IEICE TRANS. INF. & SYST., VOL.E92–D, NO.12 DECEMBER 2009

PAPER

On Identifying Useful Patterns to Analyze Products in Retail Transaction Databases*

Unil YUN[†a)], ***Member***

SUMMARY Mining correlated patterns in large transaction databases is one of the essential tasks in data mining since a huge number of patterns are usually mined, but it is hard to find patterns with the correlation. The needed data analysis should be made according to the requirements of the particular real application. In previous mining approaches, patterns with the weak affinity are found even with a high minimum support. In this paper, we suggest weighted support affinity pattern mining in which a new measure, weighted support confidence (ws-confidence) is developed to identify correlated patterns with the weighted support affinity. To efficiently prune the weak affinity patterns, we prove that the ws-confidence measure satisfies the anti-monotone and cross weighted support properties which can be applied to eliminate patterns with dissimilar weighted support levels. Based on the two properties, we develop a weighted support affinity pattern mining algorithm (WSP). The weighted support affinity patterns can be useful to answer the comparative analysis queries such as finding itemsets containing items which give similar total selling expense levels with an acceptable error range $\alpha\%$ and detecting item lists with similar levels of total profits. In addition, our performance study shows that WSP is efficient and scalable for mining weighted support affinity patterns.

key words: *data mining, weighted pattern mining, ws-confidence, weighted support affinity*

1. Introduction

The extensive growth of data has created the need to find useful patterns among the huge data. Frequent pattern mining algorithms have been extensively studied with the broad applications and efficient pattern mining algorithms have been developed such as constraint-based pattern mining [5], closed pattern mining [9] and frequent pattern mining without support thresholds [3], [10]. These approaches may reduce the number of patterns but weak affinity patterns are still found in the result sets. To mine correlated patterns, interesting measures [6], [7], [12] have been proposed which detect correlated patterns. However, these measures are based on the supports of patterns so they could not mine correlated patterns related to the importance (weight). There have been studies [2], [8], [11], [13] to mine weighted frequent patterns which consider importance of items and patterns with the items. but the result patterns include weak affinity patterns.

Manuscript received June 30, 2006.
Manuscript revised August 6, 2007.

[†]The author is an assistant professor with Division of Computer Engineering, Chungbuk National University, South Korea.

[*]"This work was supported by the grant of the Korean Ministry of Education, Science and Technology" (The Regional Core Research Program / Chungbuk BIT Research-Oriented University Consortium).

a) E-mail: yunei@chungbuk.ac.kr

DOI: 10.1587/transinf.E92.D.2430

1.1 Problem Definition

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a unique set of items. A transaction database, TDB, is a set of transactions in which each transaction, denoted as a tuple ⟨tid, X⟩, contains a unique transaction identifier, tid, and a set of items. The support of a pattern is the number of transactions containing the pattern in the database. The problem of frequent pattern mining is to find the complete set of patterns satisfying a support threshold in the transaction database. To prune infrequent patterns, frequent pattern mining usually uses the anti-monotone property [1]. That is, if a pattern is infrequent pattern, all super patterns of the pattern must be infrequent patterns. Using the anti-monotone property, infrequent patterns can be pruned.

1.2 Typical Example

Let us give a motivating example for this work in retail databases. In general, cheap items have high frequencies (supports) and items with high prices (profits) have low frequencies. Previous pattern mining algorithms only focus on the supports of items so they cannot find itemsets with low supports but high weights although these patterns can be interesting patterns. Weight based pattern mining can discover weighted patterns. However, there is another concern in real retail databases.

Table 1 shows an example of weights (normalized prices) and supports (frequencies) of products (items) in a retail database. We assume that a weight between 1 and 40 is a low weight (↓), a weight between 50 and 100 is a high weight (↑), and a weight between 40 and 50 is a normal weight (-).

As you can see, several types of items exist: (1) Items

Table 1 An example of the item analysis in a retail database.

Product (item)	***A***	***B***	***C***	***D***	***E***
Weight (1-100)	30 (↓)	20 (↓)	90 (↑)	80 (↑)	90 (↑)
Support (1-1000)	800 (↑)	900 (↑)	200 (↓)	300 (↓)	700 (↑)
Product (item)	***F***	***G***	***H***	***I***	***J***
Weight (1-100)	70 (↑)	10 (↓)	40 (↓)	50 (−)	45 (−)
Support (1-1000)	800 (↑)	300 (↓)	100 (↓)	500 (−)	550 (−)

Table 2 An example of retail database.

Bar code	Item	Price	Support (frequency)	Normalized Weight
1	Laptop Computer	1200$	5000	1.2
2	Desktop Computer	700$	3000	0.7
3	Digital camera	300$	30000	0.3
4	Memory stick	200$	20000	0.2
5	Memory card	150$	10000	0.15
6	Hard disk	100$	5000	0.1
7	Mouse	40$	80000	0.04
8	Mouse pad	10$	100000	0.01

with low supports and high weights, (2) Items with high supports and low weights, (3) Items with high supports and high weights, (4) Items with low supports and low weights and so on. For instance, items within itemsets "AB" have low weights but high supports and all items of an itemset "CD" have high weights and low supports, but itemsets such as "AD", "BC", "AE", "AG", "CG", "EG", "GI" are different characteristics. Although two items B and C (or items A and D) have the same total sale amounts, items within the itemsets show dissimilar portions of the prices (profits) or the frequencies. The patterns with the similar (same) total profit may have different styles. We can analyze the items more exactly by identifying the patterns with the different weighted support affinity. In this paper, we propose a framework for mining correlated patterns with the weighted support affinity called WSP. We define a ws-confidence measure and concept of a weighted support affinity pattern by the ws-confidence. The weighted support affinity patterns can be applied in the analysis of patterns in retail databases. The main contributions of this paper are as follows (1) Introduction of the concept of the correlated pattern with the weighted support affinity, (2) Illustration of how the weighted support affinity patterns can be used in real applications of retail databases, (3) Suggestion of a new measure, ws-confidence to detect the correlated patterns, (4) Illustration of two properties, the cross weighted support property and the anti-monotone property of the ws-confidence to prune weak affinity patterns, (5) Development of WSP algorithm and experimental study

2. WSP: Weighted Support Affinity Pattern Mining with the High Correlation

We study the problem of correlated pattern mining with similar weighted support levels. To efficiently detect the weight affinity patterns, a new measure, ws-confidence is defined. We show that the ws-confidence satisfies the anti-monotone property, define cross weight property and prove that the ws-confidence satisfies the cross weight property. Based on the two properties, the weak weighted support affinity patterns are eliminated faster.

2.1 Preliminaries

Weight-based pattern mining is essential in that the approach not only reduces search space but also extracts more important patterns. As the first step, to set up weights of items, an attribute value of the retail items can be used. For example, prices (profits) of items could be used as a weight factor in market basket data. However, the real values of items are not suitable for weight values because of the big variation. Table 2 shows an example of a retail database. In general, we can know that variation of items' prices is so big that the prices cannot be directly used as weights. Therefore, the normalization process is needed which adjusts for differences among data in order to create a common basis for comparison.

Definition 2.1 Weight of a frequent pattern and Weight Range (WR)

A weight of an item is a non-negative real number which is assigned to reflect the importance of each item in the transaction database. Given a set of items, $I = \{i_1, i_2, \ldots, i_n\}$, the weight of a pattern is formally defined as

$$\text{Weight(P)} = \frac{\sum_{i=1}^{\text{length(P)}} Weight(x_i)}{legnth(P)}$$

In the equation, the length (length (P)) of a pattern P is the number of items in the pattern P. For example, the length of a pattern "i_2, i_3, i_6" is 3. The weights for items are assigned with $a_w \le w \le b_w$ according to items' importance or priority. The weights with $a_w \le w \le b_w$ are normalized with specific range (weight range).

From this example, weights of items are given between 0.31 and 1.5". According to the normalization process, the final weights of items are determined and items, and patterns with the items have their own weights.

Definition 2.2 Weighted support (WSupport) and weighted frequent pattern

A weighted support of a pattern is defined as the resultant value of multiplying the pattern's support with the weight of the pattern. A pattern is called a weighted frequent pattern if the weighted support of the pattern is no less than a minimum threshold. A weighted support is used

to prune weighted infrequent patterns.

Definition 2.3 Maximum Weight (MaxW) and Minimum Weight (MinW)

Maximum Weight (MaxW) is defined as the value of the maximum weight of items in transaction databases and Minimum Weight (MinW) is defined as a minimum weight of items within a conditional pattern.

Example 1: Table 4 shows example sets of items with different weight ranges. The MaxW of WR_1 is 1.1, the MaxW of WR_2 is 0.9, and the MaxW of WR_3 is 0.6.

Given TDB in Table 3, a minimum support, 2, and WR_1 as weights of items in Table 4, after weighted infrequent items are pruned, the weighted frequent list is ⟨a:4, b:4, c:4, d:5, f:3, g:3, h:2⟩, and, a weight list by the weight ascending order is ⟨f:0.7, c:0.8, h:0.8, g:0.9, b:1.0, d:1.0, a:1.1⟩. The projected conditional database (CDB("ad")) for a conditional pattern "ad" is {(c, g, b), (f, b), (c, h, g, a), (c, b)}. MinW ("ad") is 1.0 and weight ("ad") is 1.05 so we can know that the MinW (1.0) of the conditional pattern "ad" is less than the weight (1.05) of the conditional pattern "ad" in this example.

2.2 The Pattern Growth Method by a Weight Ascending Order and Bottom up Traversal Strategy

WSP is based on the pattern growth method with weight ascending prefix trees and bottom up traversal strategy. On this framework, the weight constraints are pushed deeply in the mining process. Given a frequent pattern $C_n(c_1c_2, .., c_n)$, C_n-conditional database is generated by $C_{n-1}(c_1c_2, .., c_{n-1})$-conditional database by collecting all the transactions containing an item c_n and removing following items: (1) weighted infrequent items, (2) all frequent items after the item c_n in the weighted frequent items sorted by a weight ascending order, and (3) the item c_n itself. The pattern $C_n(c_1c_2, .., c_n)$ is called conditional pattern (prefix pattern) and the set of transactions containing the pattern C_n is called C_n-conditional database (CDB(C_n)).

Table 3 Transaction database TDB.

TID	Set of items
100	a, b, c, d, g
200	a, b, d, f
300	a, b, c, d, g, h
400	c, f, g
500	a, b, c, d
600	d, f, h

Table 4 Example sets of items with different WRs.

Item	a	b	c	d	f	g	h
Support	4	4	4	5	3	3	2
Weight ($0.7 \leq WR_1 \leq 1.1$)	1.1	1.0	0.8	1.0	0.7	0.9	0.8
Weight ($0.7 \leq WR_2 \leq 0.9$)	0.9	0.75	0.8	0.85	0.75	0.7	0.85
Weight ($0.2 \leq WR_3 \leq 0.6$)	0.6	0.5	0.3	0.3	0.2	0.5	0.4

Let an item p_i be a weighted frequent item in a transaction database, TDB. The p_i-conditional database (p_i-projected database) for the item p_i is derived from the transaction database by collecting all the transactions containing the item p_i and removing following items: (1) weighted infrequent items, (2) all items after the item p_i in the weighted frequent items sorted by a weight ascending order, and (3) the item p_i itself.

2.2.1 Constructing Modified FP-Tree by Weight Ascending Order

Given a transaction database: {(a, b, c, d, g), (a, b, d), (a, b, c, d, h), (c, f, g), (a, b, c, d), (d, f, h)}, a min_sup of 2, a min_wsconf of 0.6, and a weight list is ⟨a:0.7, b:0.6, c:0.3, d:0.5, f:0.2, g:0.5, h:0.4⟩, the FP-trees in WSP algorithm are made as follows. Scan the transaction database one time and count the support of each item and check the weight of each item. After the first scan of the transaction database, we find a frequent list: ⟨a:4, b:4, c:4, d:5, f:2, g:2. h:2⟩. From the weighted support constraint in the pruning condition 1, weighted infrequent items "f", "g" and "h" are pruned because the weighted supports (1.4) of multiplying the supports (2) of items "f", "g" and "h" with MaxW (0.7) is less than the min_sup (2). Before items in each transaction are inserted in the FP-tree, weighted frequent items in each transaction of the transaction database are sorted by a weight ascending order: {(c, d, b, a), (d, b, a), (c, d, b, a), (c), (c, d, b, a), (d)}. The sorted frequent items in each transaction are sequentially inserted in a global FP-tree along a path from the root to the corresponding node. Figure 1 presents the global FP-tree and the corresponding header table for this mining process in WSP. As shown in Fig. 1, with weight ascending prefix trees and bottom up traversal strategy, given a conditional pattern P, and a pattern Q in the conditional database with the conditional pattern P, you can see that weight (P) is no less than the weight of a pattern Q within a transaction in the Conditional Database.

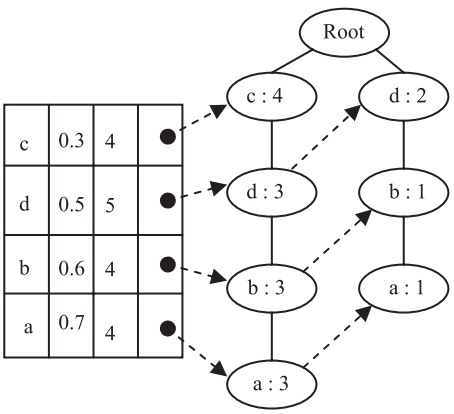

Fig. 1 The global FP-tree by the weight ascending order.

Table 5 Weighted support affinity levels.

Candidate patterns	ws-confidence	Candidate patterns	ws-confidence	Candidate patterns	ws-confidence
AB	0.59 ((20/30)* (800/900))	AC	0.14 ((30/80)* (300/800))	CG	0.125 ((10/80)* (300/300))
CD	0.59 ((80/90)* (200/300))	AE	0.29 ((30/90)* (700/800))	CI	0.375 ((50/80)* (300/500))
EF	0.68 ((70/90)* (700/800))	AG	0.125 ((10/30)* (300/800))	EG	0.048 ((10/90)* (300/700))
GH	0.375 ((10/20)* (300/400))	AI	0.375 ((30/50)* (500/800))	EI	0.397 ((50/90)* (500/700))
IJ	0.82 ((45/50)* (500/550))	CE	0.38 ((80/90)* (300/700))	GI	0.12 ((10/50)* (300/500))

2.3 The ws-Confidence

In real market business, not only the frequencies but also the prices (profits) are important factors. Our approach focuses on patterns containing items with similar total profits or total sale expenses. To detect the correlated patterns, we define the ws-confidence measure and the concept of weighted support affinity patterns.

Definition 2.4 The ws-confidence

A weighted support confidence of a pattern $P = \{i_1, i_2, \ldots, i_m\}$, denoted as ws-confidence, is a measure that reflects the overall weighted support affinity among items within the itemsets. It is defined as the ratio of the minimum weighted support of the items within the pattern to the maximum weighted support of the items within the pattern. That is, this measure is defined as

$$\text{WSconf(P)} = \frac{\text{Min}_{1\le j\le m}\{\text{support}(\{i_j\}) * \text{weight}(\{i_j\})\}}{\text{Max}_{1\le k\le m}\{\text{support}(\{i_k\}) * \text{weight}(\{i_k\})\}}$$

There may be other ways to examine the weighted support affinity levels. More complex definitions may detect more exact affinity levels. However, based on the definition of the ws-confidence, we will define two properties which are used for identifying strong weighted support affinity patterns.

2.3.1 The Usefulness (Necessity) of the Weighted Support Affinity Pattern

Patterns with similar weighted support levels can be useful to process comparative analysis queries in a real retail application. If the market manager wants to find item lists with similar total sale cost (frequency * price), they need to divide all items by comparing the total sale (profit) amount.

A market managers' main job is to analyze the sold item's trend. Sometimes, in real business, marketing managers or trend analysts are interested in products (items) with high prices (profits) and/or high frequencies. They want to analyze the patterns containing the items with more emphasis on some particular (high profits or high frequencies) products (items) and less emphasis on other products. In real market business, not only the frequencies but also the prices (profits) are important factors.

Trend analysts are mainly interested in comparing and analyzing total sale amount or total profits of each item in retail databases. According to the data analysis, the marketing policies about items' price decision are different. In other words, the comparison and analysis of correlated sequential patterns is essential to make plans. As a motivating application, in real business, marketing managers would like to know the item lists which give similar total selling expense levels with an acceptable error range $\alpha\%$. Trend analysts want to examine itemsets with similar levels of total profits (multiplying the profit of an item with the sale number of the item). Weighted support affinity pattern mining can give answers about the comparative analysis queries and discover interesting patterns which cannot be detected by the conventional sequential pattern mining approaches. Let us illustrate the usefulness of the weighted support affinity in market basket data. Given a retail database in Table 1 and a minimum (weighted) support threshold of 20,000, except for the items "G" and "H", the total profits (weighted support) of other items are greater than the minimum (weighted) support. However, the correlation of the patterns may be low. For instance, the items "A" and "D" have the same total profits 24000 but the affinity of two items is so low because an item "A" has a low price and high frequency. Meanwhile, the other item "D" has a high price but a low frequency. The trend analyst wants to identify two types of itemsets containing the items because the market policies of the items are totally different. As another example, the patterns {D, E} and {F, I} have weak affinity. Although the itemsets' total profits are greater than the minimum threshold (24000), the items within the patterns have different weight and support portions. Based on the definition of the ws-confidence, Table 5 shows the ws-confidence of itemsets with two items. The itemsets "AB", "CD", "EF" and "IJ" and the patterns with the itemsets have weighted support affinity with more than ws-confidence 0.5. On the other hand, the ws-confidences of the pattern containing the items such as "AC", "AG", "CG", GI" are less than 0.2. The ws-confidence is useful for detecting patterns with the similar weighted support affinity. Previous pattern mining algorithms cannot detect patterns with the high affinity and weight based pattern mining also cannot find the similar weighted support levels. Weighted support affinity pattern mining can give answers about the comparative analysis queries and discover interesting patterns which cannot be detected by the conventional pattern mining approaches. Correlated patterns can be used in dividing customers into several segments based on their similar purchasing behavior, exploring the association structure between the sales of different products, and determining the products' reasonable prices by analyzing total sales amounts.

2.3.2 The Anti-Monotone and Cross Weighted Support Properties of the ws-Confidence

Property 1 The anti-monotone property of the ws-confidence

If the ws-confidence of a pattern P is less than a minimum ws-confidence, the ws-confidence of any super pattern of the pattern P is also less than the minimum ws-confidence. Based on the characteristic, the pruning of weak affinity patterns can be efficiently performed.

Lemma 1 The ws-confidence measure has the anti-monotonic property.

Given a pattern $P = \{i_1, i_2, \ldots, i_m\}$, $\mathrm{Max}_{1\le k\le m}\{\mathrm{support}(\{i_k\}) * \mathrm{weight}(\{i_k\})\}$ of the pattern P is always greater than or equal to that of a subset of pattern P and $\mathrm{Min}_{1\le j\le m}\{\mathrm{support}(\{i_j\}) * \mathrm{weight}(\{i_j\})\}$ of the pattern P is always less than or equal to that of a subset of pattern P. Therefore, we know that

$$\frac{\mathrm{Min}_{1\le j\le m}\{\mathrm{support}(\{i_j\}) * \mathrm{weight}(\{i_j\})\}}{\mathrm{Max}_{1\le k\le m}\{\mathrm{support}(\{i_k\}) * \mathrm{weight}(\{i_k\})\}} \le \frac{\mathrm{Min}_{1\le j\le m-1}\{\mathrm{support}(\{i_j\}) * \mathrm{weight}(\{i_j\})\}}{\mathrm{Max}_{1\le k\le m-1}\{\mathrm{support}(\{i_k\}) * \mathrm{weight}(\{i_k\})\}}$$

That is, if the ws-confidence of the pattern $\{i_1, i_2, \ldots, i_m\}$ is no less than a minimum ws-confidence, so is every subset of size $m-1$. Thus, the ws-confidence can be used to prune the exponential search space. ■

Property 2 Cross weighted support property

Given a threshold t, a pattern P is a cross weighted support pattern with respect to the threshold t if the pattern contains two items x and y which have different levels ((support ({x}) * weight ({x}))/ (support ({y}) * weight ({y})) < t. where $0 < t < 1$) of weighted supports. For any cross weighted support pattern P with regard to a threshold t, it is guaranteed that wsconf (P) is less than t.

Lemma 2 The ws-confidence has cross weighted support property.

Assume that there is a cross weighted support pattern P that contains at least two items X and Y such that (support ({x}) * weight ({x}))/ (support ({y}) * weight ({y})) < t. where $0 < t < 1$.

$$\begin{aligned}
&\mathrm{WSconf(P)} \\
&= \frac{\mathrm{Min}_{1\le j\le m}\{\mathrm{support}(\{i_j\}) * \mathrm{weight}(\{i_j\})\}}{\mathrm{Max}_{1\le k\le m}\{\mathrm{support}(\{i_k\}) * \mathrm{weight}(\{i_k\})\}} \\
&\le \frac{\mathrm{Min}_{1\le j\le m}\{\ldots, \mathrm{support}(\{x\}) * \mathrm{weight}(\{x\}), \ldots, \mathrm{support}(\{y\}) * \mathrm{weight}(\{y\}), \ldots\}}{\mathrm{Max}_{1\le k\le m}\{\ldots, \mathrm{support}(\{x\}) * \mathrm{weight}(\{x\}), \ldots, \mathrm{support}(\{y\}) * \mathrm{weight}(\{y\}), \ldots\}} \\
&\le \frac{\mathrm{support}(\{x\}) * \mathrm{weight}(\{x\})}{\mathrm{Max}_{1\le k\le m}\{\ldots, \mathrm{support}(\{x\}) * \mathrm{weight}(\{x\}), \ldots, \mathrm{support}(\{y\}) * \mathrm{weight}(\{y\}), \ldots\}} \\
&\le \frac{\mathrm{support}(\{x\}) * \mathrm{weight}(\{x\})}{\mathrm{support}(\{y\}) * \mathrm{weight}(\{y\})} < t
\end{aligned}$$

As a result, we know that the value of the ws-confidence is less than the min_wsconf for any cross support pattern P with regard to a ws-confidence threshold, t. ■

2.3.3 The Pruning Example of the Anti-Monotone and Cross Weighted Support Properties on the ws-Confidence

In this section, we show how the anti-monotone property and the cross weighted support property can be used in our pruning strategy. From the anti-monotone property of the ws-confidence, if the ws-confidence of a pattern is less than the min_wsconf, any super pattern of the pattern is removed. Meanwhile, given an item x, all patterns that contain x and at least an item with weighted support less than $t \cdot \mathrm{weight}(x) \cdot \mathrm{support}(x)$ (for $0 < t < 1$) are cross weighted support patterns and the ws-confidences of the patterns are less than t. The cross weighted support patterns can be directly pruned without calculating the ws-confidences.

Given transaction database TDB in Table 3 and WR_2 for weights of items, a weight list for eight items <a: 0.9, b:0.75, c:0.8, d:0.85, f:0.75, g:0.7, h:0.85>, and the minimum ws-confidence of 0.8, the support list of items is {a: 4, b: 4, c: 4, d: 5, f: 3, g: 3, h: 2}. The ws-confidence (0.70) of a pattern "cf" is less than the minimum ws-confidence (0.8) so the pattern is pruned. From the anti-monotone property, we can prune the super patterns such as "cfb" and "cfg" since these patterns have one subset "cf" which is not a weight support affinity pattern. Meanwhile, we can prune cross weighted support patterns by the cross weighted support property. With the support list and weight list, the weighted support list by the ascending order is computed which is: {h: 1.7 (2 * 0.85), g: 2.1 (3* 0.7), f: 2.25 (3 * 0.75), b: 3.0 (4 * 0.75), c: 3.2 (4 * 0.8), a: 3.6 (4 * 0.9), d: 4.25 (5 * 0.85)}. We can find an item "f" with wsupport ("f") = 2.25 (3 * 0.75) < wsupport ("b") * min_wsconf (0.8) = 2.4. If we split the item list into two group {items "h", "g", and "f"} and {items "b", "c", "a", and "d"}, any pattern including items from both groups is the cross weighted support pattern with the min_wsconf because the ws-confidence is always less than the minimum ws-confidence for cross weighted support patterns. In this example, without applying the cross weighted support property, the cross weighted support patterns such as "hb", "cg", "af", and "df" have to be generated as candidate patterns and prune them later by computing the ws-confidence values of the patterns. Note that those patterns such as the patterns "hb", "cg", "af", and "df" are not pruned by the anti-monotone property because every subset of the patterns is the weighted support affinity pattern.

2.4 WSP: Mining Weighted Support Affinity Patterns

Definition 3.5 Weighted support affinity pattern

A pattern is weighted support affinity pattern if (1) The weighted support of the pattern is no less than a minimum support, and (2) The ws-confidence of the pattern is no less than a minimum ws-confidence.

Pruning condition 1: (Weighted support constraint)

A pattern P is a weighted frequent pattern if and if only

|P| > 0 and the weighted support (wsupport (P)) of the pattern P is no less than min_sup.

A pattern can be a weighted frequent pattern although sub patterns of the pattern are weighted infrequent. Therefore, we cannot use the anti-monotone property of weighted frequent patterns during mining process. To overcome the problem, we use an approximate maximum weight to prune weighted frequent patterns earlier but maintain the anti-monotone property. Based on weight ascending prefix trees and bottom up traversal strategy in the pattern growth method, the approximate maximum weight is efficiently calculated. In a transaction database, a Maximum Weight (MaxW) among items is used as the approximate maximum weight. Meanwhile, after the global FP-tree is constructed, a Minimum Weight (MinW) of items within a conditional pattern in the conditional database is used as the approximate maximum weight to compute the weighted support in a conditional database. MinW is a maximum weight value in conditional database so it is more accurate.

From using the approximate weighted support, any super pattern of the weighted infrequent patterns whose weighted support is less than the minimum support, is weighted infrequent pattern so it can be pruned. Note that the approximate weighted support is an approximate value so in final step, we should check if the pattern is really weighted frequent pattern (support (P) * weight (P) ≥ min_sup) and prune weighted infrequent patterns which satisfy this condition ("support (P) * MaxW (MinW) ≥ min_sup").

Pruning condition 2: (ws-confidence ≥ min_wsconf)

A pattern P is a weighted support affinity pattern if and if only |P| > 0 and wsconf (P) ≥ min_wsconf. In the pruning condition 2, the anti-monotone property and the cross weight property are applied to prune weak affinity patterns.

2.4.1 Mining Weighted Support Affinity Patterns

As shown at Fig. 1 in Sect. 2.2.1, after the global FP-tree is constructed from the transaction database, WSP divides mining the FP-tree into mining smaller FP trees by the bottom up traversal approach. From the global FP-tree, WSP mines (1) the patterns containing item "a" which have the highest weight, (2) the patterns including "b" but not "a", (3) the patterns containing "d" but no "a" or "b", and finally (4) the patterns containing "c" but no "a", "b" or "d". First, for a node with the item "a", we generate a conditional database by starting from the item a's head and following a's node-link. The conditional database for the conditional pattern "a:3" (pattern:support) contains two transactions: {(bdc:3) and (bd:1)}. In WSP, item "c:3" is pruned by the ws-confidence. From a local item "c:3" and a conditional pattern "a", the candidate pattern is "ac:3" and the ws-confidence (0.429 ((4*0.3)/(4*0.7))) of the pattern "ac:3" is less than the min_wsconf (0.6). Meanwhile, the ws-confidences of the local items "b" and "d" with the conditional pattern "a" are greater than the min_wsconf. As a result, "a"-conditional database returns the patterns "ab:4" and "ad:4" as a weighted support affinity patterns and the local FP-trees are constructed recursively. Second, for the node with the item "b" in the FP-tree, WSP derives a conditional database {(dc:3), and (d:1)} with the conditional pattern (b:3). A local item "c:3" is pruned by the ws-confidence. After pruning the item in the conditional database, the projected FP-tree for the conditional pattern "b:3" is constructed and the "b"-conditional database returns the pattern "bd:4" as a weighted support affinity pattern (ws-confidence of the pattern "bd" is 0.96 ((4*0.6)/(5*0.5)). In this way, we can build local conditional FP-trees from the global FP-tree and mine weighted support affinity patterns from them recursively.

2.4.2 WSP Algorithm

WSP algorithm: mining Weighted Support affinity Patterns

Input: (1) A transaction database: TDB,
(2) Minimum support: min_sup,
(3) Minimum ws-confidence: min_wsconf

Output: The set of weighted support affinity patterns.

Begin

1. Let WFP be the set of weighted support affinity patterns that satisfy the constraints. Initialize WFP ← {};
2. Scan TDB once to find the global weighted frequent items satisfying the following condition: If the approximate weighted support (support * MaxW) of an item is no less than a minimum support, the item is a weighted frequent item.
3. Sort items of WFP by a weight ascending order.
4. Scan the TDB again and build a global FP-tree using weight_order.
5. Call WSP (FP-tree, {}, WFP)

Procedure WSP (Tree, α, WFP)

1: For each a_i in the header of Tree do
2: set $\beta = \alpha$ U a_i;
3: Get a set I_β of items to be included in β conditional database, CDB_β;
4: For each item in I_β,
Compute its count in β conditional database;
5: For each b_j in I_β do
6: If (sub (β b_j) * MinW < min_sup) delete b_j from I_β;
7: If (wsconf (β b_j) < min_wsconf) delete b_j from I_β;
8: End for
9: $Tree_\beta$ ← FP_Tree_Construction (I_β, CDB_β)
10: If $Tree_\beta \neq 0$ then
11: Call WSP ($Tree_\beta$, β, WFP)
12: End for

The WSP algorithm calls the recursive procedure WSP (Tree, α, WFP). Line 6 generates weighted frequent patterns. Line 7 detects weighted support affinity patterns. If a pattern in a conditional database satisfies the two constraints in lines 6 and 7, it is inserted into a local FP-tree in line 9.

If the local FP-tree is not empty, the procedure WSP (Tree$_\beta$, β, WFP) is called recursively in line 11.

3. Performance Evaluation

We report our experimental results on the performance of WSP in comparison with two recently developed algorithms, Hyperclique Miner [12] and WFIM [12]. Hyperclique Miner is a mining algorithm to generate patterns involving items with similar support levels. WSP and Hyperclique Miner have different target applications so we can see the different result sets by comparing WSP with Hyperclique Miner. To detect support affinity patterns, Hyperclique Miner defined h-confidence (h-conf) measure. h-conf of a pattern "$i_1, i_2, \ldots, i_m$" is defined as support $(\{i_1, i_2, \ldots, i_m\})/\text{Max}_{1\le k\le m}\{\text{support}(\{i_k\})\}$. H-confidence is a support affinity measure but ws-conf measure considers weight and support affinity simultaneously. WFIM is the first weighted frequent itemset mining algorithm to use a pattern growth method. In this experimental test, normalized weights with weight range are used.

The three real datasets used are Connect, Pumsb and Mushroom in Table 6. More information about the real datasets can be found in the frequent pattern mining dataset repository (http://fimi.cs.helsinki.fi/data/). The synthetic datasets T10I4Dx contain from 100 k to 1000 k transactions. The synthetic datasets are generated from the IBM dataset generator (http://www.almaden.ibm.com/software/projects/hdb/resources.shtml). WSP was written in C++ and experiments were performed on a sparcv9 processor operating at 1062 MHz, with 2048MB of memory on a Unix machine. In our experiments, we used normalized weights with different weight ranges (Weights: 0.1 - 0.2 in Connect dataset, 0.2 - 0.5 in Pumsb datasets, and 0.6 - 0.9 in Mushroom datasets) by a random generation function.

Table 6 Data characteristics.

Data sets	Size	#Trans	#Items	Average (Maximum) Transaction length
Connect	12.14M	67557	150	43 (43)
Pumsb	14.75M	49046	2113	74 (74)
Mushroom	0.83M	8124	120	23 (23)
T10I4Dx	5.06 - 50.6M	100k - 1000k	1000	10 (31)

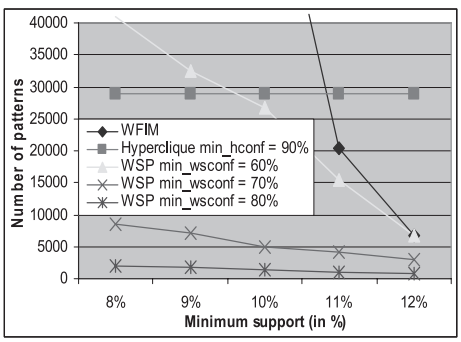

Fig. 2 Efficiency of the ws-confidence (Connect).

3.1 Comparison with Other Algorithms

First, we evaluate the performance on the Connect dataset. In Fig. 2 and Fig. 3, WSP generates fewer patterns than Hyperclique Miner and WFIM, and WSP is faster than other two algorithms. Even, in Fig. 3, we could not show the runtime of Hyperclique Miner because the runtime becomes very large as the minimum support is decreased. In Fig. 4 and Fig. 5, we report the evaluation results for the Pumsb dataset. We used a high 90% h-conf threshold to find patterns with high support affinity. As shown in Fig. 4 and Fig. 5, the result patterns from Hyperclique Miner are larger than the result patterns from WSP (using both 80% and 90% ws-conf thresholds). Moreover, WSP is the fastest among

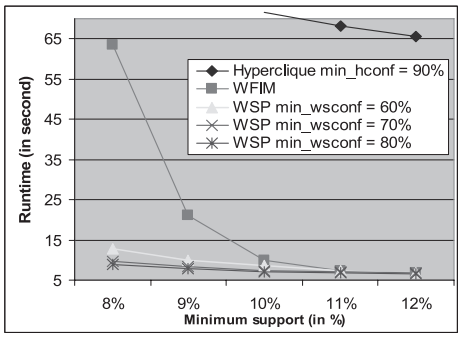

Fig. 3 Processing time (Connect).

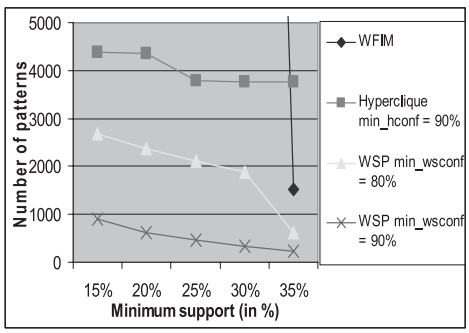

Fig. 4 Efficiency of the ws-confidence (Pumsb).

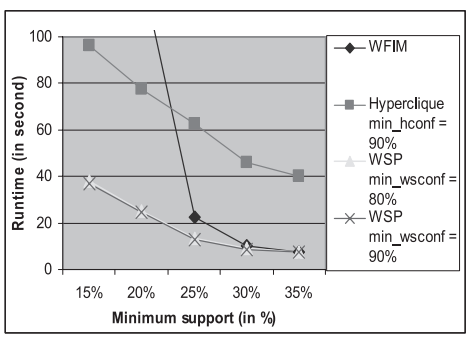

Fig. 5 Processing time (Pumsb).

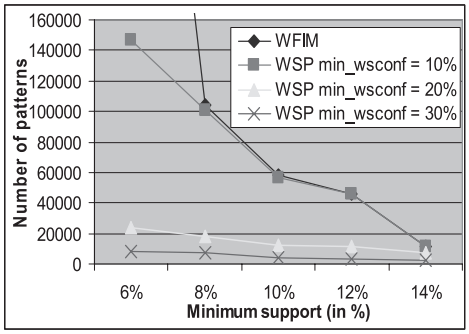

Fig. 6 Efficiency of the ws-confidence (Mushroom).

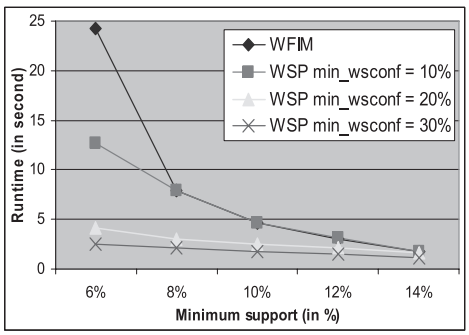

Fig. 7 Processing time (Mushroom)

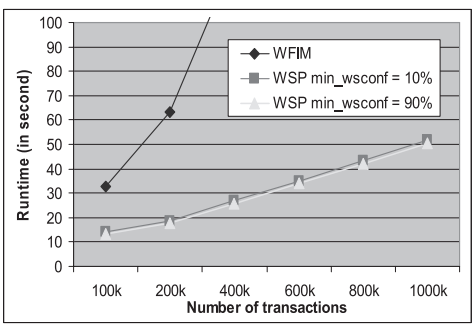

Fig. 8 Scalability test (T10I4Dx).

three algorithms and generates fewer patterns. Figure 6 and Fig. 7 demonstrate the results of performance test using the Mushroom dataset. WSP outperforms WFIM and Hyperclique Miner. Note that the results of Hyperclique Miner could not be shown in the Fig. 6 and Fig. 7 because the number of patterns and runtimes are dramatically increased when only minimum h-confidence thresholds (min_hconf) of less than 14% are used. It is not surprising that the number of patterns and the runtime can be reduced by using the ws-confidence, h-confidence or support thresholds but we can find following meaningful results: (1) Previous mining algorithms cannot detect weighted support affinity patterns. Hyperclique Miner can mine patterns with similar support levels. However, from above experiments, we can know that the result patterns of Hyperclique Miner are totally different from these of WSP. Additionally, both WSP and WFIM algorithms used weight constraints but WFIM is not effective to find the affinity patterns. (2) WSP can identify patterns with the weighted support affinity.

3.2 Scalability Test

T10I4DxK dataset is used to test scalability with the number of transactions. The scalability test is performed with regard to the number of transactions from 100 K to 1000 K on the T10I4Dx synthetic datasets. We set a minimum support as 0.003% and use normalized weights between 0.1 and 0.9. In Fig. 8, we can see that WSP has much better scalability. Without ws-confidence pruning, in WFIM, the runtime increases dramatically.

4. Conclusion

In this paper, we proposed the problem of mining correlated patterns with the weighted support affinity. Our approach focused on mining correlated patterns containing items with similar total profits or total sale expenses. To identify the weighted support affinity patterns, we defined the ws-confidence measure. Based on the framework, WSP algorithm is developed to detect correlated patterns with the weighted support affinity by pushing the ws-confidence into the pattern growth approach. The performance analysis shows that WSP is efficient and scalable for mining weighted support affinity patterns. WSP may be less efficient than previous FP-tree structure in terms of the memory usage. If space usage (main memory usage) is the main issue, the weight of each item can be stored in the header table instead of each node. Then, the memory usage about weight values of items depends on only the number of unique items, not the number of nodes. Thus, extra memory usage in weight based pattern mining can be negligible.

References

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," Proc. 20th International Conference on Very Large Data Bases, (VLDB'94), pp.487–499, Sept. 1994.
[2] C.H. Cai, A.W. Chee Fu, C.H. Cheng, and W.W. Kwong, "Mining association rules with weighted items," Proc. Sixth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2005), July 1998.
[3] Y.L. Cheung and A.W. Fu, "Mining frequent itemsets without support threshold: With and without item constraints," IEEE Trans. Knowl. Data Eng., vol.16, no.9, pp.1052–1069, 2004.
[4] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," Data Mining and Knowledge Discovery, vol.8, pp.53–87, Jan. 2004.
[5] J. Pei, J. Han, and L.V.S. Lakshmanan, "Pushing convertible constraints in frequent itemset mining," Data Mining and Knowledge Discovery, vol.8, no.3, pp.227–252, Jan. 2004.
[6] E.R. Omiecinski, "Alternative interest measures for mining associations in databases," IEEE Trans. Knowl. Data Eng., vol.15, no.1, pp.57–69, May 2004.
[7] P.-N. Tan, V. Kumar, and J. Srivastava, "Selecting the right interestingness measure for association patterns," Proc. 8th ACM SIGKDD Conference, pp.32–41, July 2002.
[8] F. Tao, "Weighted association rule mining using weighted support and significant framework," Proc. Ninth ACM SIGKDD In-

ternational Conference on Knowledge Discovery and Data Mining, pp.661–666, Aug. 2003.

[9] J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the best strategies for mining frequent closed itemsets," Proc. Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.236–245, Aug. 2003.

[10] J. Wang, J. Han, Y. Lu, and P. Tzvetkov, "TFP: An efficient algorithm for mining top-K frequent closed itemsets," IEEE Trans. Knowl. Data Eng., vol.17, no.5, pp.652–664, May 2005.

[11] W. Wang, J. Yang, and P.S. Yu, "Efficient mining of weighted association rules (WAR)," Proc. Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.270–274, Aug. 2000.

[12] H. Xiong, P.N. Tan, and V. Kumar, "Mining strong affinity association patterns in data sets with skewed support distribution," Proc. Third IEEE International Conference on Data Mining (ICDM 2003), pp.387–394, Dec. 2003.

[13] U. Yun and J.J. Leggett, "WFIM: Weighted frequent itemset mining with a weight range and a minimum weight," Proc. Fifth SIAM International Conference on Data Mining, pp.636–640, April 2005.

Unil Yun received the MS degree in Computer Science and Engineering from Korea University, Seoul, Korea, in 1997, and the PhD degree in Computer Science from Texas A&M University, Texas, USA, in 2005. He worked at Multimedia Laboratory, Korea Telecom, from 1997 to 2002. After receiving the PhD degree, he worked as a post-doctoral associate for almost one year at the Computer Science Department of Texas A&M University. After then, he worked as a senior researcher in Electronics and Telecommunications Research Institute (ETRI). Currently, he is an assistant professor in School of Electrical & Computer Engineering, of Chungbuk National University, Cheung Ju, South Korea. His research interests include data mining, information retrieval, database systems, artificial intelligence, and digital libraries.