**Mining Non-Redundant Inter-Transaction Rules**

CHUN-SHENG WANG  
Department of Information Management  
Jinwen University of Science and Technology  
New Taipei City, 231 Taiwan

Inter-transaction pattern and rule mining has been the subject of active research and has potential application in many areas. All the previous inter-transaction rule mining studies proposed generating a full set of inter-transaction rules (i.e., all frequent and confident rules) from a full set of inter-transaction patterns (i.e., all frequent patterns). However, generating a full set inter-transaction rules can be very expensive. In this paper, to resolve the explosive growth of inter-transaction rules, we propose a new research of mining non-redundant inter-transaction rules. We investigate and characterize non-redundant inter-transaction rules, study the quality inter-transaction rule sets with respect to completeness and tightness. We develop an algorithm named NRITR-Miner to mine the non-redundant inter-transaction rules efficiently and an algorithm named ITR-Miner to mine the full-set of inter-transaction rules for comparing purpose. We demonstrate via experiments that the proposed algorithm is efficient and scalable, and outperform the compared algorithm in all cases.

Keywords: association rules, closed pattern, data mining, inter-transaction pattern, inter-transaction rules, non-redundant rules

1. INTRODUCTION

Mining association rules is a fundamental problem in the area of data mining, in which inter-transaction association rule first proposed by Lu, Feng, and Han has been the subject of active research [3-5, 8, 15, 16, 18]. Traditional association rule mining algorithms focus on association rules among itemsets within a transaction. However, the inter-transaction association rule represents some association relationships among the itemsets from different transactions. Given a database containing transactions, inter-transaction rule identifies inter-transaction patterns appearing with enough support and confidence. It has potential application in many areas such as analysis of market data, stock market data, weather forecasting data, etc.

Inter-transaction rules describe association relationships among the itemsets from different transactions [1, 10, 17]. An inter-transaction rule expresses as \( X \rightarrow Y \), specifies that there is sufficiently high confidence that the pattern \( X \) will occur in some transactions following an occurrence of \( Y \) in other transactions. For example, an intra-transactional association rule may like: “If the stock prices of Microsoft and IBM go up, the price of Apple stock is likely to go up two days later.” Compared to inter-transaction patterns, rules allow better understanding of inter-transactional behaviors exhibited in a transaction database. Consider the above example of price behavior in a stock market [15]: ‘If the stock prices of Microsoft and IBM go up, the price of Apple stock is likely to
The price go up inter-transaction pattern \{Microsoft (0), IBM (0), Apple (2)\} is the pattern showing the price go up behavior. However, image a stock market with numerous companies with various price behaviors. The pattern \{Microsoft (0), IBM (0), Apple (2)\} will tend to occur with a low support. Mining with low support will return the pattern, however, typically along with many irrelevant or spurious patterns. Rules can throw away many spurious patterns by introducing the notion of confidence the set of patterns. Only rules satisfying both support and confidence thresholds are mined.

In inter-transaction rules research, Feng et al. [3] applied inter-transaction association rule mining algorithms to the prediction of trends in meteorological and stock market data. Lu et al. [15] proposed the EH-Apriori algorithm, which uses the Apriori algorithm to discover frequent inter-transaction itemsets. To enhance their algorithm’s efficiency, they used a hashing technique to reduce the number of candidate itemsets of length two. Feng et al. [4] used templates to reduce the number of rules. Tung et al. [16] developed an algorithm, called FITI (First Intra Then Inter), which discovers frequent intra-transaction itemsets and uses them to generate frequent inter-transaction itemsets. Lee & Wang [8] proposed an algorithm, called ITP-Miner, which uses an ITP-tree to mine all frequent inter-transaction itemsets in a depth-first search manner. It has been shown that the ITP-Miner algorithm outperforms the previous EH-Apriori and FITI algorithms. More recently, Wang & Chu [18] proposed an algorithm, called PITP-Miner, which utilizes a projection based approach to mine frequent inter-transaction patterns efficiently. It has been shown that the PITP-Miner algorithm outperforms the previous inter-transaction mining algorithms in most cases.

All the previous inter-transaction rule mining studies [3-5, 8, 9, 15, 16, 18] propose generating a full set of inter-transaction rules (i.e., all frequent and confident rules) from a full set of inter-transaction patterns (i.e., all frequent patterns). Generating a full set inter-transaction rules can be very expensive. The number of frequent patterns is exponential to the maximum patterns length: if an inter-transaction pattern of length \(l\) is frequent, all its \(O(2^l)\) sub-patterns are frequent as well. Each frequent pattern of length \(l\) can possibly generate \(l-1\) rules (depending on the minimum confidence threshold). Hence, there is an exponential growth in the number of rules with respect to the maximum pattern length.

To resolve the explosive growth of inter-transaction rules, we propose mining a non-redundant set of inter-transaction rules. The notion of rule inference is core of our method and is used to define and remove redundancy among rules. When using the set of mined rules as a guideline, replacing a full set of rules with the non-redundant subset of rules does not impact the accuracy of the guideline [2, 11, 14, 21].

There have been many studies on mining frequent inter-transaction patterns [3-5, 8, 9, 15-18]. These studies include those mining a compact representation of patterns, referred as closed inter-transaction patterns [9]. This compact representative can be mined with much more efficiency than the full set of frequent pattern. However, there has not been any study relating this non-redundant set of inter-transaction rules. In this paper, we focus on the following research questions: What do we mean by a non-redundant set of inter-transaction rules? How to use inter-transaction patterns to form non-redundant rules? How a non-redundant set of inter-transaction rules be obtained from closed inter-transaction patterns. What type of inter-transaction patterns need to be
mined to form non-redundant inter-transaction rules? How to characterize the non-redundant set of inter-transaction rules? How to design an efficient algorithm to obtain a non-redundant set of inter-transaction rules from inter-transaction patterns? In this paper, we address the above research questions. We investigate and characterize non-redundant inter-transaction rules. In addition, we propose algorithms and perform a performance study on mining a non-redundant set of inter-transaction rules. A rule set can be formed by composing patterns. The compositions are evaluated based on the two criteria of completeness and tightness. A rule set is complete, if each frequent and confident rule can be inferred by one of the rules in the rule set. A rule set is tight, if the set contains no redundant rules. We characterize a tight and complete set of non-redundant rules configuration.

Therefore, in this paper, the contributions of our work are as follows:

1. We propose a concept of non-redundant inter-transaction rules.
2. We study the quality inter-transaction rule sets with respect to completeness and tightness.
3. We characterize a tight and complete set of non-redundant inter-transaction rules based on compositions of patterns.
4. We develop an algorithm named NRITR-Miner to mine the non-redundant inter-transaction rules efficiently.
5. We develop an algorithm named ITR-Miner to mine the full-set of inter-transaction rules for comparing purpose.
6. We demonstrate via experiments that the proposed NRITR-Miner algorithm is efficient and scalable, and outperform the compared ITR-Miner algorithm in all cases.

The remainder of this paper is organized as follows. Section 2 introduces the non-redundant inter-transaction rules mining problem and defines some notations. In Section 3, we characterize a tight and complete set of non-redundant inter-transaction rules based on compositions of patterns. Then we present the NTITR-Miner algorithm for mining non-redundant inter-transaction rules, and the ITR-Miner algorithm for mining full-set of inter-transaction rules. Section 4 describes the experiments and their performance results. Finally, we present our conclusions and indicate some future research directions in Section 5.

2. DEFINITIONS

Let \( I = \{i_1, i_2, \ldots, i_m\} \) be a set of distinct items, and \( A = \{a_1, a_2, \ldots, a_n\} \) be a set of domain attributes. A transaction database \( DB \) contains a set of transactions in the form of \( <dat, T_{dat}> \), where \( dat \in A, T_{dat} \subseteq I, T_{dat} \) is an itemset, and \( dat \) is the domain attribute of \( T_{dat} \) that describes a property, such as the time stamp associated with \( T_{dat} \). We can also say that the \( T_{dat} \) itemset occurs in a transaction with the domain attribute \( dat \), or occurs at \( dat \).

For example, Fig. 1 shows a transaction database \( DB \) containing six transactions, namely, \( <1, T_1>, <2, T_2>, <3, T_3>, <4, T_4>, <5, T_5>, \) and \( <6, T_6> \). \( T_1 \) is an itemset \( \{a, b, d\} \)
that occurs at dat=1. If this database registers the stocks that record an increase at the end of each trading day, the first transaction means the stock prices of a, b, and d went up on day 1.

Definition 1 (Extended transaction) Let <u, T_u> and <v, T_v> be two transactions in a transaction database. The relative distance between u and v is defined as (u – v), where u > v, and v is called the reference point. With respect to v, an item i_k at u is called an extended item and denoted as i_k(u – v), where (u – v) is called the span of the extended item. Similarly, with respect to v (or the transaction at v), a transaction T_u at u is called an extended transaction and denoted as T_u(u – v). Therefore, an extended transaction consists of a set of extended items, i.e., T_u(u – v) = {i_s(u – v), ..., i_s(u – v)}, where s is the number of items in T_u.

For example, in the database shown in Fig. 1, the extended transaction of the transaction at dat=3 with respect to the transaction at dat=1 is \{a(2), b(2)\}.

Definition 2 (Maxspan) An inter-transaction pattern is defined as a set of extended items, \{x_i(d_1), x_i(d_2), ..., x_i(d_k)\}, where d_k = maxspan, maxspan is a user-specified maximum span, x_i(d_j) < x_i(d_k), and 1 ≤ j ≤ k.

Definition 3 (Mega-transaction) For a transaction <d_1, T_d1> in a transaction database DB, a mega-transaction, M_{d_1}, is defined as a set of extended transactions in D with respect to d_1, i.e., M_{d_1} = T_{d_1(0)} \cup T_{d_1(d_1-2)} \cup T_{d_1(d_1-1)} \cup ..., \cup T_{d_1(d_k-1)}), where <d_1, T_{d_1(0)}, <d_2, T_{d_2}, ..., <d_k, T_{d_k}> are consecutive transactions in DB, d_1 ≤ d_k, and d_k-d_1 ≤ maxspan.

Note that M_{d_1} is also an inter-transaction pattern in which d_1 is the reference point. For example, with maxspan=1, the database in Fig. 1 contains six mega-transactions: M_1 = \{a(0), b(0), d(0), b(1)\}, M_2 = \{b(0), a(1), b(1)\}, M_3 = \{a(0), b(0), a(1), (1), (1)\}, M_4 = \{a(0), b(0), c(0), a(1), b(1), (1), (1)\}, M_5 = \{a(0), b(0), c(0), d(0), a(1), c(1)\}, and M_6 = \{a(0), c(0)\}.

Definition 4 (Pattern length and concatenation) The number of extended items in a pattern is called the length of the pattern. A pattern of length l is called an l-pattern. A pattern P_1 + P_2 denotes the concatenation of pattern P_1 and pattern P_2.

For example, \{a(0), b(0), a(1), c(1)\} is a 4-pattern, \{a(0), c(0)\} + \{d(1), c(2)\} = \{a(0), c(0), d(1), c(2)\}.

Definition 5 (Sub-pattern) Let \{x_1(i_1), x_2(i_2), ..., x_m(i_m)\} and \{y_1(j_1), y_2(j_2), ..., y_n(j_n)\} be two patterns. We say that Y is a super-pattern (or superset) of X if we can find a
non-negative integer \( s \) and \( m \) extended items in \( Y, y_s(j_{i1}), y_s(j_{i2}), \ldots, y_m(j_{im}) \), such that \( y_s(j_{i1})=x_1(i_1), y_s(j_{i2})=x_2(i_2), \ldots, y_m(j_{im})=x_m(i_m) \). In other words, \( X \) is a sub-pattern (or subset) of \( Y \), denoted as \( X \subseteq Y \). Note that if \( s \) is equal to 0, we say \( X \) is a proper subset of \( Y \), denoted as \( X \subset Y \).

For example, \( \{a(0), c(0)\} \subset \{a(0), b(0), a(1), c(1)\} \) and \( \{a(0), a(1)\} \subset_p \{a(0), b(0), a(1), c(1)\} \).

**Definition 6** (Pattern matching) A mega-transaction \( M \) is said to match a pattern \( P \) if \( P \subseteq M \). This is denoted by \( P[M] \). The inverse, that \( M \) does not match \( P \), is denoted by \( \neg P[M] \).

As an example, consider a pattern \( P = \{a(0), b(1)\} \) and two mega-transactions \( M_1 = \{a(0), b(0), d(0), b(1), c(1)\} \) and \( M_2 = \{b(0), d(0), a(1), b(1)\} \). The first mega-transaction \( M_1 \) matches \( P \) (denoted by \( P[M_1] \)) since \( P \subset M_1 \). The second mega-transaction \( M_2 \) does not match \( P \) (denoted by \( \neg P[M_2] \)) since \( P \not\subset M_2 \).

**Definition 7** (Frequent pattern) Given a transaction database \( DB \), a user-specified minimum support threshold \( \text{min\_sup} \), and an inter-transaction pattern \( X \), let \( D_x \) be the set of mega-transactions in \( DB \) matching \( X \). The support of \( X, \text{sup}(X) \), is defined as \( |D_x| \). If \( \text{sup}(X) \geq \text{min\_sup} \), we can say that \( X \) is a frequent pattern.

Consider the example in Fig. 1. Assume that \( \text{max\_span}=1 \), \( \text{min\_sup}=3 \) and \( X=\{a(0), c(1)\} \). Since the mega-transaction formed by the third and fourth transactions, \( \{a(0), b(0), a(1), b(1), c(1)\} \), matches \( X \), \( \text{sup}(X) \) is set to 1. Also, the mega-transaction formed by the fourth and fifth transactions, \( \{a(0), b(0), c(0), a(1), b(1), c(1), d(1)\} \), matches \( X \). Thus, \( \text{sup}(X) \) is incremented by 1; that is, \( \text{sup}(X)=2 \). Moreover, the mega-transaction formed by the fifth and sixth transactions, \( \{a(0), b(0), c(0), d(0), a(1), c(1)\} \), matches \( X \), so \( \text{sup}(X) \) is incremented by 1; that is, \( \text{sup}(X)=3 \). Therefore, \( X \) is a frequent pattern.

Forming inter-transaction rules from frequent inter-transaction patterns under the support-confidence framework is analogous to forming association rules from frequent itemsets.

**Definition 8** (Inter-transaction rule) An inter-transaction rule is denoted by \( r = X \rightarrow Y \langle s, c \rangle \), where \( X=\{x_1(i_1), x_2(i_2), \ldots, x_m(i_m)\} \) and \( Y=\{y_1(j_1), y_2(j_2), \ldots, y_n(j_n)\} \) are inter-transaction patterns and \( s \) and \( c \) are the support and confidence values. We omit the support and confidence value if it is clear or irrelevant to the context. A rule \( r \) is constructed from two inter-transaction patterns: \( X \) and \( Y \), where \( X \cap Y = \emptyset \) and \( i_m < j_1 \). The support of \( r \), denoted by \( \text{sup}(r) \), is defined to be equal to \( \text{sup}(X+Y) \). On the other hand, the confidence of \( r \), denoted by \( \text{con}(r) \), is defined to be equal to \( \text{sup}(X+Y)/\text{sup}(X) \). A rule with support higher than a threshold \( \text{min\_sup} \) is considered frequent. A rule with confidence higher than a threshold \( \text{min\_conf} \) is considered confident.

### 3. THE PROPOSED METHOD

#### 3.1 Inference of Inter-Transaction Rules

Our approach to mining a non-redundant set of inter-transaction rules lies in a
construction based on rule inference. In the section we first define rule inference and mention properties relating to pattern-sets and rule inference.

**Definition 9 (Rule satisfiability)** A mega-transaction $M$ is said to satisfy an inter-transaction rule $r$ of the form $X \rightarrow Y$ if either one of the following two cases holds.

1. It matches the pre-condition and subsequently the post-condition of the rule, i.e., $\exists M_1, M_2. M = M_1 + M_2 \land X[M_1] \land Y[M_2]$. 
2. It does not match the pre-condition of the rule, i.e., $\exists M_1, M_2. M = M_1 + M_2 \land \neg X[M_1]$. 

A mega-transaction $M$ satisfying a rule $r$ is denoted by $r[M]$; otherwise, it is denoted by $\neg r[M]$. For example, consider a rule $r_1 = \{a(0)\} \rightarrow \{b(1), c(1), d(1)\}$ and two mega-transactions $M_1 = \{a(0), b(0), d(0), b(1), c(1)\}$ and $M_2 = \{b(0), d(0), a(1), b(1)\}$. For the first case, $M_1$ satisfies $r_1$ ($r_1[M_1]$) since $b(1)$ occurs after the occurrence of $a(0)$ in $M_1$. For the second case, $M_2$ also satisfies $r_1$ ($r_1[M_2]$) since we cannot find any $a(0)$ in $M_2$.

**Definition 10 (Rule inference)** Given a transaction database $DB$ and two inter-transaction rules $r_1, r_2$. $r_1$ is said to infer $r_2$ if and only if both of the following two cases holds:

1. $r_2[M]$ whenever $r_1[M]$, for every mega-transaction $M$ in $DB$. 
2. $\text{sup}(r_1) = \text{sup}(r_2)$ and $\text{conf}(r_1) = \text{conf}(r_2)$.

**Definition 11 (Redundant rules)** An inter-transaction rule is said to be redundant in a set of rules $R$ if it can be inferred by another rule in $R$. For example, consider the following two rules: $r_1 = \{a(0)\} \rightarrow \{b(1), c(1), d(1)\}$ and $r_2 = \{a(0)\} \rightarrow \{c(1)\}$ having the same support and confidence. $r_2$ is redundant since it can be inferred by $r_1$.

We now identify some properties associated with inter-transaction patterns. We then leverage on these properties to highlight the properties of rule inference and rule coverage.

**Property 1:** Consider two patterns $P_x$ and $P_y$ appearing in a transaction database $DB$. If $P_x \subset_p P_y$ and $\text{sup}(P_x) = \text{sup}(P_y)$, then $P_x$ and $P_y$ are supported by the same set of mega-transactions.

**Property 2 (Transitivity).** If $r_x$ inter $r_y$ and $r_y$ infers $r_z$, then $r_x$ infers $r_z$.

Before stating the necessary and sufficient condition of rule inference, we state 2 lemmas.

**Lemma 1:** Suppose for every mega-transaction $m$ in $DB$, we have $b \subset_p m$ when $a \subset_p m$, then it must be the case that $b \subset_p a$.

**Lemma 2:** Suppose for every mega-transaction $m$ in $DB$, we have $b \subset_p m$ when $a \subset_p m$, then it must be the case that $a \subset_p b$.

The sufficient and necessary condition of rule inference is as follows.
Property 3 (Sufficient and necessary inference): Given \( r_1 = X_1 \rightarrow Y_1 \) and \( r_2 = X_2 \rightarrow Y_2 \) generated from \( DB \). \( r_2 \) infers \( r_1 \) if and only if the following four conditions hold, (1) \( X_2 \subset_p X_1 \), (2) \( X_1 + Y_1 \subset_p X_2 + Y_2 \), (3) \( \text{sup}(r_1) = \text{sup}(r_2) \) and (4) \( \text{conf}(r_1) = \text{conf}(r_2) \).

Proof: The right-to-left direction: Suppose the four conditions hold. Condition (1) ensures that whenever \( X_2 \) does not hold \( X_1 \) will also not hold. Condition (2) ensures that whenever \( X_2 + Y_2 \) holds, \( X_1 + Y_1 \) also holds. The above conditions imply whenever \( r_2 \) holds, \( r_1 \) also holds. Conditions (3) and (4) ensure that \( r_2 \) has the same support and confidence as \( r_1 \). Hence, the above are sufficient condition for inference as in Definition 10. The left-to-right direction: Suppose \( r_2 \) infers \( r_1 \). Then \( r_2 \) and \( r_1 \) have equal support and confidence. Hence, conditions (3) and (4) hold. We only need to prove that conditions (1) and (2) hold. For any mega-transaction \( m \), we have \(-r_2[m]\) iff \( X_2 \subset_p m \) and \( X_2 + Y_2 \not\subset_p m \), and \(-r_1[m]\) iff \( X_1 \subset_p m \) and \( X_1 + Y_1 \not\subset_p m \). Taking the contra positive of \( r_2 \) infers \( r_1 \), we have \(-r_2[m]\) whenever \(-r_1[m]\). Thus \( X_2 \subset_p m \) whenever \( X_1 \subset_p m \), and \( X_2 + Y_2 \not\subset_p m \) whenever \( X_1 + Y_1 \not\subset_p m \). As \( m \) is arbitrary, by lemma 1, we conclude \( X_2 \subset_p X_1 \), proving condition (1). Also, by lemma 2, we conclude \( X_1 + Y_1 \subset_p X_2 + Y_2 \), proving condition (2).

As an example, let \( DB \) be a database of two transactions: \( \{a, b\} \) and \( \{a, b, d\} \). Let \( r_1 \) be \( \{a(0)\} \rightarrow \{b(1)\} \), \( r_2 \) be \( \{a(0)\} \rightarrow \{b(1), d(1)\} \) and \( r_3 \) be \( \{a(0)\} \rightarrow \{d(1)\} \). Then \( r_1 \) and \( r_2 \) satisfy the sufficient inference property. Hence \( r_2 \) infers \( r_1 \). However, \( r_1 \) does not infer \( r_3 \) and \( r_2 \) does not infer \( r_3 \), as \( \{a(0)\} + \{d(1)\} \not\subset_p \{a(0)\} + \{b(1)\} \) and \( \{a(0)\} + \{b(1)\} \not\subset_p \{a(0)\} + \{d(1)\} \).

3.2 Non-Redundancy of Inter-Transaction Rules

Generating inter-transaction rules from a full set of inter-transaction patterns can be exorbitant. Wang et al. [20] have shown that the size of a full set of sequential patterns is exponential to the maximum length of patterns in the closed pattern set.

We therefore advocate using generator and closed pattern sets to generate inter-transaction rules [7, 12, 13]. Furthermore, instead of generating all frequent and confident inter-transaction rules, we strive to generate a non-redundant set of inter-transaction rules from which all other rules can be derived. There are three technical challenges pertaining to this research direction: (1) The assurance that all interesting rules can be logically inferred from this non-redundant set (i.e., the non-redundant set is complete); (2) the assurance that the set of non-redundant rules is tight; and (3) the need to compute the support and confidence of all rules efficiently. The purpose of this section is to characterize the set of generating inter-transaction rules with respect to two properties namely:

1. Completeness: All frequent and confident rules can be inferred from the generated set of rules.
2. Tightness: There exists no two different rules \( r \) and \( r' \) in the final set of rules where \( r \) infers \( r' \).

Definition 12 (Frequent, Closed) An inter-transaction pattern \( P \) is considered frequent in \( DB \) when its support \( \text{sup}(P) \), exceeds a minimum threshold \( \text{min} \text{sup} \). A frequent inter-transaction pattern \( P \) is considered to be closed if there exists no proper super-pattern of \( P \) having the same support as \( P \) [19, 20, 22].
Definition 13 (Generator, Prefixed-Generator) A frequent inter-transaction pattern \( P \) is considered to be a generator in \( DB \) if there exists no proper sub-pattern of \( P \) having the same support as \( P \) in \( DB \). A frequent inter-transaction pattern \( P \) is considered to be a Prefixed-Generator if there exists no pattern \( P' \) where \( P' \) is a prefix of \( P \) and has the same support as \( P \).

Definition 14 (NR-Rule) Based on the above pattern sets, we define a set of inter-transaction rules, \( NR-Rule \), as the following set of rules: \( \{ X \rightarrow Y | X \in \text{Prefixed-Generator}, X + Y \in \text{Closed} \} \).

Proposition 4: \( NR-Rule \) is complete.

Proof: For every rule \( r \) that is frequent and confident, we would like to show that there exists \( r' \in NR-Rule \), where \( r' \) infers \( r \). First, consider an arbitrary rule \( r \) of the form \( X \rightarrow Y \), composed from two frequent patterns \( X \) and \( X + Y \). It can be inferred by another rule \( r_1 \) composed from \( X_1 \in \text{Prefixed-Generator} \) and \( X + Y \). This is the case since there must be a prefix \( X_1 \) of \( X \) that belongs to \( \text{Prefixed-Generator} \) with the same support as \( X \). Composing \( X_1 \) with \( X + Y \), will form a rule \( r_1 = X_1 \rightarrow Y_1 \) where \( X_1 + Y_1 = X + Y \). From Property 3, \( r_1 \) infers \( r \). Second, \( r_1 = X_1 \rightarrow Y_1 \) can be inferred by another rule \( r_2 \), composed from \( X_1 \) and \( X_1 + Y_2 \in \text{Closed} \). This is the case since there must be a super-pattern of \( X_1 + Y_1 \) which is in \( \text{Closed} \) with the same support as \( X_1 + Y_1 \). From Property 3, \( r_2 \) infers \( r_1 \). Complete now follows by transitivity of rule inference.

Proposition 5: \( NR-Rule \) is tight.

Proof: We prove that the set of rules is tight by contradiction. Suppose a rule \( r = X \rightarrow Y \) in \( NR-Rule \) is redundant and is inferred by another rule \( r' = X' \rightarrow Y' \) in \( NR-Rule \). From Property 3, we have: (1) \( X \subseteq pX \); (2) \( X + Y \subseteq pX' + Y' \); (3) \( \sup(r) = \sup(r') \); and (4) \( \text{conf}(r) = \text{conf}(r') \). Since both \( X + Y \) and \( X' + Y' \) are in \( \text{Closed} \), it is not possible that \( X + Y \subseteq pX' + Y' \) and \( \sup(r) = \sup(r') \). Also, since both \( X \) and \( X' \) are in \( \text{Prefixed-Generator} \), it is not possible that \( X \subseteq pX \), and \( \text{conf}(r) = \text{conf}(r') \). Therefore, tightness is proved by this contradiction.

Theorem 1: \( NR-Rule \) is complete and tight.

Proof: This is proved by Propositions 4 and 5.

3.3 A Discussion

In this section we discuss and compare our proposed non-redundant inter-transaction rules with prior researches on non-redundant association rules [21] and non-redundant sequential rules [11]. Traditional association rule focus on association relationship among unordered set of items (itemsets) within a transaction. The author in [21] uses the Minimal-Generator to generate all non-redundant association rules NRA-Rules, whose definitions and theorem are defined as follows.
Definition 15 (Minimal-Generator) A frequent itemset $I$ is considered to be a Minimal-Generator in $DB$ if there exists no subset of $I$ having the same support as $I$ [21].

Definition 16 (NRA-Rule) Based on the above definition, the author defines a set of non-redundant association rules, NRA-Rule, as the following set of rules: $\{X \rightarrow Y \mid X \in \text{Minimal-Generator}, X \cup Y \in \text{Minimal-Generator}\}$ [21].

Theorem 2: NRA-Rule is complete but not tight [21].

The difference between NR-Rule and NRA-Rule consists in two aspects. First, NR-Rule uses Prefixed-Generator and Closed patterns to generate non-redundant inter-transaction rules, while NRA-Rule uses Minimal-Generator patterns to generate non-redundant association rules. Second, NR-Rule is complete and tight, while NRA-Rule is complete but not tight.

Unlike the unordered itemsets in an association rule, a sequential rule expresses a relationship between two ordered sequences happening on after another. The authors in [11] use the Prefix-Key and CS-Closed to generate all non-redundant sequential rules NRS-Rules, whose definitions and theorem are defined as follows.

Definition 17 (CS-Closed) A frequent sequential pattern $S$ is considered to be a CS-Closed in a sequence database if there exists no proper super-sequence of $S$ having the same support as $S$ [11].

Definition 18 (Prefix-Key) A frequent sequential pattern $S$ is considered to be a Prefix-Key in a sequence database if there exists no sequential pattern $S'$ where $S'$ is a prefix of $S$ and has the same support as $S$ [11].

Definition 19 (NRS-Rule) Based on the above definitions, the authors define a set of non-redundant sequential rules, NRS-Rule, as the following set of rules: $\{X \rightarrow Y \mid X \in \text{Prefix-Key}, X+Y \in \text{CS-Closed}\}$[11].

Theorem 3: NRS-Rule is complete but not tight [11].

To compare NR-Rule with NRS-Rule. First, since Prefixed-Generator in NR-Rule is similar to Prefix-Key in NRS-Rule, and Closed in NR-Rule is similar to CS-Closed in NRS-Rule, the generation methods of the two types of non-redundant rules are similar. However, they are different in that NR-Rule is complete and tight while NRA-Rule is complete but not tight.

3.4 Mining Algorithms of Inter-Transaction Rules

To generate the set of non-redundant inter-transaction rules, NR-Rule, one must first mine the patterns. There is an existing PITP-Miner algorithm for mining all frequent inter-transaction patterns [18]; and an existing ICMiner algorithm for mining all closed inter-transaction patterns [9]. However, there is no algorithm in existing literature to mine Prefixed-Generator. Therefore, we modify the PITP-Miner to concurrently mine the sets of Prefixed-Generator and Closed inter-transaction patterns. The modified PITP-Miner algorithm is shown in Fig. 2, where the modified parts are in lines 9 and 10. In line
Algorithm: PITP-Miner: Mining frequent, closed, prefixed-generator patterns by a projection based approach.
Input: An input transaction database $DB$, a minimum support $min\_sup$, and a maximum span $max\_span$.
Output: A complete set of frequent inter-transaction patterns $FP$, closed patterns $C$, and prefixed-generator patterns $PG$.
1. Construct a hash table $H$. Let $H=\emptyset$, $FP=\emptyset$, $C=\emptyset$, and $PG=\emptyset$;
2. Scan $DB$ to find all frequent 1-patterns and update $H$;
3. For each frequent 1-pattern $p$ do:
   4. Construct a $p$-projected database $D|_{p}$ by the hash table pruning strategy to check $H$;
   5. Call Project($D|_{p}, p, H, FP, C, PG, min\_sup, max\_span$);
6. End For
7. Output $FP$, $C$, $PG$;

Subroutine: Project($D|_{p}, p, H, FP, C, PG, min\_sup, max\_span$)
Input: A projected database $D|_{p}$, a pattern $p$, a hash table $H$, a set of frequent patterns $FP$, a set of closed patterns $C$, a set of prefixed-generator patterns $PG$, a minimum support $min\_sup$, and a maximum span $max\_span$.
Output: The current set of frequent inter-transaction patterns $FP$, closed patterns $C$, and prefixed-generator $PG$.
8. Add $p$ to $FP$;
9. If $p$ is a closed pattern by Definition 12 add $p$ to $C$;
10. If $p$ is a prefixed-generator pattern by Definition 13 add $p$ to $PG$;
11. Scan $D|_{p}$ to find all locally frequent e-items $E$;
12. For each e-item $c$ in $E$ do:
   13. Append $c$ to $p$ to form a new pattern $pc$;
   14. Construct a $pc$-projected database $D|_{pc}$ by using the hash table pruning strategy to check $H$ and the ancestor node pruning strategy to check $E$;
   15. Call Project($D|_{pc}, pc, H, FP, C, PG, min\_sup, max\_span$);
16. End For

Fig. 2. The PITP-Miner algorithm.
3. Initialize $PG$ and $C$ to Prefixed-Generator and Closed accordingly;
4. For each closed inter-transaction pattern $Z \in C$
5. Let $Rules = \{}$;
6. For each $X$ and $Y$, where $X \in PG$, $X+Y = Z$, and $sup(Z) / sup(X) \geq \text{min}_\text{conf}$;
7. Let $r = X \rightarrow Y$, $sup = sup(X+Y)$ and $conf = sup(X+Y)/sup(X)$;
8. Add $(sup, conf)$ to $Rules$;
9. End For
10. $NR-Rule = NR-Rule \cup Rules$;
11. End For
12. Output $NR-Rule$

Fig. 3. NRITR-Miner: Mining non-redundant inter-transaction rules.

Our proposed algorithm (NRITR-Miner algorithm) is shown in Fig. 3. The algorithm is based on the definition of $NR-Rule$, which uses the sets of Prefixed-Generator and Closed inter-transaction patterns. The algorithm uses PITP-Miner algorithm (lines 1-3) for mining the Prefixed-Generator and Closed sets concurrently. For each closed inter-transaction pattern $Z$ in Closed, it then trying to find patterns $X$ and $Y$ in $Z$ (line 6), where $X$ contained in Prefixed-Generator and the confidence of $X$ composting $Y$ is higher than the minimum confidence. Next, we form a rule $r = X \rightarrow Y$ with the support and confidence (line 7), and add the rule to $NR-Rule$ (lines 8, 10). After iterating for all possible patterns in Closed, $NR-Rule$ will correspond to the set of non-redundant inter-transaction rules which is then output (line 12).

We measure the performance benefit of mining a set of non-redundant inter-transaction rules over mining a full-set of inter-transaction rules. For comparison purpose, Fig. 4 describes an ITR-Miner algorithm mining a full-set of inter-transaction rules, which mine the set of all frequent inter-transaction patterns. We use PITP-Miner algorithm to mine all frequent inter-transaction patterns. For each of frequent pattern $Z$, we form all rules in $Z$. Each rule is of the form $X \rightarrow Y$, where $X$ is a prefix of $Z$ and $X+Y = Z$.

**Algorithm:** ITR-Miner: Mining a full-set of Inter-Transaction Rules.

**Input:** A transaction database $DB$, a minimum support $\text{min}_\text{sup}$, and a minimum confidence $\text{min}_\text{conf}$.

**Output:** A full-set of inter-transaction rules $All-Rule$.

1. Let $FP = \{}$, $All-Rule = \{}$;
2. Use PITP-Miner to concurrently mine all frequent inter-transaction patterns with support $\geq \text{min}_\text{sup}$;
3. Initialize $FP$ to all frequent inter-transaction patterns;
4. For each frequent inter-transaction pattern $Z \in FP$
5. Let $Rules = \{}$;
6. For each $X$ and $Y$, where $X \in FP$, $X+Y = Z$, and $sup(Z) / sup(X) \geq \text{min}_\text{conf}$;
7. Let $r = X \rightarrow Y$, $sup = sup(X+Y)$ and $conf = sup(X+Y)/sup(X)$;
8. Add $(sup, conf)$ to $Rules$;
9. End For
10. $All-Rule = All-Rule \cup Rules$;
11. End For
12. Output All-Rule;

Fig. 4. ITR-Miner: Mining a full-set of inter-transaction rules.

3.5 An Example

Consider the example in Fig. 1. Assume that \( \maxspan = 1 \), \( \minsup = 3 \), and \( \minconf = 0.75 \). The first step PITP-Miner algorithm mines a complete set of frequent inter-transaction patterns \( FP \), closed patterns \( C \), and prefixed-generator patterns \( PG \), which are shown in Fig. 5, where the number after the pattern represents the pattern’s support.

<table>
<thead>
<tr>
<th>Pattern Type</th>
<th>Patterns</th>
</tr>
</thead>
</table>
| \( FP \)     | \{a(0):5, b(0):5, c(0):3, \{a(0), b(0):4, a(0), c(0):3, a(0), a(1):3, \{a(0), b(1):3, \{a(0), c(1)):3, \{b(0), a(1)):4, \{b(0), b(1):4, \{b(0), c(1)):3, \{a(0), b(0), a(1)):3, \{a(0), b(0), b(1)):3, \{a(0), b(0), c(1)):3, \{a(0), a(1), c(1)):3, \{b(0), a(1), b(1)):3, \{b(0), a(1), c(1)):3, \{a(0), b(0), a(1), c(1)):3 |}

Fig. 5. Patterns mined from PITP-Miner.

In NRITR-Miner algorithm, it uses \( PG \) and \( C \) mined from PITP-Miner to generate a set of non-redundant inter-transaction rules \( NR-Rule \). For a closed inter-transaction pattern \{b(0), a(1)):4 in \( C \), since \{b(0)):5 in \( PG \) is a prefixed-generator pattern, the algorithm generates a non-redundant inter-transaction rule \{b(0))→{a(1)):4, 0.8) in \( NR-Rule \), where the numbers after the rule represents the rule’s support and confidence. A similar procedure can be applied to \{b(0), b(1)):4, \{a(0), b(0), c(1)):3 in \( C \) and \{b(0)):5, \{a(0), b(0)):4 in \( PG \) to generate \{b(0))→{b(1)):4, 0.8), \{a(0), b(0))→{b(1)):3, 0.75), \{a(0), b(0))→{a(1), c(1)):3, 0.75) in \( NR-Rule \). Fig. 6 shows the set of non-redundant inter-transaction rules \( NR-Rule = \{b(0))→{a(1)):4, 0.8), \{b(0))→{b(1)):4, 0.8), \{a(0), b(0))→{b(1)):3, 0.75), \{a(0), b(0))→{a(1), c(1)):3, 0.75).

On the other hand, in ITR-Miner algorithm, it uses \( FP \) mined from PITP-Miner to generate a full-set of inter-transaction rules \( All-Rule \). For frequent inter-transaction patterns \{a(0), b(0), a(1)):3, \{a(0), b(0), c(1)):3 in \( FP \), since \{a(0), b(0)):4 in \( FP \) is a frequent patterns, the algorithm generates three redundant inter-transaction rules \{a(0), b(0))→{a(1)):3, 0.75), \{a(0), b(0))→{c(1)):3, 0.75), and \{a(0), b(0))→{a(1), c(1)):3, 0.75) in \( All-Rule \). A similar procedure can be applied to \{b(0), a(1)):4, \{b(0), b(1)):4, \{a(0), b(0), b(1)):3, it \( FP \) and \{b(0)):5, \{a(0), b(0)):4 in \( FP \) to generate \{b(0))→{a(1)):4, 0.8), \{b(0))→{b(1)):4, 0.8), \{a(0), b(0))→{b(1)):3, 0.75) in \( All-Rule \). Fig. 6 shows the full-set of inter-transaction rules \( All-Rule \).
4. PERFORMANCE EVALUATION

To evaluate the performance of the NRITR-Miner algorithm, we compared it with the ITR-Miner algorithm on a synthetic dataset. All experiments were conducted on an IBM Compatible PC with an Intel(R) Core(TM) i7-3520M CPU 2.9 GHz, 8 GB main memory running on Microsoft Windows 7 Enterprise. Each algorithm was implemented using Python programming language running under Python 2.7 environment.

4.1 Synthetic Data

We use a similar approach suggested by Tung et al. [16] to generate the synthetic datasets. We then conduct several experiments under various parameter settings. The default values for these parameters are listed as follows: $|D|$ is the total number of transactions in the database and its default value is 10,000; $T$ is the average length of the transactions and its default value is 10; $F$ is the number of potential frequent inter-transaction patterns and its default value is 1,000; $MT$ is the maximum length of the transactions and its default value is 200; $L$ is the average length of potentially frequent inter-transaction patterns and its default value is 5; $ML$ is the maximum length of potentially frequent inter-transaction patterns and its default value is 10; $|I|$ is the number of distinct items and its default value is 500; $S$ is the maximum span and its default value is 3. Note that the support of a pattern is defined as the fraction of megatransactions containing the pattern in the database in the experimental section.

4.2 Experiments on synthetic data

Fig. 7 illustrates the effect of minimum support on the execution time, where the support varies from 0.08% to 0.12% and the minimum confidence fixed at 50%. The result shows that as the minimum support thresholds decrease, the execution time of both algorithms increases. In Fig. 7, the NRITR-Miner runs much faster than the ITR-Miner algorithm by 20-100 times.

Fig. 8 shows the effect of the minimum support on the number of rules where the support varies from 0.08% to 0.12% and the minimum confidence fixed at 50%. In Fig. 8, the number of non-redundant inter-transaction rules mined by NRITR-Miner reduced up to 50 times than the number of full-set of inter-transaction rules mined by ITR-Miner algorithm.

In Figs. 9 and 10, we show the impact of minimum confidence on the performance of the algorithms, where the minimum confidence varies from 50% to 90% and the minimum support fixed at 0.12%. In Fig. 9, the NRITR-Miner is more efficient than the
Fig. 7. The effect of minimum support on the execution time.

Fig. 8. The effect of minimum support on the number of rules.

Fig. 9. The effect of minimum confidence on the execution time.

Fig. 10. The effect of minimum confidence on the number of rules.

Fig. 11. The effect of the number of transactions on the execution time.

Fig. 12. The comparison of inter-transaction rules mining algorithms.

Fig. 13. Gazelle: the effect of minimum support on the execution time.

Fig. 14. Gazelle: the effect of minimum support on the number of rules.
ITR-Miner algorithm up to 20 times. In Fig. 10, when minimum confidence increases, less number of rules is generated. The number of non-redundant inter-transaction rules mined by NRITR-Miner reduced ranging from 30–50 times than the number of full-set of inter-transaction rules mined by ITR-Miner algorithm.

Fig. 11 shows the effect of the number of transactions on the execution time when the number of transactions varies between 10K and 2000K. Although the execution times of the two algorithms increase almost linearly as the number of transactions increases, the NRITR-Miner algorithm runs faster than the ITR-Miner algorithm in all cases.

In this paper, we use the PITP-Miner algorithm to generate frequent inter-transaction patterns in ITR-Miner algorithm. Since the ITP-Miner [8] and FITI [16] algorithms also mine all frequent inter-transaction patterns, we can use them in ITR-Miner to generate full-set of inter-transaction rules. Fig. 12 illustrates the comparison of ITR-Miner, NRITR-Miner, ITP-Miner, FITI, and PITP-Miner algorithms for mining inter-transaction rules, where the support varies from 0.08% to 0.12% and the minimum confidence fixed at 50%. In Fig. 12, the NRITR-Miner runs faster than the ITR-Miner, the ITR-Miner runs faster than the ITP-Miner, and the ITP-Miner runs faster than the FITI algorithms. Fig. 12 also illustrates the composition of the execution times in NRITR-Miner and ITR-Miner. The result shows that the execution time of the PITP-Miner is very close to that of the NRITR-Miner. This result means the first step consumes more execution times on mining rules in NRITR-Miner algorithm. On the other hand, the result shows that the execution time of the PITP-Miner is very far away from that of the ITR-Miner. This result means the second step consumes more execution times on mining rules in ITR-Miner algorithm.

4.3 Experiments on Real Data

We use a real dataset called Gazelle to evaluate the performance of the algorithms. The dataset was obtained from the KDDCup 2000 competition and corresponds to clickstream data for the Gazelle.com web-site [6]. This dataset contains 59,601 transactions and 498 distinct items; the average transaction length is 2.5, and the maximum transaction length is 267.

We performed run time versus minimum support and the number of rules versus minimum support experiments on the Gazelle real dataset. The results, shown in Figs. 13 and 14 respectively, demonstrate that the NRITR-Miner algorithm outperforms the ITR-Miner algorithm in all cases.

In summary, since the NRITR-Miner algorithm employs the notion of non-redundant for mining inter-transaction rules, it is more efficient and scalable than the ITR-Miner algorithms in all cases. Therefore, the experimental results show that the non-redundant rule approach is a feasible and effective solution for the inter-transaction rule mining problem.

5. CONCLUSION

In this paper, we proposed and characterize a non-redundant inter-transaction rules. We study the quality inter-transaction rule sets with respect to completeness and tightness. Then, we characterize a tight and complete set of non-redundant inter-tran-
transaction rules based on compositions of patterns. For comparing purpose, we develop an algorithm named NRITR-Miner to mine the non-redundant inter-transaction rules efficiently and an algorithm named ITR-Miner to mine the full-set of inter-transaction rules. We demonstrate via experiments that the proposed NRITR-Miner algorithm is efficient and scalable, and outperform the compared ITR-Miner algorithm in all cases. The study shows large improvements in both runtime and compactness of mined non-redundant inter-transaction rules over mining a full-set of inter-transaction rules. Both runtime and the number of rules are improved and reduced up to orders of magnitudes.

Although we have shown that the NRITR-Miner algorithm can mine the non-redundant inter-transaction rules efficiently, there are still some issues to be addressed in future research. First, we can try to improve further the efficiency of the mining algorithm. We could also investigate the application of non-redundant set of inter-transaction rules in various real world application domains.

REFERENCES


Chun-Sheng Wang (王春笙) received an MBA and a Ph.D. degree in Information Management from National Taiwan University, Taiwan, in 1996 and 2007, respectively. He is currently an Assistant Professor of Information Management at the Jinwen University of Science and Technology, Taiwan. His papers have appeared in Journal of Systems and Software, Information Sciences, Data and Knowledge Engineering, Expert Systems with Applications, Journal of Network and Systems Management, etc. His current research interests include data mining, information systems, and network management.