Knowledge-Based Systems 61 (2014) 76–88

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Discovering valuable frequent patterns based on RFM analysis without customer identification information

Ya-Han Hu [*], Tzu-Wei Yeh

Department of Information Management, National Chung Cheng University, Chiayi 62102, Taiwan, ROC

ARTICLE INFO

Article history:
Received 19 September 2012
Received in revised form 14 February 2014
Accepted 15 February 2014
Available online 21 February 2014

Keywords:
Association rules
RFM analysis
Constraint-based mining
Customer identification
Utility pattern mining

ABSTRACT

RFM analysis and market basket analysis (i.e., frequent pattern mining) are two most important tasks in database marketing. Based on customers' historical purchasing behavior, RFM analysis can identify a valuable customer group, while market basket analysis can find interesting purchasing patterns. Previous studies reveal that recency, frequency and monetary (RFM) analysis and frequent pattern mining can be successfully integrated to discover valuable patterns, denoted as *RFM-customer-patterns*. However, since many retailers record transactions without collecting customer information, the RFM-customer-patterns cannot be discovered by existing approaches. Therefore, the aim of this study was to define the *RFM-pattern* and develop a novel algorithm to discover complete set of RFM-patterns that can approximate the set of RFM-customer-patterns without customer identification information. Instead of evaluating values of patterns from a customer point of view, this study directly measures pattern ratings by considering RFM features. An RFM-pattern is defined as a pattern that is not only occurs frequently, but involves a recent purchase and a higher percentage of revenue. This study also proposes a tree structure, called an *RFM-pattern-tree*, to compress and store entire transactional database, and develops a pattern growth-based algorithm, called *RFMP-growth*, to discover all the RFM-patterns in an RFM-pattern-tree. Experimental results show that the proposed approach is efficient and can effectively discover the greater part of RFM-customer-patterns.

1. Introduction

Database practitioners and researchers have recently expressed dramatic interest in database marketing [6,19,26,27,33,34,39]. The aim of database marketing is to improve the effectiveness of customer service, enhance the quality of sales, and retain valuable customers in the most profitable way. In marketing practice, businesses have begun to analyze customer databases while building their marketing plans. Database marketing is therefore becoming a powerful tool for companies to gain competitive advantages [34,37].

Recency, frequency and monetary (RFM) analysis is a powerful and well-known tool in database marketing, and is widely used to measure the value of customers based on their prior purchasing history. Hughes [26] is the first to introduce the basic concept of RFM analysis, defining valuable customers as those simultaneously having high recency, frequency, and monetary values. Recency represents the time interval between the time of the latest event (e.g. transaction) and the present one, frequency represents the number of events occurring in historical records, and monetary represents the total dollar value of payments. RFM analysis has many applications in customer segmentation, including retailing [9,10,31], banking [22], e-Commerce [12,28,30], etc.

The concept of RFM analysis has been successfully integrated into the mining process in the past few years [11,22,30,31]. Specifically, they first partition customers based on RFM analysis and then apply data mining techniques to discover patterns for a group of valuable customers (i.e., RFM-customers). In this study, we call patterns of the RFM-customers as *RFM-customer-patterns*. It is worth noting that, to discover complete set of RFM-customer-patterns, the transactions of RFM-customers must be identified through a membership card or login account prior to the beginning of the mining process. However, in real-life applications, many retailers record transactions without collecting customer identification information. Take as an example, 7-Eleven, the biggest chain of convenience stores in Taiwan, holds over four thousands branches. Since most customers check out by cash, only the items purchased in each transaction can be recorded at the point of sale (POS). In other words, the purchase information recorded in the transactional database cannot be used to identify the RFM-customer or any RFM-customer-patterns. Although some

* Corresponding author. Tel.: +886 5 2720411; fax: +886 5 2721501.
E-mail address: yahan.hu@mis.ccu.edu.tw (Y.-H. Hu).

http://dx.doi.org/10.1016/j.knosys.2014.02.009
0950-7051/© 2014 Elsevier B.V. All rights reserved.

businesses do not record customer information, this does not mean they have no RFM-customers. In this case, the RFM-customer-patterns are hidden in transactional database. Therefore, the optimal method of retrieving such patterns from transactional databases (i.e., without customer identification information) is a crucial research topic; however, no previous work has addressed this concern.

The goal of this study is to discover the set of *approximate* RFM-customer-patterns without customer identification information. Specifically, we defined the *RFM-patterns* that approximately match the relevant set of RFM-customer-patterns, which can be directly identified using transactional databases (i.e., no customer identification information). This study is the first to address how to discover RFM-customer-patterns without using customer identification information. To efficiently discover a comprehensive set of RFM-patterns, we developed a novel tree structure for storing transactional databases and a tree-based mining algorithm.

The design of the RFM-patterns is described as follows. In RFM analysis, a customer is defined as an RFM-customer if his/her transactions frequently occur in both the whole database and the recent period, and more so if the customer provides high revenue for business. Therefore, RFM-customers have higher possibilities to make transactions with higher recency and monetary values, and the frequent patterns induced by those transactions may have higher possibilities to be RFM-customer-patterns.

The basic idea of RFM-pattern is that, through the consideration of RFM value of each transaction, the RFM-customer-patterns can be highlighted in the frequent pattern mining process. According to the above observations, this study first develops a scoring-based method to evaluate the value of patterns. Specifically, the recency score is calculated based on the occurrences of a pattern in a recent period. More recent occurrences of a pattern lead to a higher recency score. The frequency score refers to the support count of the pattern. The monetary score denotes the amount of money that all customers spent on this pattern. Based on the scoring method, the RFM-patterns is defined as the frequent patterns with their recency, frequency, and monetary scores satisfying the user-specified minimum recency, frequency, and monetary thresholds, respectively.

After the definition of the RFM-patterns, this study extends the well-known pattern-growth-based methods (i.e., FP-growth) for efficiently discovering a complete set of RFM-patterns. A compact tree structure, called *RFM-pattern-tree*, is designed to compress and store all necessary information about the mining of RFM-patterns. Based on this tree, we develop an efficient pattern growth algorithm, called *RFMP-growth*, for mining complete set of RFM-patterns.

The rest of the paper is organized as follows. Section 2 reviews recent studies on RFM-related pattern mining and constraint-based pattern mining. Section 3 formally defines the problem and the RFM-patterns. Section 4 proposes the RFM-pattern-tree and Section 5 develops the RFMP-growth algorithm. Section 6 provides the preparation of data, experimental setup and evaluation results. Section 7 concludes our study.

2. Related work

2.1. RFM analysis and frequent pattern mining

RFM analysis is a powerful tool for assessing customer lifetime value. Customers can be segmented into various groups that exhibit similar RFM values [37]. Based on the segmentation results, customized marketing strategies can be developed for specific customer groups. RFM analysis has been widely used in various real-life applications, such as identifying profitable customers from bank database [22], product recommendations for valuable customers [30,31], and customer churn management [11].

Researchers have recently combined RFM analysis with frequent pattern mining techniques to develop methods of retrieving interesting customer purchase patterns. Existing research can be divided into two main categories. The first category generates patterns by directly evaluating customer values based on RFM analysis. This approach can be further subdivided into two categories: (1) customer segmentation and frequent pattern mining for each customer group [11,12,30,31]; and (2) sequential pattern mining with RFM-related constraints [9,10,25].

The first approach is a combination of both clustering and frequent pattern mining techniques. For example, Liu and Shih [31] adopts weighted RFM analysis by first applying an analytic hierarchy process (AHP) to evaluate customer value. Conventional association rule mining techniques have been conducted to build product recommender systems for different customer groups. Cheng and Chen [11] combines RFM analysis with rough set theory to retrieve rules for customer classification. Chiang [12] proposes RFMDR, an extended version of RFM analysis, to identify valuable customers and generated fuzzy association rules. Next, since sequence databases record complete customer purchasing behavior, the concept of RFM analysis can be easily integrated into sequential pattern mining process. Chen and Hu [9] proposes the CFR-PostfixSpan algorithm to consider recency, frequency, and compactness constraints in sequential pattern mining. Chen et al. [10] proposes the RFM-Apriori algorithm for sequential pattern mining with the consideration of recency, frequency, and monetary values. The results of RFM-Apriori algorithm maintain the following three properties: (1) customers must spend a certain amount of money on the patterns; (2) patterns occur at least once in a specified time period; and (3) patterns occur more frequently than a specified frequency threshold in the database. Note that the studies above are designed to analyze customer historical records, which are infeasible if customer information is unavailable.

The second category redefines frequent patterns and regards parts of RFM features as constraints in the mining process [15,23,29,44,45]. For the concept of recency, previous studies reveal that the purchasing behaviors of users may change over time, and patterns that frequently occurred in the past may rarely happen in the future. Dong and Li [15] defines *emerging patterns*, where the support of a discovered pattern must grow significantly from different time period databases. It can be assumed that the new behavior information in the database more accurately predicts future behavior than does the old behavior. Wan and An [44] further proposes *transitional patterns*. The aim of this study is to identify a *significant milestone*, a time point at which the frequency of a transitional pattern changes the most. The TP-mine algorithm can discover a complete set of transitional patterns and their significant milestones.

An emerging data mining technique, called *utility mining* [3,16,23,29,41,45], can be adopted to measure the value of a pattern from a monetary point of view. Conventional frequent pattern mining assumes that itemsets with a high occurrence are of interest to users, which totally ignores the semantic behind the items (i.e., the usefulness of frequent patterns). Therefore, utility mining attempts to measure the semantic significance of items/itemsets from a user perspective. Liu et al. [29] proposes a two-phase algorithm that holds downward closure property by using transaction-weighted utilization. The CTU-Mine algorithm proposed by Erwin et al. [16] extends the two-phase algorithm, using a pattern growth approach. Ahmed et al. [3] develops a pattern growth algorithm by using the “build once, mine many” principle; a tree structure was developed to efficiently mine utility patterns. While considering the profit that itemsets can generate for a business, the utility mining technique can find a complete set of frequent itemsets that satisfies a user-specified minimum profit threshold, i.e., high utility itemset.

Although previous studies emphasize the merits of RFM analysis in frequent pattern mining, they fail to deliberate all the RFM features at a time. For example, emergent patterns and transactional patterns consider the concepts of recency and frequency, and entirely disregard the concept of monetary. Utility mining only focuses on the profit of a frequent pattern, and therefore cannot reflect changes in user's recent purchasing behavior. In other words, high utility itemsets may be irrelevant to a decision maker if they only occurred in the past. Therefore, this study integrates the concept of RFM analysis into frequent pattern mining process. Without customer identification information, we will propose a new approach in this study which can directly identify valuable patterns by measuring the RFM value of patterns.

2.2. Pattern elicitations with constraints

Frequent pattern mining plays a crucial role in association rule discovery [1,2,7,8,20,21,24,42,43]. An itemset is considered as a frequent pattern if it satisfies a user-specified frequency constraint. Researchers have published numerous follow-up studies and applications on frequent pattern mining over the last decade. Experimental evaluations reveal that, instead of uncovering thousands of patterns by considering frequency only, incorporation of additional constraints into the mining process can yield more promising results (i.e., interesting patterns) [4,5,9,18,35,36,42].

Srikant et al. [42] first proposes a generalized association rule mining with item constraint and taxonomy. To meet user's interest, this approach discovers frequent patterns containing specific items at multiple levels of abstraction. Ng et al. [35] proposes the CAP algorithm to include syntactic constraints (e.g. *min*, *max*, *count*, and *sum*, etc.) to decrease the computation cost of finding frequent patterns. Some researchers [4,36] have recently conducted systematic studies or thorough experimental evaluations on constraint-based frequent pattern mining. Pei et al. [36] specifies seven categories of constraint-based sequential pattern mining, including both temporal and non-temporal constraints. Non-temporal constraints are also applicable to frequent pattern mining. The *item* constraint and *super-pattern* constraint specify that the discovered patterns should contain or not contain the specific subset/superset of items, respectively. The *length* constraint specifies the threshold of pattern length. The *regular expression* constraint satisfies user-specified templates. The *aggregate* constraint considers pattern aggregation information in the mining process. Bonchi and Lucchese [4] divides constraint-based mining into five categories, including *antimonotone*, *monotone*, *succinct*, *convertible*, and *inconvertible*, based on the interactions between constraints and mining algorithms.

Several variations on frequent patterns have been developed [7,8,21]. Burdick et al. [7] proposes the concept of *maximal frequent itemset* (MFI), which is defined as a pattern that exhibits no superset patterns in the set of MFI. A depth-first search strategy has been developed to efficiently determine the MFI based on the itemset lattice. Han et al. [21] proposes top-k frequent closed itemset mining, which can be used to efficiently generate association rules without mining a complete set of frequent patterns. Using a market basket analysis, Chen et al. [8] considers the shelf information involved in frequent pattern mining processes. Spatial constraints are included to determine specific spatial association rules for use in managing supermarket shelves.

Researchers have begun to address high-utility pattern mining by using various constraints [14,32,40]. Chu et al. [14] proposes mining temporal high-utility patterns with sliding window constraint from data streams. In other words, temporal high-utility patterns can be located by removing old transactions and including new transactions in the mining process. Shie et al. [40] extends the concept of maximal patterns to mine high-utility patterns. A high-utility pattern is considered to be maximal if it is not a subset of any other high-utility patterns [32]; therefore, the maximal high-utility pattern is a compact representation of high-utility patterns that facilitates efficient mining.

Pattern elicitation with constraints has been recognized as an effective method of discovering both valuable frequent and utility patterns. While constraints can be pushed deep into the mining algorithm, it is likely to achieve efficiency due to the limited search space. This study defines RFM features as constraints and designs a novel frequent pattern search algorithm for efficiently discovering a complete set of RFM-patterns.

3. Problem definition

This section first defines the problem of mining RFM-patterns in transactional databases. The transactions used in traditional association rule mining often record itemsets only. Since this study measures the RFM values of patterns, we represent a transaction in the following way for storing necessary information.

Let I be a set of all items in database DB, and let a transaction t be represented as $< time_t, (a_1, q_{a_1})(a_2, q_{a_2}) \ldots (a_m, q_{a_m}) >$, where $time_t$ stands for the transaction time when transaction t occurs, the m pairs, (a_k, q_{a_k}), denote m different items in t, and the purchased quantity for item a_k is q_{a_k} $(1 \leqslant k \leqslant m$ and $a_k \in I)$. For an *itemset* A $(A \subseteq I)$, A is *contained* in t if all items in A also occur in t.

Definition 1 (*Frequency score and F-pattern*). The *frequency score* of itemset A in database DB, denoted as $FScore_{DB}(A)$, is defined as the number of transactions containing itemset A.

Given a user-specified minimum frequency threshold α, itemset A is called an *F-pattern* if $FScore_{DB}(A)$ is no less than α multiplied by the total number of transactions in DB, denoted as $|DB|$. That is, itemset A is an *F-pattern* if $FScore_{DB}(A) \geqslant |DB| \times \alpha$.

Definition 2 (*Monetary score and M-pattern*). Let $p(a_k)$ denote the unit price of item a_k. Assume that itemset A is contained in transaction t. The *monetary score* of itemset A in t, denoted as $MScore(A, t)$, is then defined as $MScore(A, t) = \sum_{a_k \in A} p(a_k) \times q_{a_k}$. Given a transaction database DB, the monetary score of itemset A in DB, denoted as $MScore_{DB}(A)$, represents the sum of all monetary scores of each transaction containing itemset A; that is, $MScore_{DB}(A) = \sum_{t_s \in DB} MScore(A, t_s)$, where t_s $(1 \leqslant s \leqslant |DB|)$ denotes the s-th transaction in DB. For an itemset A in DB, itemset A is called an *M-pattern* if its $MScore_{DB}(A)$ is no less than a user-specified minimum monetary threshold β; that is, $MScore_{DB}(A) \geqslant \beta$.

As mentioned in Section 1, the more recent occurrence of a pattern, the higher its recency. Without loss of generality, the recency score of a pattern in a transaction is in reverse proportional to the time gap between the current time and the transaction time of the transaction. Many expressions can denote the reverse proportional relationship of these two values. Here, this study adopts the formula defined by Russ et al. [38].

Definition 3 (*Recency score and R-pattern*). Assume that itemset A is contained in t, the recency score of itemset A in t, denoted by $RScore(A, t)$, is defined as $RScore(A, t) = (1 - \delta)^{time_{current} - time_t}$, where δ is a user-specified *decay speed* $(\delta \in (0, 1))$ and $time_{current}$ denotes the current time. For transaction database DB, the recency score of itemset A in DB, denoted as $RScore_{DB}(A)$, is defined as the sum of all recency scores of each transaction containing itemset A, i.e., $RScore_{DB}(A) = \sum_{t_s \in DB} RScore(A, t_s)$. For an itemset A in DB, itemset A is called an *R-pattern* if $RScore_{DB}(A)$ is no less than a user-specified recency threshold γ, i.e., $RScore_{DB}(A) \geqslant \gamma$.

TID	Transaction
t_1	<0, (A, 1)(B, 2)(F, 1)>
t_2	<18, (A, 1)(B, 1)(F, 2)>
t_3	<22, (A, 2)(C, 1)>
t_4	<39, (A, 2)(B, 1)(E, 3)>
t_5	<45, (C, 1)(E, 7)>
t_6	<68, (A, 1)(E, 5)>
t_7	<90, (C, 2)(D, 1)(E, 10)>
t_8	<100, (A, 1)(C, 1)(F, 5)>
t_9	<109, (A, 2)(E, 8)(F, 10)(G, 1)>
t_{10}	<115, (A, 2)(D, 1)(E, 10)>

(a) Transaction database.

Item	Price
A	10
B	150
C	25
D	45
E	7
F	2
G	80

(b) Unit price list of items.

Fig. 1. An example of transactional database and unit price list of items.

Definition 4 (*RFM-pattern*). Given a transaction database *DB* and three user-specified thresholds α, β, and γ, an itemset *A* is called an RFM-pattern if it satisfies all the following conditions: (1) $FScore_{DB}(A) \geqslant |DB| \times \alpha$ (i.e., F-pattern), (2) $MScore_{DB}(A) \geqslant \beta$ (i.e., M-pattern), and (3) $RScore_{DB}(A) \geqslant \gamma$ (i.e., R-pattern).

Example 1. Consider the transaction database *DB* shown in Fig. 1(a) and the unit prices on the list of items shown in Fig. 1(b). Given an itemset {CE}, the *FScore* of itemset {CE} is 2 because it is contained in transactions t_5 and t_7. If α is set at 10%, then itemset {CE} is called an *F*-pattern because $FScore_{DB}(\{CE\}) \geqslant 10 \times 10\% = 1$. The *MScore* of itemset {CE} in *DB* is 194; in other words, the $MScore_{DB}(\{CE\}) = MScore(\{CE\}, t_5) + MScore(\{CE\}, t_7) = 194$. If β is set at 150, then {CE} is an M-pattern. Moreover, assume that $time_{current} = 115$ and $\delta = 0.01$, the recency score of itemset {CE} in *DB* is 1.273 (i.e., $RScore_{DB}(\{CE\}) = RScore(\{CE\}, t_5) + RScore(\{CE\}, t_{27}) = 0.495 + 0.778 = 1.273$). If γ is set at 0.95, then itemset {CE} is an R-pattern. Thus, itemset {CE} is an RFM-pattern.

Using the definitions, we formally state our problem as follows: for a given transaction database *DB*, the goal of this study is to discover a complete set of RFM-patterns. Although the concept of RFM-patterns is straightforward, it is difficult to consider the three constraints in the mining process simultaneously. The main problem is that monetary score does not have the *downward closure property*; in other words, a superset of a non-RFM-pattern may be an RFM-pattern.

Example 2. Consider the transaction database *DB* and the unit price list of items in Fig. 1. The itemset {F} is not an RFM-pattern because $MScore_{DB}(\{F\}) = MScore(\{F\}, t_1) + MScore(\{F\}, t_2) + MScore(\{F\}, t_8) + MScore(\{F\}, t_9) = 36$, which is smaller than the monetary threshold (i.e., $\beta = 100$). However, itemset {FG}, a superset of itemset {F}, is an RFM-pattern because its *RScore*, *FScore*, and *MScore* satisfy the corresponding thresholds (i.e., $RScore_{DB}(\{FG\}) = 0.94 \geqslant 0.9$, $FScore_{DB}(\{FG\}) = 1 \geqslant 10 \times 10\%$, and $MScore_{DB}(\{FG\}) = MScore(\{FG\}, t_9) = 100 \geqslant 100$).

This example shows that the *MScore* of an itemset's superset may be higher than that of the itemset. Without the downward closure property, there can be a huge amount of item combinations in the mining process, resulting in unbearable computation time and memory usage. To solve this problem, this study adopts the concept of transaction-weighted utilization [29] to effectively prune the search space:

Definition 5. Following Definition 2, assume that itemset *A* is contained in transaction *t*. The *transaction amount* of itemset *A* in transaction *t*, denoted as *ta*(*A*,*t*), is defined as the total amount of money in transaction *t*, i.e., $ta(A, t) = \sum_{a_k \in t} p(a_k) \times q_{a_k}$. The *total transaction amount* of the itemset *A* in *DB*, denoted by $tta_{DB}(A)$, is defined as the sum of transaction amounts in all transactions containing itemset *A*; that is, $tta_{DB}(A) = \sum_{t_s \in DB} ta(A, t_s)$.

Definition 6 (*RFT-pattern*). Given a transaction database *DB* and three user-specified thresholds α, β, and γ. For an itemset *A* in *DB*, *A* is called an *RFT-pattern* if it satisfies all the following conditions: (1) $FScore_{DB}(A) \geqslant |DB| \times \alpha$, (2) $tta_{DB}(A) \geqslant \beta$, and (3) $RScore_{DB}(A) \geqslant \gamma$.

Example 3. Consider Example 1, in which itemset {CE} is an R-pattern and F-pattern. Suppose the monetary threshold is set at 100. Since only transaction t_5 and t_7 contain itemset {CE}, the total transaction amount of itemset {CE} is 239 (i.e., $tta_{DB}(\{CE\}) = ta(t_5) + ta(t_7) = 239$), which exceeds the threshold. Thus, itemset {CE} is an *RFT*-pattern.

Lemma 1. *Let SRFM and SRFT denote the set of all RFM-patterns and all RFT-patterns in DB. Then SRFM is a subset of SRFT (SRFM $\subseteq$ SRFT).*

Proof. Assume that itemset *A* is contained in *t* (i.e., $A \subseteq t$). Based on Definitions 2 and 5, $MScore(A, t) = \sum_{a_k \in A} p(a_k) \times q_{a_k}$ and $ta(A, t) = \sum_{a_k \in t} p(a_k) \times q_{a_k}$. Since $\sum_{a_k \in t} p(a_k) \times q_{a_k} \geqslant \sum_{a_k \in A} p(a_k) \times q_{a_k}$, we have $ta(A, t) \geqslant MScore(A, t)$. Considering a transaction database *DB*, we can get $tta_{DB}(A) = \sum_{t_s \in DB} ta(A, t_s) \geqslant MScore_{DB}(A) = \sum_{t_s \in DB} MScore(A, t_s)$. Based on Definition 4, if itemset *A* is an RFM-pattern ($A \in SRFM$), then it must satisfy the following conditions: (1) $FScore_{DB}(A) \geqslant |DB| \times \alpha$, (2) $MScore_{DB}(A) \geqslant \beta$, and (3) $RScore_{DB}(A) \geqslant \gamma$. Because $tta_{DB}(A) \geqslant MScore_{DB}(A) \geqslant \beta$, according to Definition 6, *A* is also an RFT-pattern ($A \in SRFT$). Accordingly, $SRFM \subseteq SRFT$. □

Lemma 2 (*Downward closure property*). *Any subset of an RFT-pattern must also be an RFT-pattern.*

Proof. Assume that itemsets *A* and *A*′ are both R-patterns and F-patterns and *A*′ is a subset of *A*. Let T_A be the set of transactions containing *A* and let $T_{A'}$ be the set of transactions containing *A*′. Since $A' \subset A$, $T_{A'}$ must be a superset of T_A. Based on Definition 5, $tta_{DB}(A') \geqslant tta_{DB}(A)$. If *A* satisfies the monetary threshold β (i.e., *A* is an RFT-pattern), then *A*′ must also be an RFT-pattern because $tta_{DB}(A') \geqslant tta_{DB}(A) \geqslant \beta$. □

Based on Lemmas 1 and 2, the proposed approach can first utilize the downward closure property to reduce the search space of extracting RFT-patterns and then discover a complete set of RFM-patterns from the set of RFT-patterns.

RFM-patterns are a generalization of patterns that have been defined in previous studies [1–3,13,15–17,23]. According to the aforementioned definitions, conventional frequent pattern mining [1,2,17] only evaluates the *FScore* (i.e., F-pattern); emerging pattern mining [15] highlights the importance of the *RScore* (i.e., R-pattern); and utility pattern mining [3,13,16,17,23] considers the MScore (i.e., M-pattern). In other words, these patterns can be easily retrieved by adjusting the three thresholds. For example, the complete set of utility patterns can be determined by setting both the minimum recency and frequency thresholds to zero. This enables decision makers to flexibly determine the valuable patterns.

4. RFM-pattern-tree: design and construction

This section introduces a novel tree structure, called RFM-pattern-tree, for storing compressed and crucial information about RFT-patterns. The RFM-pattern-tree is an extended version of the well-known FP-tree structure [20].

Definition 7. An RFM-pattern-tree is a tree structure defined as follows.

(1) It consists of one root node labeled as "*null*", a set of item subtrees as the children of the root, and an *RFM-header-table* that includes all 1-RFT-patterns (i.e., RFT-patterns with only one item in it).
(2) Each node in the RFM-pattern-tree consists of seven fields: *item-name*, $RScore_{DB}$, $FScore_{DB}$, tta_{DB}, *parent-link*, *node-link*, and *child-links*. The *item-name* field stores the name of the item this node represents. The $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} fields store the values of recency score, frequency score, and total transaction amount, respectively, represented by the portion of the path reaching this node. The *parent-link* field links to the parent node; *node-link* links to the successive node in the RFM-pattern-tree carrying the same *item-name*; and *child-link* stores a list of links to all child nodes.
(3) Each entry in RFM-header-table consists of five fields, namely, *item-name*, $RScore_{DB}$, $FScore_{DB}$, tta_{DB}, and *node-link*. The *item-name* field stores the name of the item this entry represents. The fields $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} store the recency score, frequency score, and total transaction amount of this item, respectively. Finally, the *node-link* fields point to the first node of a link in the RFM-pattern-tree carrying the same *item-name* of this entry.
(4) All the items in RFM-header-table are sorted in the non-increasing order in terms of their $FScore_{DB}$.

Based on Definition 7, this study proposes the RFM-pattern-tree construction algorithm depicted in Fig. 2.

To build the RFM-header-table, the function *header_gen*() first scans the database once to collect $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of each item. If an item does not exist in the RFM-header-table, a new entry is created. The algorithm then increments the information recorded in this entry; that is, the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB}. After scanning the database, the algorithm prunes item entries if one of their $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} values does not satisfy the corresponding thresholds. The remaining item entries are a complete set of 1-RFT-patterns. The RFM-header-table is then sorted in non-ascending order based on $FScore_{DB}$.

Example 4. Consider the transaction database *DB* and the unit price list of items in Fig. 1. Assume that α, β, γ, and δ are set at 10%, 100%, 0.95%, and 0.01%, respectively, and that $t_{current}$ is 115. For the first transaction T_1, $\langle 0,(A,1)(B,2)(F,1)\rangle$, add three new entries with their *item-name* being items A, B, and F into the RFM-header-table since the table has no entry equal to those items. The values of $RScore_{DB}(T_1) = (1-0.01)^{115-0} \approx 0.315$, $FScore_{DB}(T_1) = 1$, and $tta_{DB}(T_1) = 10 \times 1 + 150 \times 2 + 2 \times 1 = 312$ are computed and stored in the entries of item A, B, and F in the RFM-header-table, respectively. Fig. 3(a) shows the RFM-header-table after scanning T_1. For the second transaction T_2, $\langle 18,(A,1)(B,1)(F,2)\rangle$, since items A, B, and F already exist in the header table, it is only necesary to update the entries for items A, B, and F with $RScore_{DB}(T_2) \approx 0.377$, $FScore_{DB}(T_2) = 1$, and $tta_{DB}(T_2) = 164$. Fig. 3(b) shows the RFM-header-table after scanning T_2. For transaction T_3, $\langle 22,(A,2)(C,1)\rangle$, its $RScore_{DB}(T_3) \approx 0.393$, $FScore_{DB}(T_3) = 1$, and $tta_{DB}(T_3) = 45$. Since item A already exists in the RFM-header-table, the algorithm updates the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} fields of the entry of item A in the table. Since item C is not in the RFM-header-table, the algorithm creates a new entry with the *item-name* being item C and enters the values, 0.393, 1, and 45, as the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of the new entry, respectively. Fig. 3(c) shows the RFM-header-table after scanning T_3. The remaining transactions can be processed in the same way. After scanning all transactions, the algorithm prunes any entries not satisfying the recency, frequency, or monetary constraints (i.e., α, β, γ), and then sorts the remaining entries in non-increasing order by $FScore_{DB}$. Fig. 3(d) shows the final RFM-header-table.

After building the RFM-header-table, the next step is to construct the RFM-pattern-tree through a second database scan. A root node labeled as *null* is created first. For each transaction, the algorithm first prunes items not in the RFM-header-table since these items do not exist in the set of 1-RFT-patterns. The remaining itemsets are then sorted based on the order of entries in the RFM-header-table. Let the sorted itemset be $[p|P]$, where p is the first item and P is the remaining item list. The algorithm starts to build the tree node by recursively calling *insert_node*() procedure.

The *insert_node*($[p|P]$, R, *Rscore*, *tta*) procedure is performed as follows. If a node R has a child, say N, such that $p = N.item\text{-}name$, then increment $N.RScore_{DB}$, $N.FScore_{DB}$, and $N.tta_{DB}$ by *Rscore*, 1, and *tta*, respectively. If R does not have a child satisfying $p = N.item\text{-}name$, then a new child of R, say N, is created with $N.RScore_{DB} = Rscore$, $N.FScore_{DB} = 1$, and $N.tta_{DB} = tta$. N's *parent-link* is linked to R, and N is linked to the tail of the *node-link* with the same *item-name* via the node-link structure. If the remaining item list P is not empty, the algorithm further divides P into $[p'|P']$, and then recursively calls the Procedure *insert_node*($[p'|P']$, R, *Rscore*, *tta*). The following example illustrates the RFM-pattern-tree construction process.

Example 5. Following Example 4, the RFM-header-table is built in Fig. 3(d). A root node labeled as "*null*" is created first. For the first transaction T_1, $\langle 0,(A,1)(B,2)(F,1)\rangle$, the algorithm first sorts the items in T_1 according to the order of items in RFM-header-table (i.e., $\langle 0,(A,1)(B,2)(F,1)\rangle$) and then calculates $RScore(T_1) = 0.315$, $FScore(T_1) = 1$, and $tta(T_1) = 312$. Since root node R does not contain any child and T_1 has three different items, new nodes A, F, and B, are created by recursively calling *insert_node*([A|FB], *root*, 0.315, 312), *insert_node*([F|B], A, 0.315, 312), and *insert_node*([B|*null*], B, 0.315, 312) to form a path of the RFM-pattern-tree (as Fig. 4(a) shows). Note that while a new node is created, all of its scores and links must be built up immediately. The $RScore_{DB}$, $FScore_{DB}$ and tta_{DB} values for nodes A, F, and B are initialized as 0.315, 1, and 312, respectively. Node A's *parent-link* links to *root*, F's *parent-link* links to A, and B's p*arent-link* links to F. Entries A, F, and B, in the RFM-header-table link to the node with the same *item-name* via node-link structure. For ease of discussion, the following discussion omits the details of building a parent-link and a node-link. The next ordered transaction T_2, $\langle 18,(A,1)(F,2)(B,1)>\rangle$, shares the same prefix (A)(F)(B) as the existing path of

```
Algorithm: Construction of RFM-pattern-tree;
Input:   $DB$; // the transaction database;
         $\alpha$ ; // minimum frequency threshold;
         $\beta$; // minimum monetary threshold ;
         $\gamma$; // minimum recency threshold ;
         $\delta$; // decay speed;
Output: the RFM-pattern-tree;
Method:
  $header = header\_gen()$;
  create a root node $R$, label it as "$null$" ;
  for each transaction $T_s$ in $DB$ do {
    remove items in $T_s$ not contained in $header$;
    sort the remaining items in $T_s$ according to the order of items in $header$;
    calculate the values of $RScore(T_s, T_s)$ and $tta(T_s)$;
    divide the sorted items of $T_s$ as $[p|P]$, where $p$ is the first item and $P$ is the remaining items;
    call $insert\_node([p|P], R, RScore(T_s, T_s), tta(T_s))$;
  }

Procedure: header_gen()
   for each transaction $T_s$ in $DB$ do {
     for each item $a_k \in T_s$ do {
       if item $a_k$ is not contained in $header$ then
         add a new entry with $item\text{-}name = a_k$ into $header$;
       update $RScore_{DB}$, $FScore_{DB}$, and $tta_{DB}$ of entry $a_k$ in $header$; } }
   remove entries in $header$ not satisfying the thresholds $\alpha$, $\beta$, or $\gamma$;
   sort entries in $header$ according to $FScore_{DB}(a_k)$ in non-increasing order;

Procedure: insert_node( $[p|P]$, $R$, $RScore$, $tta$)
  if node $R$ has a child node, say $N$, such that $N.item\text{-}name = p$ then
      $N.RScore_{DB}$ +=$RScore$ , $N.FScore_{DB}$++,   $N.tta_{DB}$+=$tta$;
  else {
     create a new node $N$ with $N.item\text{-}name = p$, $N.RScore_{DB} = RScore$, $N.FScore_{DB} = 1$, and $N.tta_{DB} =$
     $tta$, $N.parent\text{-}link = R$;
    the tail of the $node\text{-}link$ starting from the header with $item\text{-}name$=$N.item\text{-}name$ links to $N$; }
  if $P$ is not empty then {
     divide $P$ as $[p'|P']$ and call $insert\_node([p'|P'], N, RScore, tta)$; }
```

Fig. 2. RFM-pattern-tree construction algorithm.

item-name	$RScore_{DB}$	$FScore_{DB}$	tta_{DB}	link
A	0.315	1	312	
B	0.315	1	312	
F	0.315	1	312	

(a) after scaning transaction T_1.

item-name	$RScore_{DB}$	$FScore_{DB}$	tta_{DB}	link
A	0.692	2	476	
B	0.692	2	476	
F	0.692	2	476	

(b) after scaning transaction T_2.

item-name	$RScore_{DB}$	$FScore_{DB}$	tta_{DB}	link
A	1.085	3	521	
B	0.692	2	476	
F	0.692	2	476	
C	0.393	1	45	

(c) after scaning transaction T_3.

item-name	$RScore_{DB}$	$FScore_{DB}$	tta_{DB}	link
A	4.976	8	1059	
E	4.304	6	768	
C	2.525	4	329	
F	2.494	4	679	
B	1.158	3	667	
D	1.778	2	300	

(d) The final RFM-header-table.

Fig. 3. An example of the construction of RFM-header table.

RFM-pattern-tree. The algorithm directly increases the corresponded scores of nodes A, F, and B by $RScore(T_2) = 0.377$, $FScore(T_2) = 1$, and $tta(T_2) = 164$, respectively (as Fig. 4(b) shows). The *RScore*, *FScore*, and *tta* values for the third ordered transaction T_3, $\langle 22, (A,2)(C,1)\rangle$, are 0.393, 1, and 45, respectively. Since T_3 shares the same prefix (A) with the existing path of RFM-pattern-tree, the algorithm first increments the correspond scores of node A and then calls $insert\_node([C|null], A, 0.393, 45)$. For the remaining item C, since node A does not have a child with $item\text{-}name = C$, the algorithm creates a new node C as a child of node A, and set $RScore_{DB}$, $FScore_{DB}$ and tta_{DB} of C as 0.393, 1, 45, respectively (as Fig. 4(c) shows). Following the same steps in the remaining transactions, Fig. 5 shows the complete RFM-pattern-tree.

5. The RFMP-growth algorithm

This section proposes the RFMP-growth algorithm for mining a complete set of RFM-patterns. As mentioned in Section 3, the

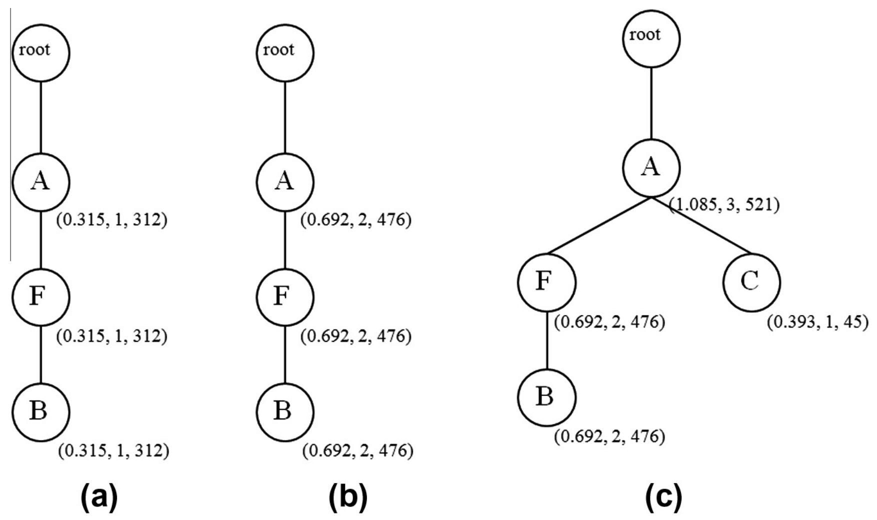


Fig. 4. Examples of inserting transaction T_1, T_2, and T_3 into RFM-pattern-tree.

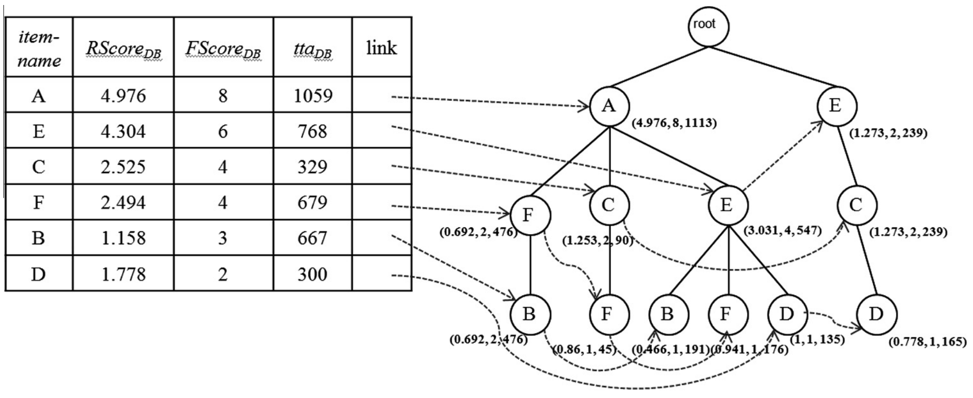
RFT-patterns hold the downward closure property while RFM-patterns do not. In addition, the set of RFT-patterns includes complete set of RFM-patterns. To efficiently prune the search space, this study divides the RFMP-growth algorithm into two phases. First, RFMP-growth retrieves all RFT-patterns from the RFM-pattern-tree. After that, in the second phase, the algorithm scans the database again to collect the monetary score of each RFT-pattern, and gets the complete set of RFM-patterns.

The first phase of RFMP-growth is similar to the FP-growth algorithm [20]. It adopts a divide-and-conquer paradigm that partitions the search space into several sub-search spaces. For example, considering the RFM-pattern-tree shown in Fig. 5, the search space is partitioned into six sub-search spaces: (1) itemsets having D; (2) itemsets having B but no D; (3) itemsets having F but no B, D; (4) itemsets having C but no F, B, D; (5) itemsets having E but no C, F, B, D; and (6) itemsets having A only. These sub-search spaces are *conditional pattern bases*. Based on the design of RFMP-growth, all the RFT-patterns with postfix equal to a_k, denoted as a_k's conditional RFT-patterns, can be discovered in a_k's conditional pattern bases. The collection of all a_k's conditional RFT-patterns forms the complete set of RFT-patterns.

In RFMP-growth, two important properties defined by Han et al. [20] hold in the construction of a_k's conditional pattern base: the *node-link property* and the *prefix path property*. The node-link property clarifies the completeness of conditional pattern base; that is, all possible a_k's conditional RFT-patterns can be retrieved through a_k's node-link. Let $node_{a_k}$ denote the complete set of nodes with *item-name* equal to a_k in the RFM-pattern-tree, follow a_k's node-link starting from RFM-header-table, the algorithm can traverse $node_{a_k}$ and then find all paths from root to all nodes in $node_{a_k}$. Hence, all the transactions information containing a_k can be obtained through these paths and the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of itemsets containing a_k can be calculated without accessing additional information. The prefix path property explains that while building the conditional pattern base by the paths from root to all nodes in $node_{a_k}$, only the prefix of the path (i.e., the path after removing nodes a_k) needs to be considered, and the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of every node in the prefix carry the same value as node a_k.

Fig. 6 shows the proposed RFMP-growth algorithm. The following discussion describes the first step of RFMP-growth. In the beginning of the RFMP-growth, we first call tree-growth subroutine with two parameters, the RFM-pattern-tree RT and the postfix itemset μ, to obtain a complete set of RFT-patterns. The parameter μ shows that RT is μ's conditional RFM-pattern-tree and the subroutine proceeds to discover μ's conditional RFT-patterns.

While calling $tree-growth(RT, \mu)$, the subroutine first collects the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of each item a_k in the



item-name	$RScore_{DB}$	$FScore_{DB}$	tta_{DB}	link
A	4.976	8	1059	
E	4.304	6	768	
C	2.525	4	329	
F	2.494	4	679	
B	1.158	3	667	
D	1.778	2	300	

Fig. 5. The complete RFM-pattern-tree.

RFM-header-table *header*. Note that the above information of item a_k represents the corresponding information of itemset $a_k \cup \mu$, say v, since each item a_k in RT co-occurs with itemset μ. If the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of item a_k are no less than the recency, frequency, and monetary requirements, itemset v is then outputted as an RFT-pattern. Based on the downward closure property, the supersets of itemset v may be RFT-patterns. This makes it possible to construct v's conditional pattern base and v's conditional RFM-pattern-tree, say RT_v, and then recursively call the *tree-growth* subroutine with two new parameters, RT_v and v, to find RFT-patterns containing itemset v.

Example 6. Fig. 5 shows the complete RFM-pattern-tree RT based on Example 6. The algorithm starts with calling $tree-growth(RT, null)$. Here, we show the tree growth procedure for item D. The RFM-header-table records the $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of itemset $D \cup \phi$ (i.e., the postfix itemset is *null* so far), where the $RScore_{DB}(D) = 1.778$, $FScore_{DB}(D) = 2$, and $tta_{DB}(D) = 300$. Since item D satisfies recency, frequency, and monetary thresholds, it is outputted as a 1-RFT-pattern. The subroutine then starts to mine a complete set of item D's conditional RFT-patterns. Two paths can be found through D's node-link: {(A:4.976, 8, 1113)(E:3.031, 4, 547)(D:1, 1, 135)} and {(E:1.273, 2, 239)(C:1.273, 2, 239)(D:0.778, 1, 165)}. According to the prefix path property, item D's conditional pattern base includes {(A:1, 1, 135)(E:1, 1, 135)} and {(E:0.778, 1, 165)(C:0.778, 1, 165)} (as Fig. 7 shows) by (1) removing item D; and (2) changing the information of the remaining nodes to node D's $RScore_{DB}$, $FScore_{DB}$, and tta_{DB}. The subroutine proceeds to generate D's conditional RFM-pattern-tree RT_D (as Fig. 8 shows) by tree construction algorithm and then calls $tree-growth(RT_D, D)$. Following the same procedure for RT_D, the algorithm retrieves $RScore_{DB}$, $FScore_{DB}$, and tta_{DB} of itemset $A \cup D$ from RT_D's RFM-header-table and outputs AD as a 2-RFT-pattern. The tree-growth procedure terminates when no item exists in the RFM-header-table. In this example, all the D's conditional RFT-patterns will be outputted, including {D}, {AD}, {EAD}, and {ED}.

After discovering all RFT-patterns from the RFM-pattern-tree, the second phase is to identify a complete set of RFM-patterns from RFT-patterns through another database scan. To increase efficiency, this study uses a hash-tree structure to efficiently collect the $MScore_{DB}$ of each RFT-pattern. This is a well-known support counting method that has been widely used in Apriori-based algorithms. For details, please refer to [1].

6. Experimental study

This section provides numerical analysis to empirically evaluate the RFM-patterns and the proposed algorithm. Two pattern-growth-based algorithms, including the FP-growth and the RFMP-growth algorithms, were implemented in Java language and tested on a DualCore Pentium E2180-2.0 GHz Windows XP system with 2 gigabytes of main memory.

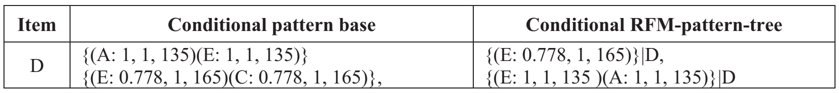
6.1. Data

The empirical evaluations in this study consider both real-life and synthetic datasets. Two real-life datasets, called SC-POS-all and Foodmart, are investigated. The SC-POS-all database records the sales data of a chain supermarket from twenty branches in Taiwan from 2001/12/26 to 2002/11/27. The Foodmart transaction database provided by Microsoft SQL server 2000 consists of sales records from 24 stores from year 1997 to 1998. A series of data-preprocessing tasks were performed, such as removing items not sold in all branches and deleting transactions with no member number. Each row in both two datasets contains transaction time, customer ID, item ID, item quantity, and item price. The final SC-POS-all dataset contained 263,082 transactions and 12,505 items while the final Foodmart dataset contained 251,395 transactions and 1560 items.

Nine synthetic datasets were generated by the IBM Quest data generator. Table 1 lists the parameter definitions and Table 2 shows the parameter settings used in data generation. Some parameters in the simulation were constant: $|L|$=10,000 and the correlation level = 0.5. Since the proposed approach considers detailed transaction information, such as item price, purchase quantity, and transaction time, the data generator produced all necessary information in the following way. First, to reflect the fact that item prices in a retailing store often have skewed distribution, the price of each item was drawn from a log-normal distribution with mean and standard deviation equal to 5 and 1.5, respectively.

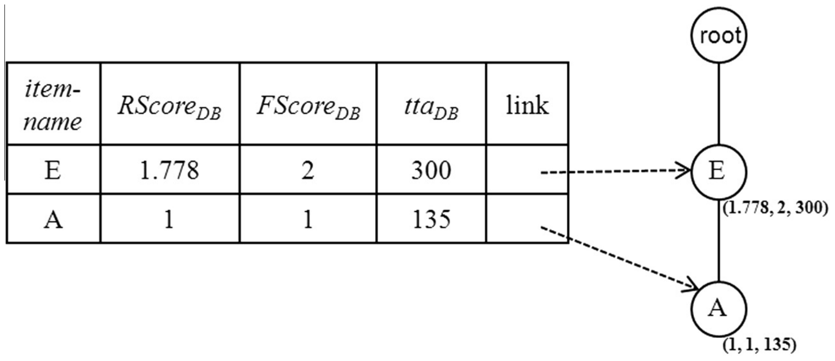
```
Algorithm: RFMP-growth algorithm
Input: $RT$; // the RFM-pattern-tree;
        $\alpha$ ; // minimum frequency threshold;
        $\beta$; // minimum monetary threshold ;
        $\gamma$; // minimum recency threshold ;
Output: the RFM-patterns
Method:
  $RFT\text{-}patterns$ = tree-growth($RT$, $null$);
  scan the database once to check the $MScore_{DB}$ of each RFT-pattern in $RFT\text{-}patterns$;
  output all of the RFM-patterns with $MScore_{DB} \geq \beta$ from $RFT\text{-}patterns$;

Procedure: tree-growth
Input: the RFM-pattern-tree, $RT$; postfix itemset, $\mu$;
Output: the RFT-patterns
Method:
  for each item $a_k$ in the RFM-header table $header$ do {
    generate pattern $v = a_k \cup \mu$ with $RScore_{DB}(v) = RScore_{DB}(a_k)$, $FScore_{DB}(v) = FScore_{DB}(a_k)$,
    and $tta_{DB}(v) = tta_{DB}(a_k)$
    if $RScore_{DB}(v)$, $FScore_{DB}(v)$, $tta_{DB}(v)$ satisfy $\alpha$, $\beta$, $\gamma$, respectively, then{
      output $v$ as an RFT-pattern;
      construct $v$'s conditional pattern base and $v$'s conditional RFM-pattern-tree $RT_v$;
      if $RT_v \neq \emptyset$ then call tree-growth($RT_v$, $v$);
    }
  }
```

Fig. 6. The RFMP-growth algorithm.



Item	Conditional pattern base	Conditional RFM-pattern-tree
D	{(A: 1, 1, 135)(E: 1, 1, 135)} {(E: 0.778, 1, 165)(C: 0.778, 1, 165)},	{(E: 0.778, 1, 165)}|D, {(E: 1, 1, 135)(A: 1, 1, 135)}|D

Fig. 7. The conditional pattern base and conditional RFM-pattern-tree of item D.



item-name	$RScore_{DB}$	$FScore_{DB}$	tta_{DB}	link
E	1.778	2	300	
A	1	1	135	

Fig. 8. The item D's conditional RFM-pattern-tree.

Table 1
Parameters.

$\|D\|$	Number of transactions
$\|T\|$	Average size of the transactions
$\|I\|$	Average size of the maximal potentially large itemsets
$\|L\|$	Number of maximal potentially large itemsets
N	Number of items

Table 2
Parameter settings of synthetic datasets.

Name	$\|D\|$ (K)	$\|I\|$	N	$\|T\|$
SYN-1	100	4	2000	25
SYN-2	150	4	2000	25
SYN-3	200	4	2000	25
SYN-4	100	8	2000	25
SYN-5	100	12	2000	25
SYN-6	100	4	1000	25
SYN-7	100	4	3000	25
SYN-8	100	4	2000	15
SYN-9	100	4	2000	5

Second, for each item a_k, a value q_{a_k} was drawn from a Poisson distribution with mean equal to 1. The drawn value q_{a_k} represents the average purchased quantity of a_k in all transactions. The purchase quantity for each item a_k in a transaction was determined by drawing values from the Poisson distribution with mean q_{a_k}. Finally, the time interval of each two successive transactions was drawn from a Poisson distribution with a mean equal to 6.

6.2. Experimental evaluation

This study uses three tests to evaluate the proposed method. The first two tests evaluate effectiveness, while the third test evaluate efficiency. Table 3 lists the parameter settings for each test. The simulation sets the parameter $t_{current}$ to be the same as the occurring time of the last transaction in the dataset in all tests.

As we mentioned in Section 1, the goal of this study is to discover patterns that can approximately match the set of RFM-customer-patterns without customer identification information. To validate the effectiveness of the proposed approach, Test I is conducted to compare the similarity between a set of RFM-customer-patterns and four sets of discovered patterns. Four types of patterns are generated, namely R-patterns (i.e., the emerging patterns), F-patterns (i.e., the conventional frequent patterns), M-patterns (i.e., the utility patterns), and RFM-patterns.

Table 3
Parameter settings of different tests.

Test	Dataset(s)	α (%)	β	γ	δ
I	SC-POS-all	0–0.5	0–10 K	0–0.5	0.0001
	Foodmart	0–0.3	0–10 K	0–0.5	0.0001
II	SC-POS-all	0.1	10 K	0.5	0.001
	Foodmart	0.1	15 K	0.03	0.001
III	SYN-1∼SYN-9	1	1 M	0.5	0.01

Fig. 9 illustrates the Test I process. Since the retailing datasets in this study include customer ID data for customers checking out with their member cards, it is possible to acquire each member's purchasing history and then identify a group of RFM-customers.

The process of discovering RFM-customer-patterns can be summarized as follows: (1) only retain transactions involving a member card to form a complete member transaction database; (2) group transactions for each member; (3) calculate each member's *RScore*, *FScore*, and *tta*; (4) for each kind of score, utilize k-Mean clustering techniques ($k = 3$) to partition customers into three groups and rank them from 1(high) to 3(low); (5) sum up the ranking score and find out the customers with a ranking score smaller than or equal to 6 (i.e., RFM-customers); (6) retain transactions belonging to RFM-customers; and (7) conduct the FP-growth algorithm to discover RFM-customer-patterns from the transactions collected in step 6.

Based on the aforementioned procedure, Test I is conducted to compare the set of RFM-customer-patterns from the set of RFM-customer transactions with four sets of patterns from the set of all member transactions. This study employs the concepts of precision and recall rate to measure the effectiveness of RFMP-growth as follows.

$$precision = \frac{|\text{The generated patterns } \cap \text{RFM} - \text{customer} - \text{patterns}|}{|\text{The generated patterns}|}$$

$$recall = \frac{|\text{The generated patterns} \cap \text{RFM} - \text{customer} - \text{patterns}|}{|\text{RFM} - \text{customer} - \text{patterns}|}$$

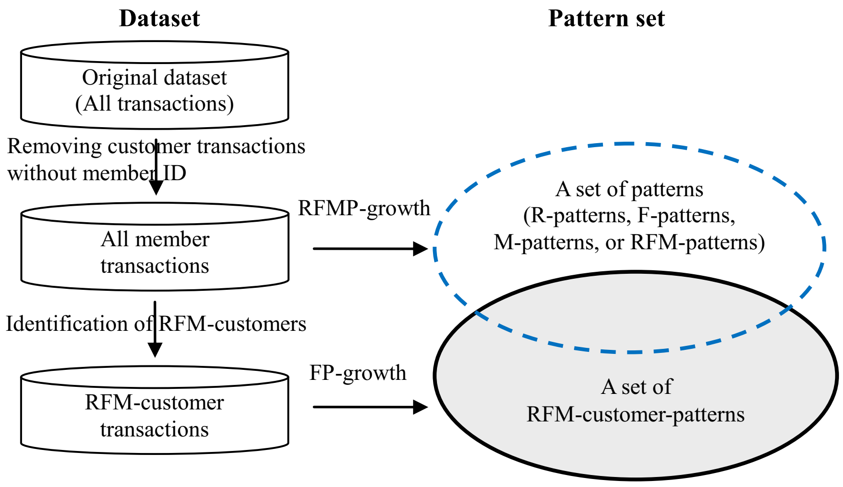

Dataset
Pattern set
Original dataset (All transactions)
Removing customer transactions without member ID
All member transactions
RFMP-growth
A set of patterns (R-patterns, F-patterns, M-patterns, or RFM-patterns)
Identification of RFM-customers
RFM-customer transactions
FP-growth
A set of RFM-customer-patterns

Fig. 9. The procedure of Test IV.

Table 4 shows the experimental results of the RFM-customer-patterns and the RFM-patterns with different parameter settings. Regarding the SC-POS-all dataset, RFMP-growth achieves precision and recall rates of 84.6% and 70.31% on average. The precision rate remains stable in all parameter settings. Results become more favorable as the value of α increases. Since the number of RFM-customer-patterns significantly decreases while α increases, it becomes more challenging to discover a complete but small set of RFM-customer-patterns. Surprisingly, the RFMP-growth achieves even higher precision and recall rates under such circumstance. Regarding the Foodmart dataset, RFMP-growth achieves precision and recall rates of 86.34% and 75.18% on average. We also observe that the precision rate achieves nearly 100% while α is less than or equal to 0.25%. Although the variations of both precision and recall rates increase while α increases to 0.3%, the results show that, with the appropriate thresholds, the proposed approach can still effectively retrieve the greater part of RFM-customer-patterns without customer identification information.

Table 5 indicates the effectiveness of the four pattern types based on the Foodmart dataset. The three worst-case RFM-patterns (Table 4) are listed as the baselines. The results indicate that the precision rates of the R-patterns are lower than those of the RFM-patterns. Although emerging pattern mining yields a high average recall rate, it tends to generate large pattern sets, requiring increased pattern-processing time. The average precision rates of the F-patterns and average recall rates of the M-patterns are higher compared with those of the RFM-patterns when the minsup value are 0.1% and 0.2%, respectively; however, the average F-pattern precision rates and M-pattern recall rates significantly decrease when the minsup value is 0.3%. These results suggest that the proposed RFM-pattern mining approach is relatively more stable compared with both the frequent and utility pattern-mining techniques when mining RFM-customer-patterns.

Next, Test II is to evaluate the future value (i.e., revenue) generated by the RFM-patterns. Two real-life datasets, SC-POS-all and Foodmart, are included in the test. We partition the dataset into two time period groups (i.e., two subsets of transactions). The set of transactions in the first time period, denoted as the training dataset, is used to generate both the frequent patterns and the RFM-patterns; while the set of transactions in the second time period, denoted as the testing dataset, is used for the calculation of the pattern revenue. Several comparisons are performed by varying the length of the time period in both training and testing datasets. We compute the percentage of frequent patterns and the corresponding revenue belonging to the set of RFM-patterns (i.e., $\frac{\#\text{of RFM-patterns}}{\#\text{of frequent patterns}}$ and $\frac{\text{revenue of RFM-patterns}}{\text{revenue of frequent patterns}}$) as the performance measure.

The results in Table 6 indicate that, on average, RFMP-growth retains less than 30% of frequent patterns from the training dataset of SC-POS-all but possesses over 60% of revenue from the testing dataset of SC-POS-all. That is, the average value of RFM-patterns is twice as high as conventional frequent patterns. Regarding the

Table 4
The comparisons between RFM-patterns and RFM-customer-patterns.

Dataset	# of RFM-customer-patterns	α (%)	β (K)	γ	# of RFM-patterns	Precision (%)	Recall (%)
SC-POS-all	1033	0.1	5	0.1	815	85.52	67.47
	(minsup = 0.1%)	0.1	10	0.1	752	86.17	62.73
		0.1	10	0.5	361	89.20	31.17
	222	0.3	5	0.1	228	83.13	62.16
	(minsup = 0.3%)	0.3	10	0.1	226	80.09	81.53
		0.3	10	0.5	166	79.82	81.98
	105	0.5	5	0.1	107	89.66	74.29
	(minsup = 0.5%)	0.5	10	0.1	107	84.11	85.71
		0.5	10	0.5	87	84.11	85.71
Foodmart	1557	0.2	5	0.1	1557	100.00	100.00
	(minsup = 0.2%)	0.2	5	0.5	1544	100.00	99.17
		0.2	10	0.1	1455	100.00	93.45
	1493	0.25	5	0.1	1510	97.15	98.26
	(minsup = 0.25%)	0.25	5	0.5	1497	97.13	97.39
		0.25	10	0.1	1417	97.32	92.36
	1010	0.3	5	0.1	669	90.28	59.80
	(minsup = 0.3%)	0.3	5	0.5	663	90.50	59.41
		0.3	10	0.1	647	90.42	57.92

Table 5
The comparisons between RFM-customer-patterns and other three patterns (Foodmart).

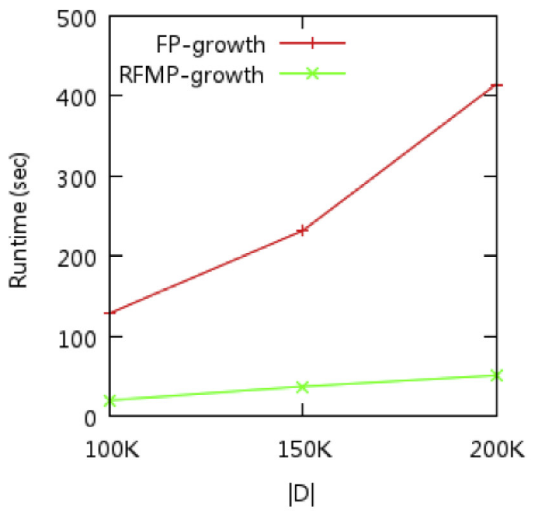
# Of RFM-customer-patterns	Pattern type	Parameter	# Of patterns	Precision (%)	Recall (%)
1557 missup = 0.2%	RFM-pattern	$\alpha = 0.2\%, \beta = 10\text{ K}, \gamma = 0.1$	1455	100	93.45
	R-pattern	$\alpha = 0, \beta = 0, \gamma = 1$	1881	78.63	94.99
		$\alpha = 0, \beta = 0, \gamma = 1.3$	1532	90.40	88.95
		$\alpha = 0, \beta = 0, \gamma = 1.5$	1373	95.70	84.39
		$\alpha = 0, \beta = 0, \gamma = 1.8$	1211	98.51	76.62
		$\alpha = 0, \beta = 0, \gamma = 2$	1093	99.91	70.13
	F-pattern	$\alpha = 0.22\%, \beta = 0, \gamma = 0$	1555	100.00	99.87
		$\alpha = 0.24\%, \beta = 0, \gamma = 0$	1532	100.00	98.39
		$\alpha = 0.26\%, \beta = 0, \gamma = 0$	1431	100.00	91.91
		$\alpha = 0.28\%, \beta = 0, \gamma = 0$	1137	100.00	73.03
		$\alpha = 0.30\%, \beta = 0, \gamma = 0$	669	100.00	42.97
	M-pattern	$\alpha = 0, \beta = 12\text{ K}, \gamma = 0$	1400	99.93	89.85
		$\alpha = 0, \beta = 14\text{ K}, \gamma = 0$	1347	99.93	86.45
		$\alpha = 0, \beta = 16\text{ K}, \gamma = 0$	1311	99.92	84.14
		$\alpha = 0, \beta = 18\text{ K}, \gamma = 0$	1269	99.92	81.44
		$\alpha = 0, \beta = 20\text{ K}, \gamma = 0$	1203	100.00	77.26
1493 missup = 0.25%	RFM-pattern	$\alpha = 0.25\%, \beta = 10\text{ K}, \gamma = 0.1$	1417	97.32	92.36
	R-pattern	$\alpha = 0, \beta = 0, \gamma = 1$	1881	75.60	95.24
		$\alpha = 0, \beta = 0, \gamma = 1.3$	1532	87.34	89.62
		$\alpha = 0, \beta = 0, \gamma = 1.5$	1373	92.50	85.06
		$\alpha = 0, \beta = 0, \gamma = 1.8$	1211	95.62	77.56
		$\alpha = 0, \beta = 0, \gamma = 2$	1093	96.80	70.86
	F-pattern	$\alpha = 0.22\%, \beta = 0, \gamma = 0$	1555	95.95	99.93
		$\alpha = 0.24\%, \beta = 0, \gamma = 0$	1532	96.61	99.13
		$\alpha = 0.26\%, \beta = 0, \gamma = 0$	1431	98.46	94.37
		$\alpha = 0.28\%, \beta = 0, \gamma = 0$	1137	99.56	75.82
		$\alpha = 0.30\%, \beta = 0, \gamma = 0$	669	100.00	44.81
	M-pattern	$\alpha = 0, \beta = 12\text{ K}, \gamma = 0$	1400	96.36	90.35
		$\alpha = 0, \beta = 14\text{ K}, \gamma = 0$	1347	96.21	86.81
		$\alpha = 0, \beta = 16\text{ K}, \gamma = 0$	1311	96.34	84.59
		$\alpha = 0, \beta = 18\text{ K}, \gamma = 0$	1269	96.69	82.18
		$\alpha = 0, \beta = 20\text{ K}, \gamma = 0$	1203	96.76	77.96
1010 missup = 0.3%	RFM-pattern	$\alpha = 0.3\%, \beta = 10\text{ K}, \gamma = 0.1$	647	90.42	57.92
	R-pattern	$\alpha = 0, \beta = 0, \gamma = 1$	1881	51.67	96.24
		$\alpha = 0, \beta = 0, \gamma = 1.3$	1532	60.51	91.78
		$\alpha = 0, \beta = 0, \gamma = 1.5$	1373	64.68	87.92
		$\alpha = 0, \beta = 0, \gamma = 1.8$	1211	67.88	81.39
		$\alpha = 0, \beta = 0, \gamma = 2$	1093	69.53	75.25%
	F-pattern	$\alpha = 0.22\%, \beta = 0, \gamma = 0$	1555	64.95	100.00
		$\alpha = 0.24\%, \beta = 0, \gamma = 0$	1532	65.86	99.90
		$\alpha = 0.26\%, \beta = 0, \gamma = 0$	1431	69.88	99.01
		$\alpha = 0.28\%, \beta = 0, \gamma = 0$	1137	79.60	89.60
		$\alpha = 0.30\%, \beta = 0, \gamma = 0$	669	90.28	59.80
	M-pattern	$\alpha = 0, \beta = 12\text{ K}, \gamma = 0$	1400	66.50	92.18
		$\alpha = 0, \beta = 14\text{ K}, \gamma = 0$	1347	66.52	88.71
		$\alpha = 0, \beta = 16\text{ K}, \gamma = 0$	1311	66.59	86.44
		$\alpha = 0, \beta = 18\text{ K}, \gamma = 0$	1269	67.30	84.55
		$\alpha = 0, \beta = 20\text{ K}, \gamma = 0$	1203	68.08	81.09

Table 6
The evaluations of the value of RFM-patterns using training and testing datasets.

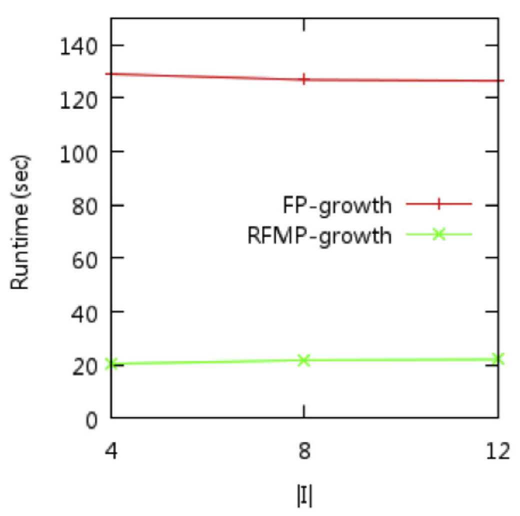
	Training data	Testing data	$\frac{\#\text{of RFM-patterns}}{\#\text{of frequentpatterns}}$ (%)	$\frac{\text{revenueof RFM-patterns}}{\text{revenueof frequentpatterns}}$ (%)
SC-POS-all	2001/12/26–2002/10/31	2002/11/01–2002/11/27	33.75	66.68
	2001/12/26–2002/09/31	2002/10/01–2002/11/27	29.74	61.89
	2001/12/26–2002/08/31	2002/09/01–2002/11/27	26.03	56.88
	2001/12/26–2002/07/31	2002/08/01–2002/11/27	27.06	55.65
		Average	29.14	60.27
Foodmart	1997/01/01–1998/08/31	1998/09/01–1998/11/30	42.66	58.25
	1997/01/01–1998/05/31	1998/06/01–1998/11/30	33.10	46.45
	1997/01/01–1998/02/28	1998/03/01–1998/11/30	24.57	36.14
	1997/01/01–1997/11/30	1997/12/01–1998/11/30	5.26	10.44
		Average	26.40	37.82

Foodmart dataset, RFMP-growth holds 37.82% of revenue from the testing dataset with retaining only 26.4% of frequent patterns from the training dataset in average. These results are promising since the RFM-patterns hold relatively high values among all patterns. In real-life applications, a large grocery retailer may offer more than ten thousand types of goods, and these items/itemsets may not have the same contribution to business. In such circumstances, RFMP-growth provides a feasible way to identify valuable patterns.

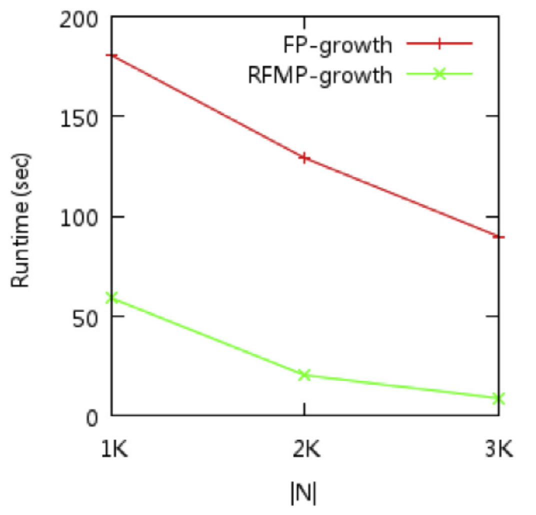
Finally, Test III is designed to evaluate the runtime scalabilities using nine synthetic datasets. Specifically, in data generation, we



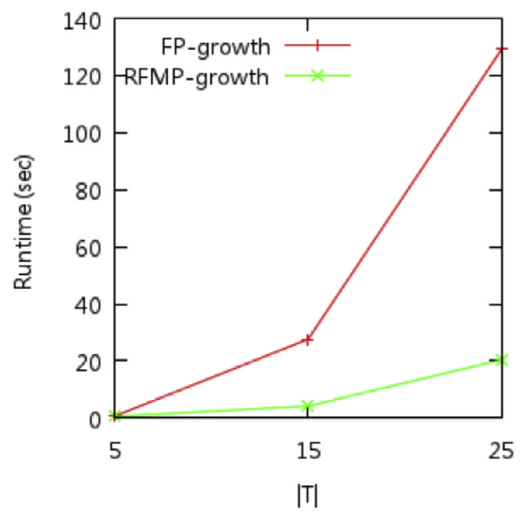
(a) Rutime vs. |*D*|



(b) Rutime vs. |*I*|



(c) Rutime vs. *N*



(d) Rutime vs. |*T*|

Fig. 10. The scalability analysis of runtime with different synthetic datasets.

varies the parameters |*D*| (from 100 K to 200 K) to decide the number of transactions; |*I*| (from 4 to 8) to control the length of potentially large itemset in a transaction; *N* (from 1000 to 3000) to decide the number of items; and |*T*| (from 5 to 25) to vary the length of transactions. Note that all the parameters regarding FP-growth and RFMP-growth are set as constant values.

Fig. 10 shows the results of runtime scalability analyses. The result shows, as expected, that RFMP-growth outperforms FP-growth in all tests. It is because RFMP-growth considers more constraints (i.e., additional thresholds) than FP-growth in the mining process. Moreover, the runtime scalability tests also indicate that both algorithms scale up exponentially with |*T*| but linearly with |*D*| and *N*. The parameter |*I*| has nearly no effect on runtime. Increasing the value of |*T*| increases both the transaction length and the possibility of having a large itemset in the transaction database. Therefore, both algorithms must recursively generate more conditional pattern bases and corresponding conditional pattern trees to discover patterns, increasing the execution time significantly. Next, increasing |*D*| increases runtime. This is because the data generator simply assumes that the customer purchase behavior is stable. That is, it has less variation in the generation of transaction data and thus has similar mining results. FP-growth has been proved to be an efficient and scalable method in frequent pattern mining. The above observations show that RFMP-growth can exhibit similar properties as FP-growth under different data characteristics.

7. Conclusion and future works

The integration of RFM analysis and association rule mining has been proven to be effective while discovering valuable customer purchasing behavior. Without customer identification information, however, previous approaches cannot be performed to discover RFM-customer-patterns from transactional databases. Integrating the concept of RFM analysis in frequent pattern mining, this study develops a novel approach to discover the set of approximate RFM-customer-patterns. We first evaluate the recency, frequency, and monetary scores of frequent patterns and, second, define RFM-patterns as approximations to RFM-customer-patterns. This study also proposes a tree structure (RFM-pattern-tree) to compactly store entire transaction database in main memory and develop a novel method (RFMP-growth) to efficiently discover a complete set of RFM-patterns.

Our novel method was investigated using detailed experiments. Three kinds of tests were designed using both synthetic and real-life datasets in our numerical study. Results show that RFMP-growth can efficiently and effectively discover more valuable patterns than conventional frequent pattern mining algorithms. More importantly, through setting proper recency, frequency, and monetary thresholds, RFMP-growth can identify most of RFM-customer-patterns. The proposed method provides a feasible way to realize the purchasing behavior from valued customer group without requiring any customer identification information.

Two primary limitations affected this study. First, we considered two retailing datasets in this purely experimental study. Further analysis on more real datasets should be conducted to examine the effectiveness of the proposed algorithm. Second, based on the design of RFM-patterns, users must repeatedly tune parameters to yield satisfactory results; however, this is time consuming. Future studies can develop maintenance mechanisms to efficiently determine the appropriate RFM-patterns each time the parameter values are adjusted.

This study could be also extended by further research. For example, it is possible to include fuzzy recency, frequency, and monetary constraints in the algorithm, creating a more flexible way to uncover other meaningful patterns. The proposed algorithm could be also valuable in frequent pattern mining for various types of application domains.

Acknowledgement

This research was supported in part by National Science Council of the Republic of China under the Grant NSC 99-2410-H-194-065 and NSC 102-2410-H-194-104-MY2.

References

[1] R. Agrawal, R. Srikant, Fast algorithms for mining association rules, in: Proceedings of 21th International Conference on Very Large Data, Bases, 1994, pp. 487–499.
[2] R. Agrawal, T. Imielinski, A. Swami, Mining association rules between sets of items in large databases, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1993, pp. 207–216.
[3] C.F. Ahmed, S.K. Tanbeer, B.-S. Jeong, Y.-K. Lee, Efficient tree structures for high utility pattern mining in incremental databases, IEEE Trans. Knowl. Data Eng. 21 (12) (2009) 1708–1721.
[4] F. Bonchi, C. Lucchese, Extending the state-of-the-art of constraint-based pattern discovery, Data Knowl. Eng. 60 (2) (2007) 377–399.
[5] J-F. Boulicaut, B. Jeudy, Constraint-based data mining, in: Data Mining and Knowledge Discovery Handbook, Springer, 2005, pp. 399–416.
[6] J.R. Bult, T.J. Wansbeek, Optimal selection for direct mail, Marketing Sci. 14 (4) (1995) 378–394.
[7] D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, T. Yiu, MAFIA: a maximal frequent itemset algorithm, IEEE Trans. Knowl. Data Eng. 17 (11) (2005) 1490–1504.
[8] Y.-L. Chen, J.-M. Chen, C.-W. Tung, A data mining approach for retail knowledge discovery with consideration of the effect of shelf-space adjacency on sales, Decis. Support Syst. 42 (3) (2006) 1503–1520.
[9] Y.-L. Chen, Y.-H. Hu, Constraint-based sequential pattern mining: the consideration of recency and compactness, Decis. Support Syst. 42 (2) (2006) 1203–1215.
[10] Y.-L. Chen, M.-H. Kuo, S.-Y. Wu, K. Tang, Discovering recency, frequency, and monetary (RFM) sequential patterns from customers' purchasing data, Electron. Comm. Res. Appl. 8 (5) (2009) 241–251.
[11] C.-H. Cheng, Y.-S. Chen, Classifying the segmentation of customer value via RFM model and RS theory, Expert Syst. Appl. 36 (3) (2009) 4176–4184.
[12] W.-Y. Chiang, To mine association rules of customer values via a data mining procedure with improved model: an empirical case study, Expert Syst. Appl. 38 (3) (2011) 1716–1722.
[13] R. Chithra, S. Nickolas, HUPT–mine: an efficient algorithm for high utility pattern mining, Int. J. Bus. Syst. Res. 6 (3) (2012) 279–295.
[14] C.-J. Chu, V.S. Tseng, T. Liang, An efficient algorithm for mining temporal high utility itemsets from data streams, J. Syst. Software 81 (7) (2008) 1105–1117.
[15] G. Dong, J. Li, Efficient mining of emerging patterns: discovering trends and differences, in: Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 1999, pp. 43–52.
[16] A. Erwin, R.P. Gopalan, N.R. Achuthan, CTU-Mine: an efficient high utility itemset mining algorithm using the pattern growth approach, in: Proceedings of the 7th IEEE International Conference on Computer and Information Technology, 2007, pp. 71–76.
[17] L. Feng, L. Wang, B. Jin, UT-tree: efficient mining of high utility itemsets from data streams, Intell. Data Anal. 17 (4) (2013) 585–602.
[18] M.N. Garofalakis, R. Rastogi, K. Shim, SPIRIT: sequential pattern mining with regular expression constraints, in: Proceedings of the 25th International Conference on Very Large Data, Bases, 1999, pp. 223–234.
[19] J. Hadden, A. Tiwari, R. Roy, D. Ruta, Computer assisted customer churn management: state-of-the-art and future trends, Comput. Oper. Res. 34 (10) (2005) 2902–2917.
[20] J. Han, J. Pei, Y. Yin, R. Mao, Mining frequent patterns without candidate generation: a frequent-pattern tree approach, Data Mining Knowl. Discovery 8 (1) (2004) 53–87.
[21] J. Han, J. Wang, Y. Lu, P. Tzvetkov, TFP: an efficient algorithm for mining top-*k* frequent closed itemsets, IEEE Trans. Knowl. Data Eng. 17 (5) (2005) 652–664.
[22] N.-C. Hsieh, An integrated data mining and behavioral scoring model for analyzing bank customers, Expert Syst. Appl. 27 (4) (2004) 623–633.
[23] J. Hu, A. Mojsilovica, High-utility pattern mining: a method for discovery of high-utility item sets, Pattern Recognit. 40 (11) (2007) 3317–3324.
[24] Y.-H. Hu, Y.-L. Chen, Mining association rules with multiple minimum supports: a new mining algorithm and a support tuning mechanism, Decis. Support Syst. 42 (1) (2006) 1–24.
[25] Y.-H. Hu, T.C.-K. Huang, Y.-H. Kao, Knowledge discovery of weighted RFM sequential patterns from customer sequence databases, J. Syst. Software 86 (3) (2013) 779–788.
[26] A.M. Hughes, Strategic Database Marketing: The Masterplan for Starting and Managing a Profitable, Customer-Based Marketing Program, third ed., McGraw-Hill Companies, 2005.
[27] Y. Jiang, J. Shang, Y. Liu, Maximizing customer satisfaction through an online recommendation system: a novel associative classification model, Decis. Support Syst. 48 (3) (2010) 470–479.
[28] Y.-M. Li, C.-H. Lin, C.-Y. Lai, Identifying influential reviewers for word-of-mouth marketing, Electron. Commer. Res. Appl. 9 (4) (2010) 294–304.
[29] Y. Liu, W.-K. Liao, A. Choudhary, A two-phase algorithm for fast discovery of high utility itemsets, in: Proceedings of the 9th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Hanoi, Vietnam, 2005, pp. 689–695.
[30] D.-R. Liu, C.-H. Lai, W.-J. Lee, A hybrid of sequential rules and collaborative filtering for product recommendation, Inf. Sci. 179 (20) (2009) 3505–3519.
[31] D.-R. Liu, Y.-Y. Shih, Integrating AHP and data mining for product recommendation based on customer lifetime value, Inf. Manage. 42 (3) (2005) 340–387.
[32] M.-Y. Lin, T.-F. Du, S.-C. Hsueh, High utility pattern mining using the maximal itemset property and lexicographic tree structures, Inf. Sci. 215 (1) (2012) 1–14.
[33] J.A. McCarty, M. Hastak, Segmentation approaches in data-mining: a comparison of RFM, CHAID, and logistic regression, J. Bus. Res. 60 (6) (2007) 656–662.
[34] J. Miglautsch, Thoughts on RFM scoring, J. Database Marketing 8 (1) (2000) 67–72.
[35] R.T. Ng, L.V.S. Lakshmanan, J. Han, A. Pang, Exploratory mining and pruning optimizations of constrained associations rules, Proc. ACM SIGMOD Int. Conf. Manage. Data (1998) 13–24.
[36] J. Pei, J. Han, W. Wang, Constraint-based sequential pattern mining: the pattern-growth methods, J. Intell. Inf. Syst. 28 (2007) 133–160.
[37] G.N. Punj, D.W. Stewart, Cluster analysis in marketing research: review and suggestions for application, J. Marketing Res. 20 (2) (1983) 134–148.
[38] G. Russ, M. Bottcher, R. Kruse, Relevance Feedback for Association Rules using Fuzzy Score Aggregation, in: Proceedings of the North American Fuzzy Information Processing Society, 2007, pp. 54–59.
[39] Q.Y. Shambour, J. Lu, A trust-semantic fusion-based recommendation approach for e-business applications, Decis. Support Syst. 54 (1) (2012) 768–780.
[40] B.-E. Shie, P.S. Yu, V.S. Tseng, Efficient algorithms for mining maximal high utility itemsets from data streams with different models, Expert Syst. Appl. 39 (17) (2012) 12947–12960.
[41] W. Song, Y. Liu, J. Li, Mining high utility itemsets by dynamically pruning the tree structure, Appl. Intell. 40 (1) (2014) 29–43.
[42] R. Srikant, Q. Vu, R. Agrawal, Mining association rules with item constraints, in: Proceedings of the 3rd International Conference on Knowledge Discovery and Data Mining, 1997, pp. 67–73.
[43] K. Tang, Y.-L. Chen, H.-W. Hu, Context-based market basket analysis in a multiple-store environment, Decis. Support Syst. 45 (1) (2008) 150–163.
[44] Q. Wan, A. An, Discovering transitional patterns and their significant milestones in transaction databases, IEEE Trans. Knowl. Data Eng. 21 (12) (2009) 1692–1707.
[45] H. Yao, H.J. Hamilton, Mining itemset utilities from transaction databases, Data Knowl. Eng. 59 (3) (2006) 603–626.