathcal{D}|$ is the number of patients.

We then compute the p-value based on the Chi-square value χ^2 and the degree of freedom $df = 1$ by looking up an χ^2 distribution table. If the p-value is less than the significance level $\alpha = 0.05$, we say the correlation between a and b is statistically significant [36]. After finishing the first step, the pairs with a strong enough correlation are passed to the second step. In the second step, the correlated pairs are further filtered to retain only those showing significant temporal orders. This is achieved through the Binomial test, that is, we test whether one code precedes another code. Given two codes a and b, we count the number of times code a occurring △ (days) before code b, and vice versa in all patients who contain both codes a and b, using only the initial instance of each code. The counts for each code pair are then compared using an exact Binomial test with a hypothesized probability of success = 0.5 and confidence = 0.95. The direction between two codes is identified by the code which occurs first more often and the magnitude is

represented by the p-value obtained from the Binomial test. Further details can be found in [8].

When there is an outcome label, it would be an interesting idea to leverage *supervised* methods to generate rules. In our application, however, there is no natural *label* for each transaction, and our main objective is to discover the temporal relations among diagnosis codes in an *unsupervised* learning setting. Therefore, *supervised* learning methods such as Random Forest [37] and Forest CERN [38], unfortunately, cannot be applied to our problem.

5.1.2. Parameter settings

Our method **LTARM** has three parameters: the minimum support threshold δ for discovering frequent codesets, the minimum direction confidence threshold λ for mining valid TARs, and the time frame $\triangle$.

In our cancer dataset, the number of codes is larger than the number of patients (1,835 vs 717). Therefore, frequent 1-codesets have low supports, where the highest support is 0.26 for the code "1526900: Radiation treatment". If we set the minimum support threshold too high (e.g., $\delta = 0.9$ or $\delta = 1.0$), then none frequent codesets could be found. Consequently, we set $\delta = 0.05$ to be able to discover frequent codesets while it is still large enough to represent the patient population, and we set $\lambda = 0.1$. For the $\triangle$ values used in **LTARM** and the temporal comorbidity analysis baseline, we use three different time frames: 1 week (7 days), 1 month (30 days), and 6 months (180 days). These three durations are chosen to cover acute, short- and medium-term toxicities.

5.1.3. Visualization of temporal associations

We believe that the interestingness of discovered rules should ultimately be determined by the domain experts. Therefore, it is insufficient to provide just a list of rules ordered by some metrics (e.g., the direction support and the direction confidence). The rules should be displayed in an interactive network graph to allow active queries by the oncologists. In particular, we visualize the rules by interactive network graphs using Gephi [39] (an interactive visualization tool) with version 0.9.2, where the oncologists can hover over a node to see all related edges highlighted, and even search for rules based on some conditions.

Visualization of temporal associations discovered by the temporal comorbidity analysis method. We visualize the temporal associations using a directed graph where each edge represents a discovered code pairs. This is the same as the visualization method mentioned in [8]. The interpretation of the graph is quite simple because each edge represents only one direction from one code to another code. Nonetheless, we cannot identify the combined temporal relations which are regarded an important characteristic of radiation treatments and toxicities in such kind of graphs.

Visualization of temporal association rules discovered by our LTARM. For TARs, we use the graph visualization method introduced in [40]. A graph is constructed from the temporal rules discovered from the dataset. For each rule, a meta node T is introduced so that each code in the left hand side of the rule represented by a node has an edge pointing towards T and each code in the right hand side of the rule which is also denoted by a node has an edge from T. For example, consider an example graph shown in Fig. 2. This graph represents the rule "*1526900: Radiation treatment → (30d) R11: Nausea and E86: Volume depletion*". The nodes (with labels) represent diagnosis codes in the left hand side and the right hand side of the rule. The meta node T is denoted by a large blue node (without a label) in the middle of the graph. Its size and color are set based on the direction support and direction confidence of the rule. Here, we set the minimum size and the maximum size of a meta node to 1 and 5, corresponding to the lowest and the highest direction supports found in the set of generated rules. Similarly, we use the default color palette in Gephi to set the color for a meta node, where the orange color corresponds to the lowest direction confidence and the violet color corresponds to the highest direction confidence.

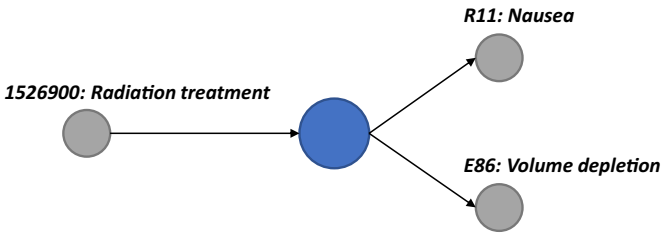


Fig. 2. An example graph for temporal association rule visualization. The graph represents the rule "*1526900: Radiation treatment → (30d) R11: Nausea and E86: Volume depletion*".

5.1.4. Result and discussion

As all 717 patients received radical radiation treatments, we are particularly interested in the toxicities following a radiation treatment. Thus, we have a close look at the subset of rules where the rule antecedent is "*1526900: Radiation treatment megavoltage >= 2 fields dual modality linear accelerator*" (or "*1526900: Radiation treatment*" for short). There are two codes for radiation treatments, namely *1526900* and *1525400*, but most patients received *1526900* for their treatment. The code *1525400* represents a single-field radiation, which is a very rare treatment regiment. In our dataset, only 24 patients received this regiment. The data are insufficient to support generalizable rules.

Toxicity patterns occurring after one week of radiation treatment. Fig. 3(a) shows the temporal associations obtained by the temporal comorbidity analysis baseline while Fig. 3(b) presents the TARs discovered by our method **LTARM**.

Fig. 3(a) confirms common toxicities/complications following a radiation treatment, including: 1) "*D70: Agranulocytosis*", 2) "*E876: Hypokalaemia*", 3) "*R11: Nausea*", 4) "*E834: Magnesium decificiency*", 5) "*R509: Fever*", 6) "*K590: Constipation*", and 7) *"E86: Volume depletion"* (or "*Dehydration*").

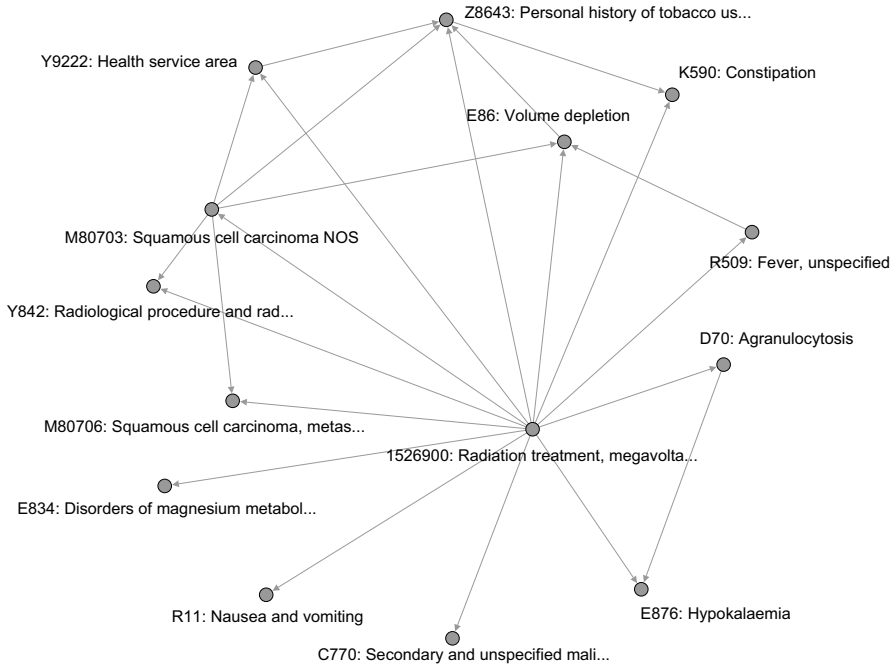
Fig. 3(b) contains most of these complications and some others. One advantage of our results is that TARs can show the complications which co-occur. In particular, "*E876: Hypokalaemia*" and "*E834: Disorders of magnesium metabolism*" are shown to co-occur following a week's radiation treatment.

Toxicity patterns occurring after one month of radiation treatment. Fig. 4(a) and (b) represent the temporal relationships between "*1526900: Radiation treatment*"and other diagnosis codes generated by the temporal comorbidity analysis baseline and our temporal association rule based method, respectively.

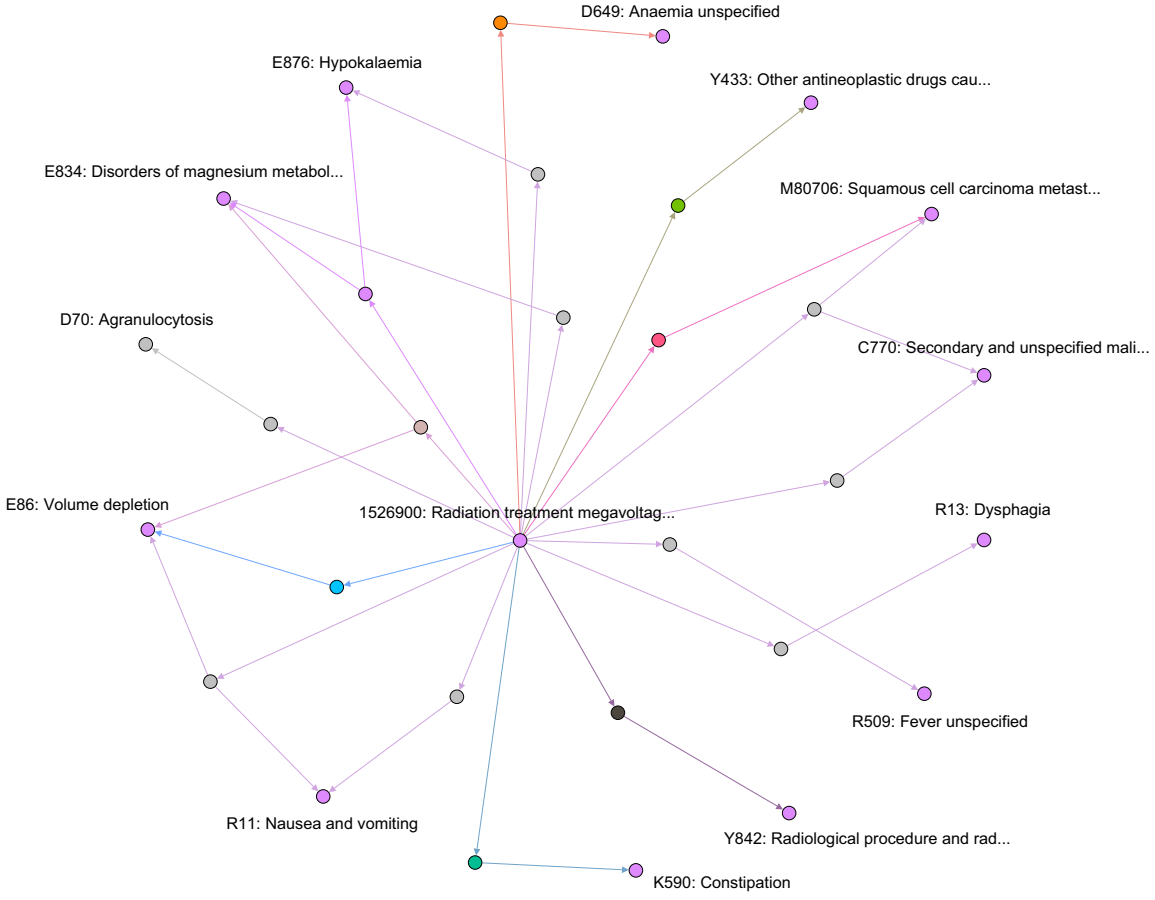
We see that two methods return consistent results, although the additional representation power allows the TARs to capture co-occurrence of toxicities between "*E86: Volume depletion*" and "*R11: Nausea and vomiting*".

Toxicity patterns occurring after six months of radiation treatment. Fig. 5 shows the temporal relationships between "*1526900: Radiation treatment*" and toxicities after six months. Fig. 5(a) shows the temporal associations generated by the temporal comorbidity analysis baseline while Fig. 5(b) represents the TARs found by our method **LTARM**.

Again, we can see different effects of the gap constraints on two methods: the temporal associations of the baseline are still similar to those found after one month whereas the TARs of our method are simpler. This demonstrates that our approach is more effective than the temporal comorbidity analysis baseline in revealing the progressive toxicity patterns of radiation treatments under

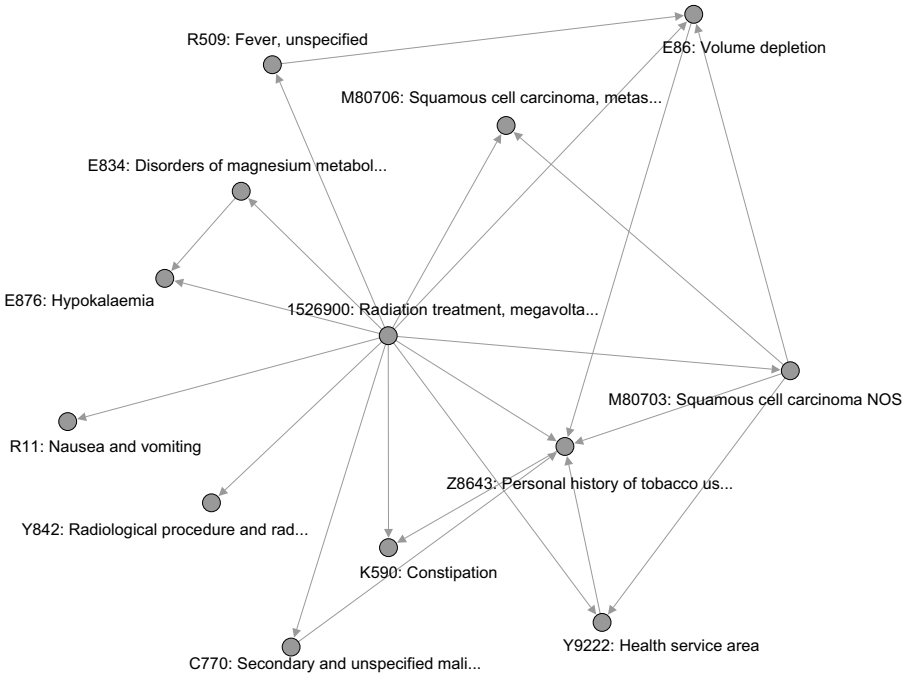


(a) Temporal associations between "*1526900: Radiation treatment*" and other codes, generated by the temporal comorbidity analysis baseline.

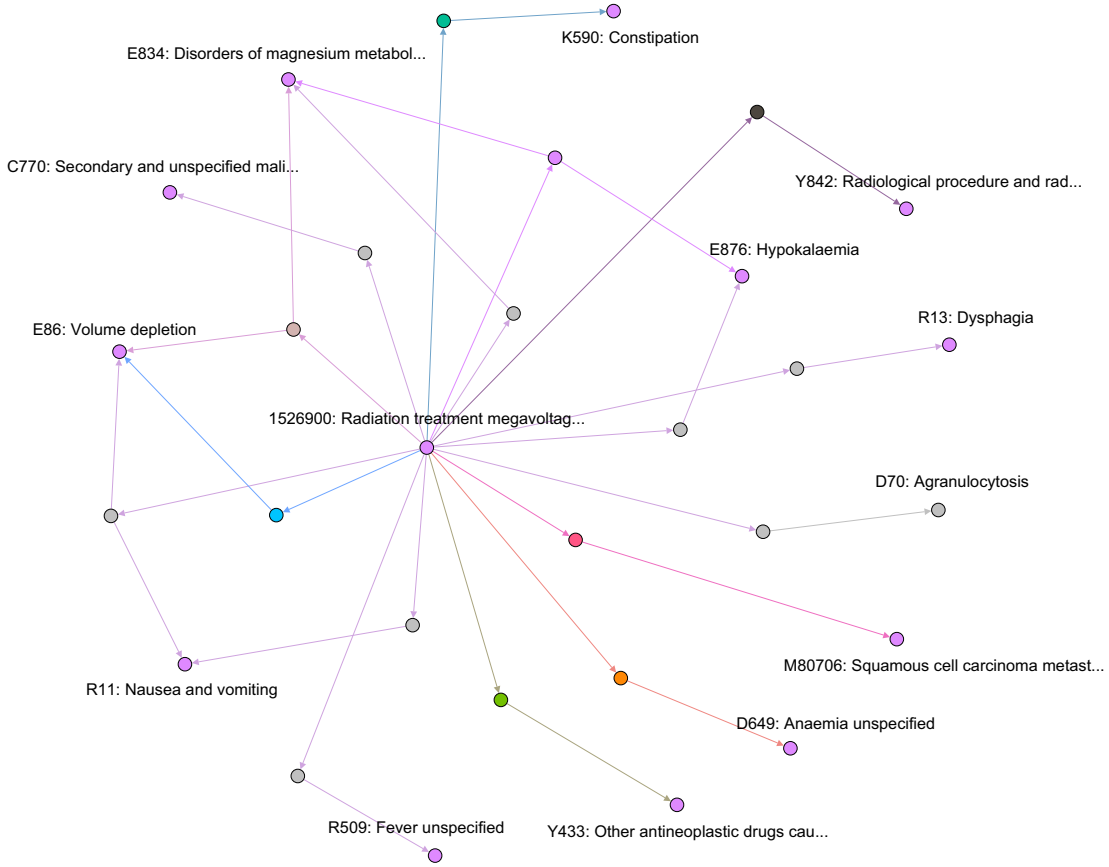


(b) Temporal association rules where the rule antecedent is "*1526900: Radiation treatment*", generated by our method **LTARM**.

Fig. 3. Network graphs showing toxicities/complications which occur after *one week* of radiation treatment. The temporal comorbidity analysis baseline produces only pairwise relations (a). Our method **LTARM** produces TARs represented by additional meta nodes (b).

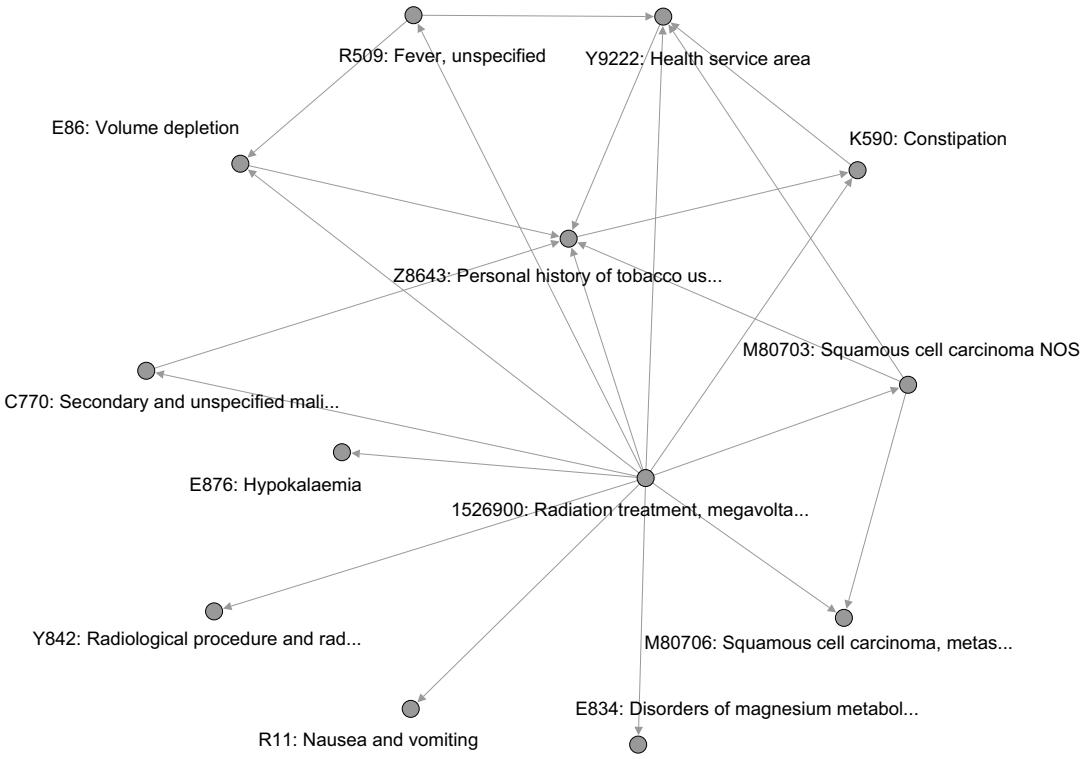


(a) Temporal associations between "*1526900: Radiation treatment*" and other codes, generated by the temporal comorbidity analysis baseline.

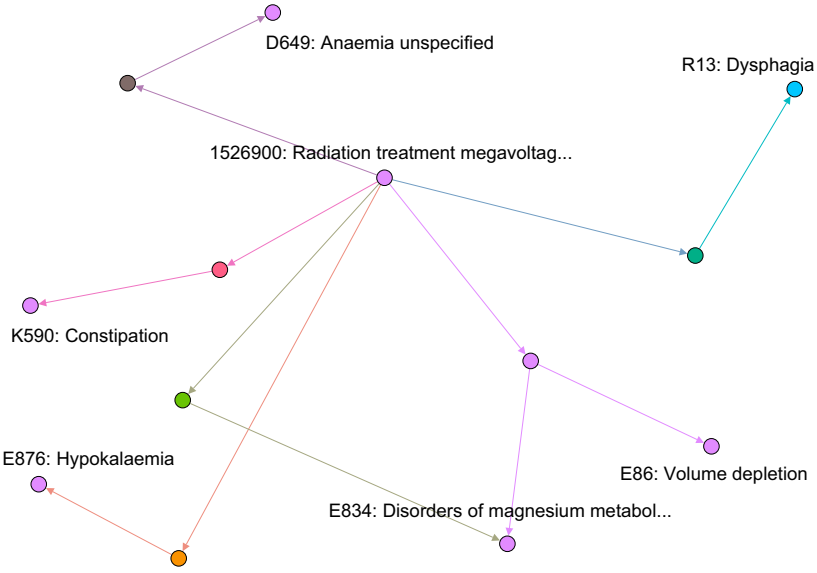


(b) Temporal association rules where the rule antecedent is "*1526900: Radiation treatment*", generated by our method **LTARM**.

Fig. 4. Network graphs showing toxicities/complications which occur after *one month* of radiation treatment. The temporal comorbidity analysis baseline produces only pairwise relations (a). Our method **LTARM** produces TARs represented by additional meta nodes (b).



(a) Temporal associations between "*1526900: Radiation treatment*" and other codes, generated by the temporal comorbidity analysis baseline.



(b) Temporal association rules where the rule antecedent is "*1526900: Radiation treatment*", generated by our method **LTARM**.

Fig. 5. Network graphs showing toxicities/complications which occur after *six months* of radiation treatment. The temporal comorbidity analysis baseline produces only pairwise relations (a). Our method **LTARM** produces TARs represented by additional meta nodes (b).

different time gap constraints. Moreover, the results returned by both methods confirm many well-known complications of radiation treatments.

In summary, although the types of cancer treatment toxicities are often known, their occurrence is not always well documented in daily care. The rules discovered reflect the patterns which have occurred in a particular patient cohort. Therefore, medical coding provides an indirect way to understand what is happening in a real patient population. Because medical coding is readily available, our temporal association rule based approach and visualization provide a solution for quality assurance and to evaluate the adverse effects of new treatment regimes.

5.2. Runtime analysis

In the second experiment, we analyze the efficiency of our proposed algorithm **LTARM** in terms of runtime. We compare **LTARM** with two other baselines. The experiment is performed on a computer with an Intel Core i7-5500U CPU at 3.00 GHz and 8 GB of RAM running OS Windows 8.1 (64-bit). The algorithms are coded in C# using Visual Studio .NET 2015 (.NET Framework 4.6.01055).

5.2.1. Baseline

For a comprehensive comparison, we employ two state-of-the-art baselines. The first baseline first mines frequent codesets from the dataset using the Apriori algorithm [14], which has to scan the dataset multiple times to compute the support of candidate codesets. It then generates TARs from mined frequent codesets, using the algorithm introduced in [14], which has to find all proper subsets of each frequent codeset to generate candidate rules. We name this baseline Apriori-TARM, which stands for *Apriori-based Temporal Association Rule Mining*. The second baseline first mines frequent codesets using the FPGrowth algorithm [15], which does not generate candidate codesets. It then generates TARs in the same way as Apriori-TARM. We name it FPGrowth-TARM, which stands for *FPGrowth-based Temporal Association Rule Mining*. Note that we implement FPGrowth-TARM based on the source code of FPGrowth algorithm provided in the SPMF library[3].

It is worth to note that we do not compare our method with Apriori and FPGrowth because *they only find frequent codesets, not TARs*. Instead, we compare with available methods for mining TARs, which extract frequent codesets using Apriori or FPGrowth and generate TARs by enumerating all possible subsets of each frequent itemset.

Complexity analysis. Apriori-TARM requires $\mathcal{O}(|\mathcal{D}| \times |\mathcal{C}| \times 2^{|\mathcal{C}|})$ for mining frequent codesets and $\mathcal{O}(|\mathcal{F}| \times 2^k)$ for generating temporal association rules (TARs). Similarly, FPGrowth-TARM requires $\mathcal{O}(|\mathcal{D}| + |\mathcal{T}|^2 \times d)$ for mining frequent codesets and $\mathcal{O}(|\mathcal{F}| \times 2^k)$ for generating TARs. Note that $|\mathcal{D}|$ is the number of patients (transactions), $|\mathcal{C}|$ is the number of diagnosis codes (items), $|\mathcal{F}|$ is the number of frequent codesets (frequent itemsets), k is the average length of all frequent codesets, $|\mathcal{T}|$ is the number of nodes on the FP-tree, and d is the depth of the FP-tree.

5.2.2. Parameter settings

All three methods Apriori-TARM, FPGrowth-TARM, and our **LTARM** require three parameters: the minimum support threshold δ for discovering frequent codesets, the minimum direction confidence threshold λ and the time gap $\triangle$ for generating TARs. Since the number of TARs generated in the second phase depends on the number of frequent codesets found in the first phase, we examine how the different values of δ affect the execution time of three methods. More specifically, we fix $\lambda = 10\%$ and $\triangle = 30$ days while we vary δ in the range [0.3%, 1.3%] with a step of 0.2%.

[3] The SPMF library: http://www.philippe-fournier-viger.com/spmf/.

5.2.3. Result and discussion

Fig. 6(a) shows the numbers of frequent codesets and TARs mined from the dataset. From the figure, we can see that these numbers increase when the δ value decreases. The number of rules is fewer than the number of frequent codesets with δ values in the range [0.7%, 1.3%]. However, it is significantly greater than that of frequent codesets at $\delta = 0.3\%$. This demonstrates that finding TARs becomes a complicated task when δ values are small.

Fig. 6(b) shows the total runtimes of three algorithms while Table 4 describes their runtimes for finding frequent codesets and generating rules. The results show that **LTARM** is much more efficient than Apriori-TARM in both two phases of the mining process. Let consider $\delta = 0.3\%$ in Fig. 6(b) and Table 4 as an example. The total mining time of **LTARM** is only 29.679 (s) compared with 224.009 (s) of Apriori-TARM. Particularly, **LTARM** takes only 2.675 (s) to build the lattice structure and 27.004 (s) to generate TARs from the lattice whereas Apriori-TARM consumes 67.687 (s) to find frequent codesets and 156.322 (s) to generate rules from these frequent codesets. For this example, **LTARM** is 8.0 times faster than Apriori-TARM. This can be explained that Apriori-TARM has to scan the dataset multiple times to discover frequent codesets and generate all subsets of a frequent codeset to derive candidate rules. In contrast, **LTARM** scans the dataset only one time to build the lattice structure and traverses the lattice to generate candidate rules. In addition, it also uses two Theorems 2 and 3 to remove invalid rules without computing the direction support and direction confidence. Consequently, **LTARM** significantly improves the mining time.

FPGrowth-TARM and **LTARM** behave very similar in finding frequent codesets (even FPGrowth-TARM is faster than **LTARM** at $\delta = 0.3\%$). However, when considering the total runtime of both two phases, **LTARM** remarkably outperforms FPGrowth-TARM. For instance, **LTARM** is 5.0 times faster than FPGrowth-TARM at $\delta = 0.3\%$. The difference is pronounced clearly because FPGrowth-TARM generates rules in the same way as Apriori-TARM.

More experiments to show the runtime efficiency of our algorithm **LTARM** on very large datasets are reported in Appendix A.

6. Conclusion and future work

In this work, we have showed that TARs can be used to understand the intricacies of adverse effects of radiation treatments and their progressions in real patient populations. Compared to the temporal comorbidity analysis method, our method based on TARs can effectively utilize the time gap constraints and capture the co-occurrence of toxicities. We have also developed an efficient algorithm for mining TARs, named **LTARM**. While existing algorithms are time-consuming, **LTARM** overcomes this weakness by employing a lattice structure and two theorems, which can help to eliminate invalid candidate rules quickly. As a result, the mining time was reduced remarkably, in particular, **LTARM** was 8.0 times faster than an Apriori-based algorithm and 5.0 times faster than a FPGrowth-based algorithm.

Our approach forms the basis for a clinical decision support system similar to our iHealthMap system [41]. We are in the process of implementing the graph visualization as an effective interface to query historical patient data. When considering treatment options for a new patient, it allows an oncologist to quickly identify the most relevant toxicity risks and find out past patients with the treatment and toxicity reaction. By comparing the current patient against these historical records and considering the prognosis [42], the oncologist can better balance the merit and risk of treatment options.

The itemset concept in temporal association rule mining provides a simple method to group co-occurring diagnosis codes.

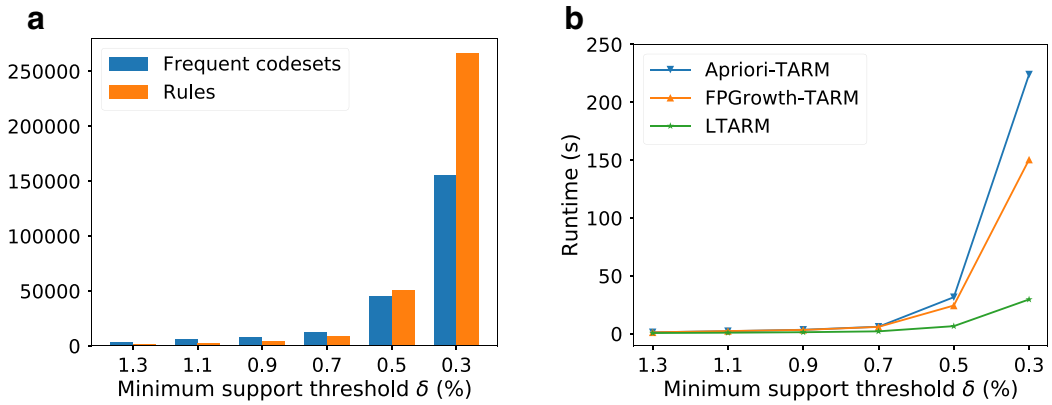


Fig. 6. The number of discovered frequent codesets and rules (a) and the total runtimes of Apriori-TARM, FPGrowth-TARM, and our **LTARM** (b) on the clinical dataset.

Table 4
Runtimes (in second) for mining frequent codesets (MFC) and generating rules (GR) of three methods with different δ values. A lower runtime means better. Bold font marks the best performance in a column.

δ (%)	1.3%		1.1%		0.9%		0.7%		0.5%		0.3%	
Method	**MFC**	**GR**	**MFC**	**GR**	**MFC**	**GR**	**MFC**	**GR**	**MFC**	**GR**	**MFC**	**GR**
Apriori-TARM	0.213	1.098	0.313	1.947	0.481	2.983	0.872	5.229	7.049	24.560	67.687	156.322
FPGrowth-TARM	0.223	1.019	0.308	2.046	0.427	2.922	0.739	5.217	1.075	23.330	**1.662**	148.804
LTARM (Ours)	**0.176**	**0.496**	**0.195**	**0.751**	**0.248**	**1.113**	**0.358**	**1.758**	**0.957**	**5.644**	2.675	**27.004**

However, it may generate hundreds of thousands of frequent codesets. In the future, we will consider alternative solutions, one of which is to apply topic modeling [43]. We are in the process of exploring this direction. Another further topic is to apply our approach to discover TARs from different data sources [44], which should provide more interesting information than those come from a single data source. Since our method needs a post-processing step to extract rules which contain a particular code (e.g., "1526900: Radiation treatment"), it will take time if the number of generated rules is very large. Thus, we will also investigate the itemset constraints [45,46] and how to integrate them into temporal association rule mining to avoid the post-processing step.

Acknowledgment

Dinh Phung gratefully acknowledges the partial support from the Australian Research Council (ARC).

Appendix A. More experiments

In this section, we conduct more experiments to compare the runtime performance of our algorithm **LTARM** with those of two baselines Apriori-TARM and FPGrowth-TARM (introduced in Section 5.2.1). Our goal is to demonstrate that **LTARM** still works efficiently on very large datasets and it is much faster than the two baselines when the numbers of generated frequent itemsets and rules are huge.

A1. Datasets

We use two very large real-world datasets, namely *EDA* and *Retail*. The *EDA* dataset[4] consists of 177,750 hospitalized patients admitted to the emergency department of the University Hospital Geelong in Australia between June 2008 and May 2015. Each patient (transaction) is a set of diagnosis codes (items) and each code in a patient is associated with a time-stamp. Since most codes appear only few times, their frequencies (supports) are very low. To be able to obtain a large number of frequent itemsets and rules, we select top-100 codes w.r.t their frequency and then select only patients who have these codes. As a result, the dataset contains 161,133 patients. We also create a smaller version of *EDA*, where we randomly select 50% its patients. We name this dataset *EDA-small*.

The *Retail* dataset[5] contains the transactions occurring between 2010/12/01 and 2011/12/09 of a United Kingdom-based online retailer. Each transaction is a set of products (items) purchased by customers and each item in a transaction is associated with a time-stamp. Following the same procedure as done with *EDA*, we select top-100 frequent products and then retain only transactions which contain these products. As a result, the dataset contains 4,006 transactions. We also create a smaller version of *Retail*, where we randomly select 50% its transactions. We name this dataset *Retail-small*.

The characteristics of four experimental datasets are summarized in Table A.5, including the number of transactions, the number of items, the minimum, maximum, and average numbers of items in a transaction.

A2. Parameter settings

For *EDA-small* and *EDA* datasets, we choose $\lambda = 1\%$ and $\triangle = 7$ days while we vary δ in the range [0.1%, 0.6%] with a step of 0.1%. For *Retail-small* and *Retail* datasets, we choose $\lambda = 10\%$ and $\triangle = 30$ days while we vary δ in the range [1%, 6%] with a step of 1%.

A3. Result and discussion

Figs. A.7–A.10 show the number of frequent itemsets, the number of rules, and the runtimes of three algorithms on four datasets. From the figures, we can observe the following points:

[4] Download from https://data.gov.au/dataset/emergency-hospitals-city-of-greater-geelong [5] Download from https://archive.ics.uci.edu/ml/datasets/OnlineRetail.

Table A1
Statistics of four transaction datasets.

Dataset	# transactions	# items	min. length	max. length	avg. length
EDA-small	80,000	100	1	24	1.80
EDA	161,133	100	1	42	1.80
Retail-small	2,000	100	1	99	10.53
Retail	4,006	100	1	99	10.65

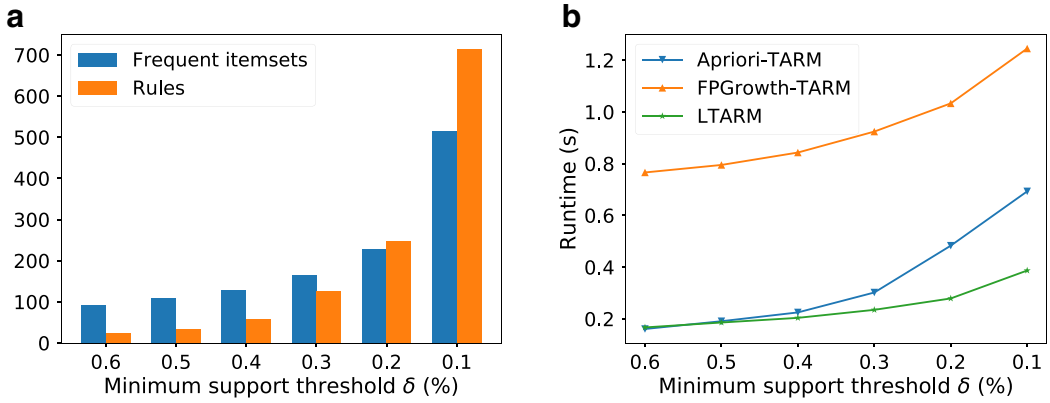


Fig. A1. The number of discovered frequent itemsets and rules (a) and the runtimes of Apriori-TARM, FPGrowth-TARM, and our **LTARM** (b) on the *EDA-small* dataset.

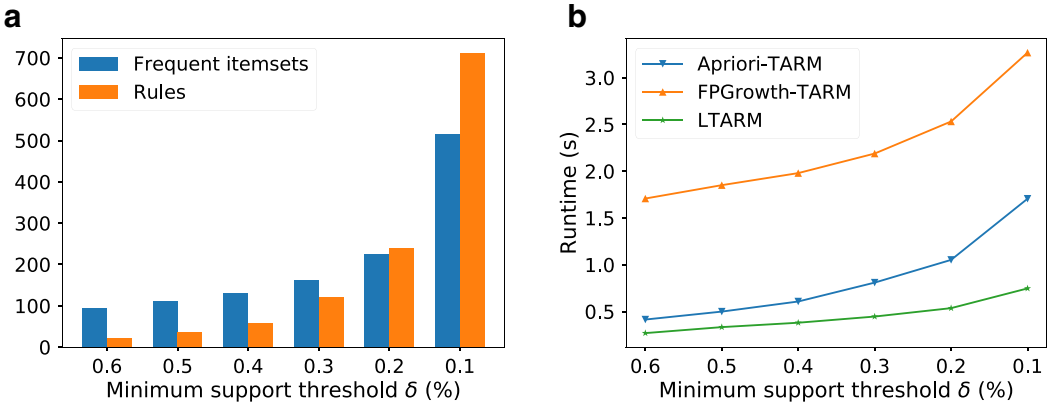


Fig. A2. The number of discovered frequent itemsets and rules (a) and the runtimes of Apriori-TARM, FPGrowth-TARM, and our **LTARM** (b) on the *EDA* dataset.

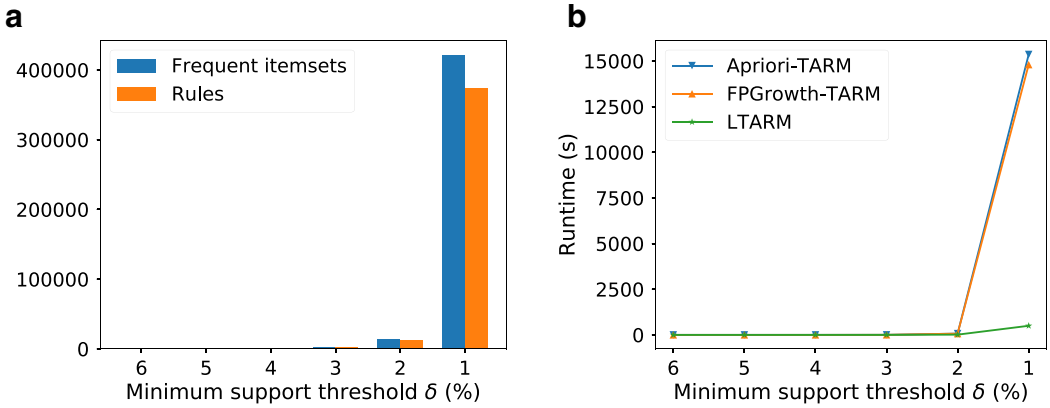


Fig. A3. The number of discovered frequent itemsets and rules (a) and the runtimes of Apriori-TARM, FPGrowth-TARM, and our **LTARM** (b) on the *Retail-small* dataset.

- Our algorithm **LTARM** always outperforms the two baselines Apriori-TARM and FPGrowth-TARM, where it achieves 1.8–30.8 and 3.2–30.1 times faster than Apriori-TARM and FPGrowth-TARM respectively.
- On two datasets *EDA-small* and *EDA*, where the numbers of frequent itemsets and rules are not very large, all three algorithms can finish their mining task quickly, and the runtime differences among three algorithms are insignificant.

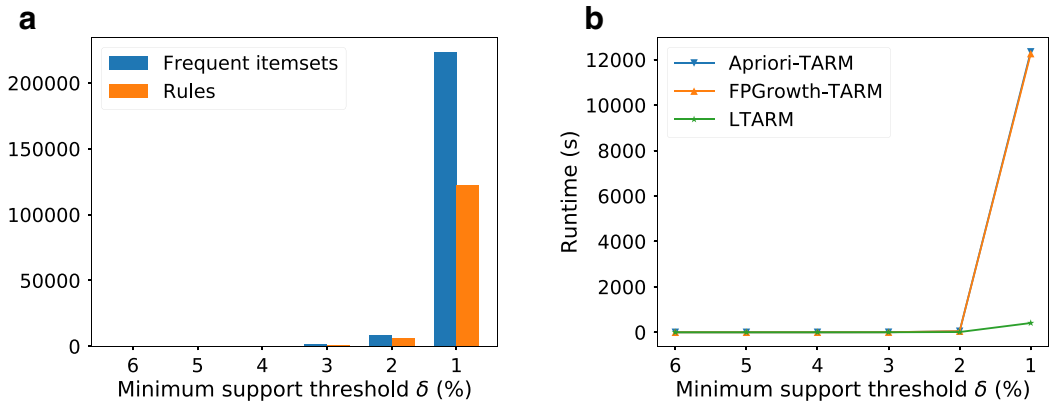


Fig. A4. The number of discovered frequent itemsets and rules (a) and the runtimes of Apriori-TARM, FPGrowth-TARM, and our **LTARM** (b) on the *Retail* dataset.

- On two datasets *Retail-small* and *Retail*, where the numbers of frequent itemsets and rules are huge, our algorithm **LTARM** is remarkably faster than the baselines. For example, consider the dataset *Retail-small* with $\delta = 1\%$ in Fig. A.9(b). The mining time of **LTARM** is only 499.114 (s) in comparison with 15,360.548 (s) and 14,803.430 (s) of Apriori-TARM and FPGrowth-TARM respectively. For this example, **LTARM** is 30.8 times faster than Apriori-TARM and 29.7 times faster than FPGrowth-TARM.

In summary, the experimental results demonstrate that our algorithm **LTARM** works efficiently on very large datasets, and it significantly outperforms the two state-of-the-art baselines in terms of runtime when the numbers of generated frequent itemsets and rules are huge.

References

[1] S. Bernard, W. Christopher (Eds.), World Cancer Report 2014, World Health Organization 2014.
[2] I. Plenderleith, Treating the treatment: toxicity of cancer chemotherapy, Can. Family Phys. 36 (1990) 1827–1830.
[3] C. Shanholtz, Acute life-threatening toxicity of cancer treatment, Crit. Care Clin. 17 (3) (2001) 483–502.
[4] J. Cox, J. Stetz, T. Pajak, Toxicity criteria of the radiation therapy oncology group (RTOG) and the european organization for research and treatment of cancer (EORTC), Int. J. Radiat. Oncol. Biol. Phys. 31 (5) (1995) 1341–1346.
[5] A. Trotti, R. Byhardt, J. Stetz, C. Gwede, B. Corn, K. Fu, L. Gunderson, B. McCormick, M. Morris, T. Rich, et al., Common toxicity criteria: version 2.0. an improved reference for grading the acute effects of cancer treatment: impact on radiotherapy, Int. J. Radiat. Oncol. Biol. Phys. 47 (1) (2000) 13–47.
[6] A. Trotti, The evolution and application of toxicity criteria, in: Seminars in Radiation Oncology, vol. 12, Elsevier, 2002, pp. 1–3.
[7] K. Yoshida, H. Yamazaki, S. Nakamara, K. Masui, T. Kotsuma, H. Akiyama, E. Tanaka, Y. Yoshioka, Comparison of common terminology criteria for adverse events v3. 0 and radiation therapy oncology group toxicity score system after high-dose-rate interstitial brachytherapy as monotherapy for prostate cancer, Anticancer Res. 34 (4) (2014) 2015–2018.
[8] D. Hanauer, N. Ramakrishnan, Modeling temporal relationships in large scale clinical associations, J. Am. Med. Inf. Assoc. 20 (2) (2013) 332–341.
[9] M. Munson, J. Wrobel, C. Holmes, D. Hanauer, Data mining for identifying novel associations and temporal relationships with charcot foot, J. Diabetes Res. 2014 (2014).
[10] H. Nam, K. Lee, D. Lee, Identification of temporal association rules from time-series microarray data sets, BMC Bioinformatics 10 (Suppl 3) (2009) S6.
[11] S. Concaro, L. Sacchi, C. Cerra, P. Fratino, R. Bellazzi, et al., Mining health care administrative data with temporal association rules on hybrid events, Methods Inf. Med. 50 (2) (2011) 166–179.
[12] D. Nguyen, W. Luo, D. Phung, S. Venkatesh, Understanding toxicities and complications of cancer treatment: a data mining approach, in: Australasian Joint Conference on Artificial Intelligence, Springer, 2015, pp. 431–443.
[13] L.T.T. Nguyen, B. Vo, L.T.T. Nguyen, P. Fournier-Viger, A. Selamat, ETARM: An efficient top-k association rule mining algorithm, Appl. Intell. (2017) 1–13.
[14] R. Agrawal, R. Srikant, Fast algorithms for mining association rules in large databases, in: VLDB, 1994, pp. 487–499.
[15] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Min. Knowl. Discov. 8 (1) (2004) 53–87.
[16] M. Zaki, K. Gouda, Fast vertical mining using diffsets, in: KDD, ACM, 2003, pp. 326–335.
[17] H. Yang, P. Chen, Data mining in lung cancer pathologic staging diagnosis: correlation between clinical and pathology information, Expert Syst. Appl. 42 (15) (2015) 6168–6176.
[18] J. Alwidian, B. Hammo, N. Obeid, WCBA: Weighted classification based on association rules algorithm for breast cancer disease, Appl. Soft Comput. 62 (2018) 536–549.
[19] N. Sharma, H. Om, Data mining models for predicting oral cancer survivability, Netw. Model. Anal. Health Inf. Bioinform. 2 (4) (2013) 285–295.
[20] W.-T. Tseng, W.-F. Chiang, S.-Y. Liu, J. Roan, C.-N. Lin, The application of data mining techniques to oral cancer prognosis, J. Med. Syst. 39 (5) (2015) 59–66.
[21] Y. Chen, R. Xu, Mining cancer-specific disease comorbidities from a large observational health database, Cancer Inform. 13 (Suppl 1) (2014) 37–45.
[22] M. Zolbanin, D. Delen, H. Zadeh, Predicting overall survivability in comorbidity of cancers: a data mining approach, Decis. Support Syst. 74 (2015) 150–161.
[23] Y. Li, P. Ning, S. Wang, S. Jajodia, Discovering calendar-based temporal association rules, Data Knowl. Eng. 44 (2) (2003) 193–218.
[24] M. Ghorbani, M. Abessi, A new methodology for mining frequent itemsets on temporal data, IEEE Trans. Eng. Manage. 64 (4) (2017) 566–573.
[25] L. Sacchi, C. Larizza, C. Combi, R. Bellazzi, Data mining with temporal abstractions: learning rules from time series, Data Min. Knowl. Discov. 15 (2) (2007) 217–247.
[26] R. Moskovitch, Y. Shahar, Fast time intervals mining using the transitivity of temporal relations, Knowl. Inf. Syst. 42 (1) (2015) 21–48.
[27] C.-W. Lin, W. Gan, P. Fournier-Viger, T.-P. Hong, Efficient algorithms for mining recent weighted frequent itemsets in temporal transactional databases, in: The 31st Annual ACM Symposium on Applied Computing, ACM, 2016, pp. 861–866.
[28] E. Winarko, J. Roddick, ARMADA – an algorithm for discovering richer relative temporal association rules from interval-based data, Data Knowl. Eng. 63 (1) (2007) 76–90.
[29] D. Patel, W. Hsu, M. Lee, Mining relationships among interval-based events for classification, in: SIGMOD, ACM, 2008, pp. 393–404.
[30] R. Alves, D. Rodriguez-Baena, J. Aguilar-Ruiz, Gene association analysis: a survey of frequent pattern mining from gene expression data, Brief. Bioinformatics 11 (2) (2010) 210–224.
[31] X. Sun, M. Orlowska, X. Zhou, Finding event-oriented patterns in long temporal sequences, in: PAKDD, Springer, 2003, pp. 15–26.
[32] X. Sun, M. Orlowska, X. Li, Finding temporal features of event-oriented patterns, in: PAKDD, Springer, 2005, pp. 778–784.
[33] World Health Organization, International Classification of Diseases (ICD), 2013, (http://www.who.int/classifications/icd/en/).
[34] National Casemix and Classification Centre, Australian coding standards, National Casemix and Classification Centre, Autralian Health Services Research Institute, University of Wollongong, eighth ed., 2013.
[35] P. Fournier-Viger, C.-W. Lin, B. Vo, T. Chi, J. Zhang, B. Le, A survey of itemset mining, in: Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 7, 2017, p. e1207, doi:10.1002/widm.1207.
[36] J. Cohen, The earth is round ($p < .05$), Ame. Psychol. 49 (12) (1994) 997–1003.
[37] R. Díaz-Uriarte, A. Andres, Gene selection and classification of microarray data using random forest, BMC Bioinformatics 7 (1) (2006) 3.
[38] N. Adnan, Z. Islam, Forest CERN: a new decision forest building technique, in: PAKDD, 2016, pp. 304–315.
[39] B. Mathieu, H. Sebastien, J. Mathieu, Gephi: An Open Source Software for Exploring and Manipulating Networks, 2009.
[40] M. Hahsler, S. Chelluboina, Visualizing association rules: introduction to the r-extension package arulesviz, R project module (2011) 223–238.
[41] R. Santu, L. Wei, T. Truyen, P. Dinh, V. Svetha, H. Richard, HealthMap: a visual platform for patient suicide risk review, Technical Report, HISA Big Data, Melbourne, Australia, 2014.
[42] S. Gupta, T. Tran, W. Luo, D. Phung, L. Kennedy, A. Broad, D. Campbell, D. Kipp, M. Singh, M. Khasraw, et al., Machine-learning prediction of cancer survival: a

retrospective study using electronic administrative records and a cancer registry, BMJ Open 4 (3) (2014) e004007.
[43] W. Luo, D. Phung, V. Nguyen, T. Tran, S. Venkatesh, Speed up health research through topic modeling of coded clinical data, International Workshop on Pattern Recognition for Healthcare Analytics, 2014.
[44] Y. Zhao, H. Zhang, F. Figueiredo, L. Cao, C. Zhang, Mining for combined association rules on multiple datasets, in: International Workshop on Domain Driven Data Mining, ACM, 2007, pp. 18–23.
[45] D. Nguyen, L.T.T. Nguyen, B. Vo, W. Pedrycz, Efficient mining of class association rules with the itemset constraint, Knowl. Based Syst. 103 (2016) 73–88.
[46] B. Vo, T. Le, W. Pedrycz, G. Nguyen, W. Baik, Mining erasable itemsets with subset and superset itemset constraints, Expert Syst. Appl. 69 (2017) 50–61.