NEW APPROACHES TO WEIGHTED FREQUENT PATTERN MINING

A Dissertation

by

UNIL YUN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

December 2005

Major Subject: Computer Science

NEW APPROACHES TO WEIGHTED FREQUENT PATTERN MINING

A Dissertation

by

UNIL YUN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	John J. Leggett
Committee Members,	Richard Furuta
	Frank M. Shipman III
	Joobin Choobineh
Head of Department,	Valerie E. Taylor

December 2005

Major Subject: Computer Science

ABSTRACT

New Approaches to Weighted Frequent Pattern Mining. (December 2005)

Unil Yun, B.S., Hong Ik University;

M.S., Korea University

Chair of Advisory Committee: Dr. John J. Leggett

Researchers have proposed frequent pattern mining algorithms that are more efficient than previous algorithms and generate fewer but more important patterns. Many techniques such as depth first/breadth first search, use of tree/other data structures, top down/bottom up traversal and vertical/horizontal formats for frequent pattern mining have been developed. Most frequent pattern mining algorithms use a support measure to prune the combinatorial search space. However, support-based pruning is not enough when taking into consideration the characteristics of real datasets. Additionally, after mining datasets to obtain the frequent patterns, there is no way to adjust the number of frequent patterns through user feedback, except for changing the minimum support. Alternative measures for mining frequent patterns have been suggested to address these issues. One of the main limitations of the traditional approach for mining frequent patterns is that all items are treated uniformly when, in reality, items have different importance. For this reason, weighted frequent pattern mining algorithms have been suggested that give different weights to items according to their significance. The main focus in weighted frequent pattern mining concerns satisfying the downward closure property.

In this research, frequent pattern mining approaches with weight constraints are suggested. Our main approach is to push weight constraints into the pattern growth algorithm while maintaining the downward closure property. We develop WFIM (Weighted Frequent Itemset Mining with a weight range and a minimum weight), WLPMiner (Weighted frequent Pattern Mining with length decreasing constraints), WIP (Weighted Interesting Pattern mining with a strong weight and/or support affinity), WSpan (Weighted Sequential pattern mining with a weight range and a minimum weight) and WIS (Weighted Interesting Sequential pattern mining with a similar level of support and/or weight affinity)

The extensive performance analysis shows that suggested approaches are efficient and scalable in weighted frequent pattern mining.

DEDICATION

To my God, parents, wife, son, and daughter for their love, patience, and encouragement.

ACKNOWLEDGEMENTS

I could not have accomplished my doctoral degree without the support of many great scholars at Texas A&M University. First of all, I would like to express my deep gratitude to my committee chair, Dr. John J. Leggett. His invaluable advice and patience allowed me to undergo and finish the challenging process of my doctoral degree program. I would also like to express my appreciation to my other committee members: Dr. Richard Furuta, Dr. Frank M. Shipman III, and Dr. Joobin Choobineh. I also would like to thank all the graduate students in CSDL, whose advice and friendship gave me a lot of encouragement for my study at Texas A&M University. Finally, I am very grateful to my parents, wife, son and daughter for their continuous moral support and encouragement.

TABLE OF CONTENTS

Page

Page

Page

Page

LIST OF FIGURES

Page

Page

Page

LIST OF TABLES

Page

1. INTRODUCTION

Data mining, also known as Knowledge Discovery or Knowledge Discovery in Databases (KDD), is defined as the process of nontrivial extraction of previously unknown and potentially useful information from data in databases [13], [31], [32]. Data mining discovers patterns hidden in data and associations between the patterns. Frequent pattern mining plays an essential role in many data mining tasks such as mining association rules [2], [3], [21], [29], [47], [65], [82], [92], mining correlations [11], [17], [38], [40], [44], [51], [73], [84], [86], [87], mining closed patterns [19], [24], [36], [39], [54], [55], [56], [64], [74], [75], [76], [77], [88], mining sequential patterns [1], [4], [14], [15], [16], [18], [33], [41], [42], [50], [59], [60], [62], [63], [69], [81], [89], [93], and so on. First introduced in 1993 by Argawal et al. [2], frequent pattern mining [20], [27], [34], [46], [48], [58], [68], [78], [95] has been one of the hot issues in the data mining field. Mining information and knowledge from very large databases is not easy work. It takes a long time to process the large datasets and the amount of discovered knowledge and number of patterns can also be very large. It is also well known that frequent pattern mining generates a very large number of frequent itemsets and association rules. Researchers have proposed frequent pattern mining algorithms that run faster than previous algorithms and generate fewer but more important patterns. Many techniques such as depth first [5], [34], [35], [40], [46], [56], [58], [77] / breadth first search [2], [3], [4], [11], [29], [65], [69], use of trees [34], [49], [56], [77] / other data structures [15],

---

This dissertation follows the style of *IEEE Transactions on Knowledge and Data Engineering.*

[23], [28], [37], [58], [85], [94], [95], top down [22], [49], [80] / bottom up traversal [27], [49], [56], [58], [66], [78], vertical [1], [92], [93], [94], [95] / horizontal [2], [11], [34], [56], [83], [86] formats, and use of constraints [6], [7], [8], [9], [10], [25], [30], [43], [45], [52], [57], [61], [67], [68], [70], [78], [79] have been developed. Most frequent pattern mining algorithms use a support measure (defined in Section 2) to prune the combinatorial search space. However, support-based pruning is not enough when considering the characteristics of real datasets. Additionally, after mining datasets to obtain the frequent patterns, there is no way to adjust the number of frequent patterns through user feedback, except for changing the minimum support. Alternative measures [40], [53], [71], [73], [84], [86] for mining frequent patterns have been suggested to address these issues. One of the main limitations of the traditional approach for mining frequent patterns is that all items are treated uniformly when, in reality, items have different importance. For this reason, weighted frequent pattern mining algorithms [12], [72], [82] have been suggested that give different weights to items according to their significance. The main focus in weighted frequent pattern mining concerns satisfying the downward closure property [2], [4].

In this research, we suggest efficient and scalable frequent pattern mining approaches with weight constraints. Our main approach is to push the weight constraints into the pattern growth algorithm while maintaining the downward closure property.

First, we propose WFIM (Weighted Frequent Itemset Mining with a weight range and a minimum weight) which is based on the pattern growth algorithm. A weight range and a minimum weight constraint are defined and items are given different

weights within the weight range. The weight and support of each item are considered separately for pruning the search space. WFIM uses an ascending weight ordered prefix tree and the tree is traversed using a bottom-up strategy. In scanning the transaction database, the maximum weight in the transaction database is used to prune weighted frequent itemsets. In the mining step, the minimum weight of a prefix in the FP-trees is used for pruning. By doing so, the downward closure property can be retained. The number of weighted frequent itemsets can be reduced by setting a weight range and a minimum weight, allowing the user to balance support and weight of itemsets. The WFIM is the first weighted frequent itemset mining approach that uses a pattern growth algorithm.

Second, we re-examine two basic but interesting constraints, a weight constraint and a length decreasing support constraint and propose WLPMiner (weighted frequent pattern mining with length decreasing constraints). WLPMiner integrates these two measures to generate fewer and more meaningful patterns. For pruning techniques, we use the notion of WSVE (Weighted Smallest Valid Extension) to apply to both the length decreasing support constraints and weight constraints, and a weight range as a supplement to maintain the downward closure property. The key insights achieved in this approach are the high performance of the WSVE property and the use of a weight range in the weight constraint. We show that combining a weight constraint with a length decreasing support constraint improves performance in terms of the number of patterns and runtime.

Third, we present a WIP (Weighted Interesting Pattern mining with a strong weight and/or support affinity) algorithm that integrates the strengths of the previous techniques and generates weighted interesting patterns according to user feedback. In WIP, a new measure, weight confidence, is defined to generate weighted hyperclique patterns with similar levels of weights. A weight range is used to decide weight boundaries and the h-confidence measure serves to identify strong support affinity patterns. WIP not only gives a balance between the two measures of weight and support, but also considers weight affinity and/or support affinity between items within patterns so more valuable patterns can be generated. To our knowledge, ours is the first work specifically to consider a level of weight affinity between items of patterns and to incorporate a weight confidence with an h-confidence.

Fourth, we developed weighted sequential pattern mining algorithms: WSpan (Weighted Sequential pattern mining with a weight range and a minimum weight) and WIS (Weighted Sequential pattern mining with a similar level of support and/or weight affinity). WSpan focuses on weighted frequent pattern mining based on the prefix projected sequential pattern growth approach. A weight range and a minimum weight are used to adjust the number of sequential patterns. In WIS, we define sequential s-confidence and w-confidence measures and the concept of weighted interesting sequential patterns by using the two measures. Our main goal in this framework is to push sequential s-confidence and/or w-confidence into the weighted sequential pattern mining algorithm based on the pattern growth method. The sequential s-confidence and/or w-confidence measures can be used to avoid generating spurious sequential

patterns that involve items from different support and/or weight levels. A comprehensive performance study shows that these algorithms are efficient and scalable in weight-based pattern mining. Moreover, it generates fewer but important patterns.

1.1 Motivation

Mining the important, correlated patterns efficiently is one of the main goals in data mining. Most of the weighted frequent pattern mining or association rule mining algorithms such as [12], [47], [72], [82] have adopted an Apriori algorithm based on the downward closure property [2]. They have suggested the sorted closure property [47], the weighted closure property [72] or other techniques [12], [82] in order to satisfy the downward closure property. Three main limitations exist in the previous weighted frequent pattern mining algorithms.

First, Apriori based algorithms use candidate set generation and test approaches. It can be very expensive to generate and test all the candidates. Performance analyses [21], [26], [96] have shown that frequent pattern growth algorithms are efficient at mining large databases and more scalable than Apriori-based approaches. However, there has been no weight based mining using the pattern growth algorithm because the downward closure property is broken by simply applying the FP-growth methodology. We will develop weighted frequent pattern mining based on the pattern growth method.

Second, frequent pattern mining algorithms such as [27], [46], [49], [58], [66], [78] have better performance when a minimum support is high and the database is sparse. The main problem with these algorithms is that they can still generate an exponentially large number of patterns when a minimum support becomes lower. As the

number of frequent patterns increases, the performance becomes worse. Although closed pattern mining approaches such as [9], [19], [24], [54], [55], [56], [77], [79], [94] are used, huge frequent patterns are still generated in large dense databases with a low minimum support. This problem also occurs in weight-based mining. We will describe ways to adjust the number of patterns through user feedback.

Third, previous weighted frequent pattern mining approaches use weight as one of the measures. The items are given different weights in the transaction database. However, they only focus on how to maintain the downward closure property. Although patterns satisfy pruning conditions used in previous weighted pattern mining algorithms [12], [72], [82], they may have different characteristics, such as different levels of support and different levels of weight. We will suggest efficient and scalable mining methods in which users can decide their levels of interest and give direction for mining their own interesting patterns.

1.2 Organization

The remainder of this dissertation is structured as follows: In Section 2, we describe the frequent pattern mining problem and related work. In Section 3, WFIM (Weighted Frequent Itemset Mining with a weight range and a minimum weight) is developed. In Section 4, we propose WLPMiner (Weighted frequent pattern mining with length decreasing support constraints), which uses advantages of WFIM and applies length decreasing support constraints. Section 5 presents a WIP algorithm that mines weighted interesting patterns with a strong weight and/or support affinity. In Section 6, we show weighted sequential pattern mining algorithms: WSpan (Weighted Sequential

pattern mining with a weight range and a minimum weight) and WIS (Weighted Sequential pattern mining with a similar level of support and/or weight affinity). As further extensions, in Section 7, the applications of weighted pattern mining are shown. Finally, conclusions are presented in Section 8.

2. PROBLEM DEFINITION AND RELATED WORK

2.1 Problem definition

The frequent pattern mining problem was first suggested by R. Agrawal, et al [2] as mining association rules between sets of items. Let $I = \{i_1, i_2... i_n\}$ 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 tid and a set of items. A pattern is called a k-pattern if it contains k items. A pattern $\{x_1, x_2,.., x_n\}$ is also represented as $x_1, x_2,.., x_n$. The support of a pattern is the number of transactions containing the pattern in the database. A weight of an item is a non-negative real number that reflects the importance of the item in the transaction database. We use the term, weighted itemset to represent a set of weighted items. A weight is given to an item within a weight range, $W_{min} \leq WR \leq W_{max}$. The problem of weighted frequent pattern mining is to find the complete set of patterns satisfying a support constraint and a weight constraint in the database.

2.2 Related work

Several means of mining frequent patterns have been explored in previous research and this area is currently very active in the data mining field.

2.2.1 Apriori algorithm

The Apriori algorithm [3] was developed by Agrawal et al in 1994. To improve performance, an anti-monotone property (downward closure property) [4] of frequent patterns was suggested by Agrawal. That is, if any length k pattern is not frequent in a

transaction database, superset patterns can not be frequent. Based on this property, many frequent pattern mining algorithms have been developed to prune infrequent patterns more efficiently. However, two main problems [34], [35], [96] exist with this approach. First, Apriori based algorithms must scan the transaction database multiple times. As the size of the database increases, the number of scans also increases. Multiple scans of the database are very costly. Therefore, one of the goals is to reduce the number of database scans. Second, the Apriori algorithm generates a large number of candidates. Keeping and counting these candidates to generate actual frequent patterns is also time and space consuming. To reduce the number of candidates and speed up the algorithms, the pattern growth approach [34], [35] was suggested.

2.2.2 Pattern growth approach

Jiawei Han [34], [35] suggested a new data structure, the FP-tree (Frequent Pattern tree) for generating candidates. The FP-tree is an extended prefix tree in which compressed data from the original database is stored. He also developed a pattern growth method to avoid the candidate generation and test operation of the Apriori-based algorithm. The pattern growth method also uses the anti-monotone property, but it recursively divides the database into sub-databases according to the frequent patterns already discovered and finds local frequent patterns using local FP-trees. FP-tree based methods mine the complete set of frequent patterns using a divide and conquer method to reduce the search space without generating all the candidates. The FP-tree based approaches are faster than the Apriori based approaches. Based on the pattern growth

method, many algorithms [27], [34], [40], [46], [56], [58], [66], [77] have been developed.

2.2.3 Weighted frequent pattern mining

Weight constraints [12], [72], [82] have recently been introduced into frequent pattern mining. Different weights are given to items according to their importance or intensity. In contrast to previous frequent pattern mining approaches which are mainly based on support constraints, weighted frequent mining approaches consider not only the frequency but also the importance of patterns. The main focus of weighted frequent itemset mining concerns the downward closure property. The downward closure property is usually broken when different weights are applied to the items. All weighted association rule mining algorithms suggested so far have been based on the Apriori algorithm.

2.2.3.1 MINWAL (Mining association rules with weighted items)

In MINWAL [12], C. H. Cai defined a weighted support measure which is calculated by multiplying the support of a pattern with the average weight of a pattern. MINWAL defined an upper bound, called k-support, to maintain the downward closure property. Support of patterns generated in level k must be greater than or equal to the k-support bound. MINWAL is based on the Apriori algorithm but most of the candidates are infrequent. According to our performance analyses, it takes too long to use the k-support bound for satisfying the downward closure property.

2.2.3.2 WAR (Efficient mining of Weighted Association Rules)

WAR (Weighted Association Rules) [82] generates frequent items without considering weights and then does post-processing during the rule generation step. The WAR algorithm is not concerned with mining frequent itemsets and is a post-processing approach. Additionally, the WAR is also based on the Apriori algorithm.

2.2.3.3 WARM (Weighted Association Rule Mining)

In WARM (Weighted Association Rule Mining) [72], the problem of breaking the downward closure property is solved by using a weighted support and developing a weighted downward closure property. However, the meaning of weighted support is different from that defined in MINWAL [12]. Weighted support of a pattern AB in the WARM is the fraction of the weight of the transactions containing both A and B to the weight of all transactions. That is, WARM does not consider the support measure. This algorithm is also based on the Apriori algorithm.

2.2.4 Frequent pattern mining with other measures / constraints

The use of measures and constraints permits user guidance and leads to effective pruning of the search space and efficient mining of frequent patterns. Frequent pattern mining approaches with other measures and constraints represents an important direction for user controlled data mining. Therefore, new measures [40], [53], [71], [73], [85], [86] and constraints [6], [7], [8], [9], [10], [25], [30], [43], [45], [52], [57], [61], [66], [67], [70], [78], [79] have been proposed extensively.

2.2.4.1 Hyperclique miner

Hyperclique Miner [86] defines hyperclique patterns that reflect the overall support affinity among items within the pattern. A hyperclique pattern is a frequent pattern that contains items that are highly affiliated with each other in terms of support. The Hyperclique Miner uses an h-confidence measure. An h-confidence of a pattern is the minimum ratio of the support of this pattern to the support of one of the items in this pattern. H-confidence is used to reflect the overall affinity among items within the pattern. A frequent pattern is called a hyperclique pattern if h-confidence of the pattern is no less than a minimum h-confidence. According to our performance test, there are many cases where the number of hyperclique patterns is not increased or decreased when the minimum support is changed and the number of hyperclique patterns is very large. This can be true even if the minimum h-confidence is increased. In addition, the Hyperclique Miner has adopted an Apriori algorithm based on the downward closure property and it only focuses on support based pruning.

2.2.4.2 LPMiner

LPMiner [66] is the first algorithm to find patterns that satisfy a length decreasing support constraint. It defines and uses the Smallest Valid Extension (SVE) property [66]. Assigning weights according to the importance of the items is a main consideration for real datasets. However, LPMiner dose not consider the importance of the items even though it takes into account length decreasing support constraints.

2.2.4.3 BAMBOO

BAMBOO [79] pushes the length decreasing support constraint deeply into closed itemset mining in order to generate more concise itemsets. While BAMBOO outperforms LPMiner, it is only concerned with the length decreasing support constraint. In other words, BAMBOO does not consider different weights of items within patterns.

3. WFIM: WEIGHTED FREQUENT ITEMSET MINING WITH A WEIGHT RANGE AND A MINIMUM WEIGHT*

3.1 Overview of WFIM

We propose an efficient algorithm called WFIM (Weighted Frequent Itemset Mining) based on the pattern growth approach. Our main goal is to balance support and weight of itemsets, push the weight constraints into the pattern growth algorithm, maintain the downward closure property, and provide a way to adjust the number of itemsets and runtime [90]. A weight range and a minimum weight constraint are defined and items are given different weights within the weight range. WFIM uses an ascending weight ordered prefix tree and the tree is traversed using a bottom-up strategy. In scanning the transaction database, the maximum weight in the transaction database is used to prune weighted frequent itemsets and in the mining step, the minimum weight of a prefix in FP-trees is used. By doing so, the downward closure property can be retained. The weight and support of each item are considered separately for pruning the search space. The number of weighted frequent itemsets can be reduced by setting a weight range and a minimum weight, allowing the user to balance support and weight of itemsets. WFIM generates smaller but important weighted frequent itemsets in large databases, particularly dense databases with low minimum support.

WFIM provides the following contributions: 1) In the WFIM, individual items are given different weights within weight range to reflect their importance, and weight constraints are pushed into the pattern growth algorithm keeping the downward closure property. 2) We can adjust the number of weighted frequent itemsets by changing parameters such as a weight range and a minimum weight although a minimum support is lower in the dense database or long database. 3) In WFIM, the weight and support of each item are considered separately for pruning the search space. WFIM allows the user to balance support and weight of itemsets. 4) We do extensive performance evaluation and analyze the relationship between a weight and a support.

3.2 Weighted frequent itemset mining

We suggest an efficient weighted frequent itemset mining algorithm in which the main approach is to push weight constraints into the pattern growth algorithm and provide ways to keep the downward closure property. We show our approach has the benefit of considering a minimum support and a minimum weight separately, so the user can balance itemset support and weight. WFIM uses the bottom-up divide and conquer method in mining weighted frequent itemsets. In general, a descending ordered prefix tree and bottom up traversal or ascending ordered prefix tree and top down traversal are used together. However, our algorithm adopts an ascending weight ordered prefix tree. The tree is traversed bottom-up because the previous matching can not maintain the downward closure property. A support of each itemset is usually decreased as the length of an itemset is increased, but the weight has a different characteristic. An itemset which has a low weight sometimes can get a higher weight after adding another item with a

higher weight, so it is not guaranteed to keep the downward closure property. For instance, assume that the minimum support is 3, the support of item “A” is 2, the support of itemset “AB” is also 2, a weight of item “A” is 1 and a weight of item “B” is 2. The weighted support of item “A” is 2 and the weighted support of itemset “AB” is 3. We can not prune item “A” even if the weighted support (2) of item “A” is less than minimum support (3) because the weighted support of itemset “AB” is equal to the minimum support and itemset “AB” is a weighted frequent itemset. To tackle this problem, frequent prefix trees are constructed by weight ascending order and these trees are traversed in bottom up.

We present our algorithm in detail and show actual examples in order to illustrate the steps in the FP-tree construction for weighted frequent itemset mining and the mining of a weighted frequent itemset from the FP tree.

Table 1. A transaction database TDB as a running example in WFIM

TID	**Set of items**	**Frequent Item list**
100	a, c, d, f, i, m	c, d, f, m
200	a, c, d, f, m, r	c, d, f, m, r
300	b, d, f, m, p, r	d, f, m, p, r
400	b, c, f, m, p	c, f, m, p
500	c, d, f, m, p, r	c, d, f, m, p, r
600	d, m, r	d, m, r

Definition 3.1 Weight Range (WR)

The weight of each item is assigned to reflect the importance of each item in the transaction database. A weight is given to an item with a weight range, $W_{min} \leq W \leq W_{max}$.

Definition 3.2 Minimum weight threshold (min_weight)

In the WFIM, we want to give a balance between the two measures of weight and support. Therefore, we define a minimum weight constraint like a minimum support in order to prune items which have lower weights.

Definition 3.3 Maximum Weight (MaxW)

The maximum weight (MaxW) is defined as the value of the maximum weight of items in a transaction database or a conditional database. In WFIM, a MaxW is used in a transaction database.

Definition 3.4 Minimum Weight (MinW)

The minimum weight (MinW) is defined as the value of the minimum weight of items in a transaction database or a conditional database. In WFIM, a MinW is used in a conditional database.

Definition 3.5 Useless itemset

An itemset X is called a useless itemset if the support of the itemset X is less than a minimum support (min_sup) and its weight is also less than a minimum weight.

Example 3.1: Table 1 shows transaction database TDB. Table 2 shows example sets of items with different weights. The minimum support (min_sup) is 3 and the frequent list is: Frequent_list = <a:2, b:2, c:4, d:5, f:5, i:1, m:6, p:3, r:4>. The MinW of a WR is the minimum weight value of the WR and the MaxW of a WR is the maximum weight value

of the WR. For example, the MinW of WR_1 is 1.0, MinW of WR_2 is 0.7 and the MaxW of WR_1 is 1.5, the MaxW of WR_2 is 1.3 and so on. If a minimum support is only considered, the second columns in Table 1, give the frequent item list.

Table 2. Example sets of items with different weight ranges

Item (min_sup = 3)	**a**	**b**	**c**	**d**	**f**	**i**	**m**	**p**	**r**
Support	2	2	4	5	5	1	6	3	4
Weight ($1.0 \leq WR_1 \leq 1.5$)	1.3	1.1	1.4	1.2	1.5	1.1	1.3	1.0	1.5
Weight ($0.7 \leq WR_2 \leq 1.3$)	1.1	1.0	0.9	1.0	0.7	0.9	1.2	0.8	1.3
Weight ($0.7 \leq WR_3 \leq 0.9$)	0.85	0.75	0.8	0.9	0.75	0.7	0.85	0.7	0.9
Weight ($0.2 \leq WR_4 \leq 0.7$)	0.5	0.3	0.6	0.4	0.7	0.3	0.5	0.2	0.7

Definition 3.6 Weighted Frequent Itemset (WFI)

An itemset X is a weighted infrequent itemset if, following pruning, condition 3.1 or condition 3.2 below is satisfied. If the itemset X does not satisfy both of these, the itemset X is called a weighted frequent itemset.

Pruning condition 3.1 (support < min_sup && weight < min_weight)

The support of an itemset is less than a minimum support and the weight of an itemset is less than a minimum weight constraint.

For finding useless items, we apply pruning condition 3.1. When a weight and a support are considered separately, there are four cases for each item: a high support and a high weight, a high support and a low weight, a low support and a high weight and then a low support and a low weight. In definition 3.5, we defined items which have a low support and a low weight as useless items. The items which have a low support and

a low weight can be pruned because these items have low frequencies and low importance. However, the items having other cases can not be pruned because these items may have higher priority although the support of the itemset is low or the itemset may have higher frequency even if the weight of the itemset is low.

Pruning condition 3.2 (support * MaxW (MinW) < min_sup)

In a transaction database, the value of multiplying itemset's support with a MaxW among items in the Transaction database is less than a minimum support. In conditional databases, the value of multiplying the support of an itemset with a MinW of a conditional pattern in the FP-trees is less than a minimum support.

Definition 3.7 Weighted association rule mining

Weighted association rule mining finds a complete set of association rules in the transaction database with support and weight constraints. The weight for each item is assigned within a weight range, $W_{min} \leq W \leq W_{max}$.

Lemma 3.1 When two conditions are applied to prune weighted infrequent itemsets, the case in which only pruning condition 3.1, but not pruning condition 3.2, is satisfied for pruning weighted infrequent itemset, is that a MaxW of a transaction database or a MinW of a conditional pattern in the FP-tree should be greater than one.

Proof: In this case, pruning condition 3.1, but not pruning condition 3.2 in the definition 3.6, should be satisfied in order to prune an itemset. That is, in condition 3.1, the support of an itemset is less than a min_sup and the weight of an itemset is less than a min_weight. However, the value of multiplying the itemset's support with a MaxW

(MinW) of an itemset should be greater than or equal to a minimum support. We can see that the following two formulas should be satisfied.

Formula 1: support < min_sup

Formula 2: support * MaxW (MinW) ≥ min_sup

We can know that the a MaxW of a transaction database or a MinW of a conditional pattern in the FP-tree must be greater than or equal to one in order to satisfy both of the formulas. For example, assume that a min support is 5, a minimum weight threshold is 0.8, a support of an itemset is 4, the weight of an itemset is 0.7 and the MaxW of an itemset in TDB is 1.3. We know that the pruning condition 3.1 is satisfied but the pruning condition 3.2 is not satisfied. Therefore, this itemset is pruned by the condition 3.1 in definition 3.6.

Lemma 3.2 There is no limitation to use the pruning condition 3.2 of definition 3.6. That is, the pruning condition 3.2 (support * MaxW (MinW) < min_sup) is can be applied without any limitations.

When only pruning condition 3.2, but not pruning condition 3.1 of definition 3.6 is satisfied to prune weighted infrequent itemsets, a MaxW (Maximum Weight) of a transaction database or a MinW (minimum weight) of a conditional pattern in the FP tree can be any value.

Proof: In this case, an itemset is pruned since pruning condition 3.2 is satisfied although condition 3.1 is not satisfied. We can see that the following two formulas should be satisfied

Formula 3: (support ≥ min_sup || weight ≥ min_weight)

Formula 4: (support * MaxW (MinW) < min_sup)

If a support of an itemset is greater than or equal to a minimum support in the Formula 3, MaxW (MinW) should be less than one to satisfy Formula 4. However, if a weight of an itemset is greater than or equal to a minimum weight threshold in the formula 3 and the support of an itemset is less than a minimum support, there is no relationship between Formula 3 and Formula 4. In other words, pruning condition 3.2 (support * MaxW (MinW) < min_sup) can be applied without any limitations.

Lemma 3.3 When two pruning conditions are applied to prune weighted infrequent itemsets, the method to use two pruning conditions, always prunes more than the approach to use only a minimum support when a MaxW of the transaction database or a MinW of the FP tree is less than one.

Proof: In the normal frequent itemset mining, every item has the same priority. That is, their weights are 1.0. If pruning condition 3.2 is only considered, we can understand that more items or itemsets will be pruned when weights of items are set as less than one. For example, assume that a minimum support is 4 and the support of an itemset is 5. In the normal frequent itemset mining, the itemset is not pruned since a weight of all the itemset is 1.0 and the support of the itemset is greater than a minimum support. However, the itemset is pruned when the weight of the itemset is 0.7 by condition 3.2 in definition 3.6.

Example 3.2: The columns in Table 3 show the set of weighted frequent itemsets after pruning weighted infrequent itemsets using pruning condition 3.1 and pruning condition 3.2 in definition 3.6 by applying different WRs. For example, when WR_3 is applied, item

p's support is 3, MaxW is 0.9 and the value (2.7) of multiplying item's support (3) with a MaxW (0.9) of an itemset in the TDB is less than minimum support (3) so item "p" can be removed. Meanwhile, the number of WFI can be increased when WR_1 is used as the weight range. The support of item "a" in the transaction database is 2. However, the value (3) of multiplying the support (2) of item "a" with a MaxW (1.5) of an itemset is equal to a minimum support (3) so this item is added in the WFI list.

Table 3. Weighted frequent itemsets with different weight ranges in WFIM

TID	**WFI list** $(0.2 \leq WR_4 \leq 0.7)$ MinW = 0.2	**WFI list** $(0.7 \leq WR_3 \leq 0.9)$ MinW = 0.7	**WFI list** $(0.7 \leq WR_2 \leq 1.3)$ MinW = 0.7	**WFI list** $(1.0 \leq WR_1 \leq 1.5)$ MinW = 1.0
100	d, f, m	c, d, f, m	c, d, f, m	a, c, d, f, m
200	d, f, m	c, d, f, m, r	c, d, f, m, r	a, c, d, f, m, r
300	d, f, m	d, f, m, r	d, f, m, p, r	b, d, f, m, p, r
400	f, m	c, f, m	c, f, p, m	b, c, f, p, m
500	d, f, m	c, d, f, m, r	c, d, f, m, p, r	c, d, f, m, p, r
600	d, m	d, m, r	d, m, r	d, m, r

Example 3.3: Let us show another example by changing a WR and a min_weight. In this example, assume that the definitions of Table1 and Table 2 are used and pruning condition 3.1 in definition 3.6 is applied using WR_2 as a weight range. If the min_weight is 1.2, items "a", "b" and "i" are pruned because the support of these items is less than a minimum support and the weight of these items is also less than a minimum weight. In a similar way, we can get the following results by changing min_weight. If min_weight is 1.1, items "i" and "b" are pruned and if min_weight is 1.0, item "i" is pruned. As a

result, the number of weighted frequent items can be changed according to different min_weights. When pruning condition 3.2 of definition 3.6 is considered, if a weight range is $1.0 \leq WR_1 \leq 1.5$, only item "I" is pruned because the value of multiplying item i's support (1) with a MaxW (1.5) is less than a minimum support (3). However, the items "a" and "b" are not pruned because the value of multiplying the support (2) of item "a" and "b" with a MaxW (1.5) is equal to a minimum support (3).

In a similar fashion, we can get the following results by changing WRs. If the weight range is $0.7 \leq WR_2 \leq 1.3$, item "a", "b", and "i" are pruned. If the weight range is $0.7 \leq WR_3 \leq 0.9$, item "a", "b", "i" and "p" are pruned and if the weight range is $0.2 \leq WR_4 \leq 0.7$, item "a", "b", "c", "i", "p" and "r" are pruned. Thus, the number of weighted frequent items can be adjusted by using the different WRs.

3.3 WFIM (Apriori): WFIM based on Apriori algorithm

Before developing WFIM based on pattern growth approach, we describe WFIM based on Apriori approach. We apply a weight range and a minimum weight threshold in Apriori based approach. Therefore, the performance can be improved by efficiency of a weight range and a minimum weight. We will compare WFIM (Apriori) with WFIM based pattern growth approach which will be described. In WFIM (Apriori), the MaxW of items in transaction database is calculated, if the value of multiplying the support of an itemset with the MaxW among items in the transaction database is less than a minimum support, the itemset is pruned. Now, we summarize the weighted frequent itemset mining process based on Apriori approach and show the mining algorithm.

ALGORITHM. [WFIM (Apriori)]: Weighted Frequent Itemset Mining with a weight range and a minimum weight based on Apriori algorithm in a large transaction database.

Input: (1) A transaction database : TDB,

(2) Minimum support threshold : min_sup,

(3) Weights of the items within weight range : w_i,

(4) Minimum weight threshold: min_weight.

Output: The complete set of WFI (Weighted Frequent Itemsets).

Method:

1. Scan TDB once and find the weighted frequent items satisfying the following definition: An itemset X is a weighted frequent itemset if the following pruning conditions 1.1 and 1.2 are not satisfied.

Condition 1.1: (support < min_sup && weight < min_weight)

Condition 1.2: (support * MaxW < min_sup)

2. For the size of itemsets in (2, 3, … , k-1) do

a. In subsequent pass called pass k, generate candidate itemsets, C_k using the weighted frequent itemsets found in the (k-1) pass. WFIM (Apriori) uses WFI_{k-1} * WFI_{k-1} to find a candidate set of itemsets C_k in which * means concatenation operation.

b. Next, the database is scanned again and the support of candidates, C_k are counted.

c. A candidate C_k is a weighted frequent itemset (WFI_k) if the following pruning conditions 2.1 and 2.2 are not satisfied.

Condition 2.1: (support < min_sup && weight < min_weight)

Condition 2.2: (support * MaxW < min_sup)

d: Prune the weighted infrequent itemsets in C_k. if the candidate, C_k is not satisfying the condition 2.1 and condition 2.2.

3. Generate all WFI (Weighted Frequent Itemsets).

WFIM (Apriori) is based on Apriori algorithm so it need candidate generations and test. The number of weighted frequent itemsets can be reduced by setting a weight range and a minimum weight, allowing the user to balance between the two measures of weight and support of itemsets. However, the overhead of generating all candidates and test them is so huge. Therefore, we propose efficient and scalable algorithm, WFIM (Weighted Frequent Itemset Mining) based on pattern growth method.

3.4 WFIM based on pattern growth algorithm

3.4.1 FP (Frequent Pattern) tree structure

Table 4. WFI after pruning and sorting with a WR: 0.7–0.9 in WFIM

TID	weight ascending order item list (item: weight)
100	(f: 0.75) (c: 0.8) (m: 0.85) (d: 0.9)
200	(f: 0.75) (c: 0.8) (m: 0.85) (d: 0.9) (r: 0.9)
300	(f: 0.75) (m:0.85) (d: 0.9) (r: 0.9)
400	(f: 0.75) (c: 0.8) (m: 0.85)
500	(f: 0.75) (c: 0.8) (m: 0.85) (d: 0.9) (r: 0.9)
600	(m: 0.85) (d: 0.9) (r: 0.9)

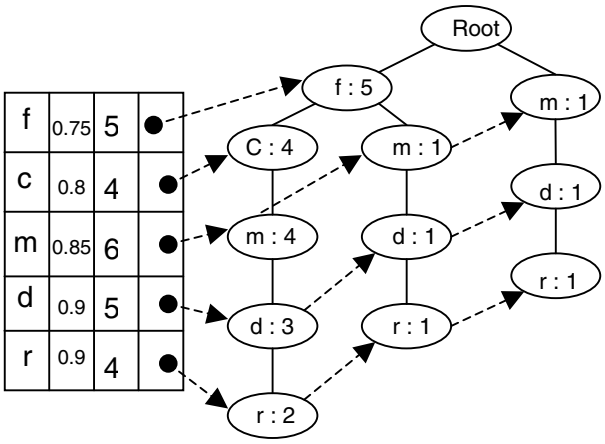

Figure 1. The global FP tree in WFIM.

The FP-trees in our 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 this, sort the items in weight ascending order. Although supports of items may be lower than the minimum support and infrequent, the items can not be deleted since infrequent items may become weighted itemsets in the next step. The weighted infrequent items are removed according to pruning conditions 3.1 and 3.2 in definition 3.6. For instance, assume that WR_3 is used as a WR, min_sup is 3 and min_weight is 0.7. Then, items “a”, “b”, “p”, and “i” are removed. Table 4 shows the result of removing weighted infrequent itemsets and sorting them by weight ascending order. When an item is inserted in the FP-tree, as already discussed, a weighted infrequent item is removed and the rest, weighted frequent items and itemsets, are sorted by weight ascending order. As shown in [34], [35], each node in the FP-tree has item-id, a weight, count and node link. Separate header tables exist for each FP tree and there is an entry for each item in the

header table. Figure 1 presents the global FP-tree and corresponding header table for this example in WFIM.

3.4.2 Bottom up divide and conquer

After a global FP-tree is generated from the transaction database, WFIM mines frequent itemsets from the FP-tree. The weighted frequent itemsets are generated by adding items one by one. WFIM adapts the divide and conquer approach for mining weighted frequent itemsets. It divides mining the FP-tree into mining smaller FP trees. In the above example of Figure 1, WFIM mines (1) the patterns containing item "r" which has the highest weight, (2) the patterns including "d" but not "r", (3) the patterns containing "m" but no "r" nor "d", (4) the patterns containing "c" but no "r", "d" nor "m", and finally the patterns containing item "f". WFIM can find all the subsets of weighted frequent itemsets. To begin with, for node r, we generate a conditional database by starting from r's head and following r's node link. The conditional database for prefix "r:4" contains three transactions: <dmcf:2>, <dmf:1> and <dm:1>. Previous FP-growth algorithms only consider a support in each conditional database so an item "c" is only pruned because the support (2) of item "c" is less than a minimum support (3). However, in WFIM, before constructing the FP-tree, conditions in definition 3.6 are applied in order to check whether it is a weighed frequent itemset. From Table 2, we know that the weights of item "r", "d", "m", "c", and "f" are 0.9, 0.9, 0.85, 0.8 and 0.75 respectively. There is only one item "r" in the conditional pattern, so the MinW of the conditional pattern is 0.9. By applying pruning conditions, we can see that not only an item "c" but also an item "f" are pruned because the value (2.7) of multiplying item f's

support (3) with a MinW (0.9) is less than a minimum support (3). Although the support of item “f” is equal to the minimum support, the item “f” with a lower weight can be deleted. After that, Fig. 2 (a) shows a FP- tree with prefix “r”. As you can see, the conditional FP-tree for prefix “r:4” is a single path tree <md:4>. We can generate all kinds of combinations of items including a conditional pattern “r:4”. That is, “r:4”, “rm”, “rd” and “rmd”.

The important point is that we can use a MinW of conditional patterns instead of the MaxW of the conditional database or the MaxW of a conditional pattern to prune weighted infrequent itemsets because WFIM uses an ascending weight ordered prefix tree structure to construct conditional databases and the MinW of a prefix is always greater than or equal to the weights of all the items in a conditional database. Hence, the MinW of a conditional pattern in the conditional database can be used for applying the pruning conditions.

For node “d”, WFIM derives a frequent pattern (d:5) and three paths in the FP-tree : <mcf:3>, <mf:1> and <m:1>. In the FP-growth algorithm, no item is pruned. However, in WFIM, item “c” is pruned in this conditional database with prefix “d”, since the value (2.7) of multiplying item c’s support (3) with the MinW (0.9) is less than a minimum support (3) although the supports of these items are greater than a minimum support. However, items “m” and “f” are not pruned. After removing weighted infrequent itemsets in the conditional database, the projected FP-tree for the prefix “d:5” is constructed.

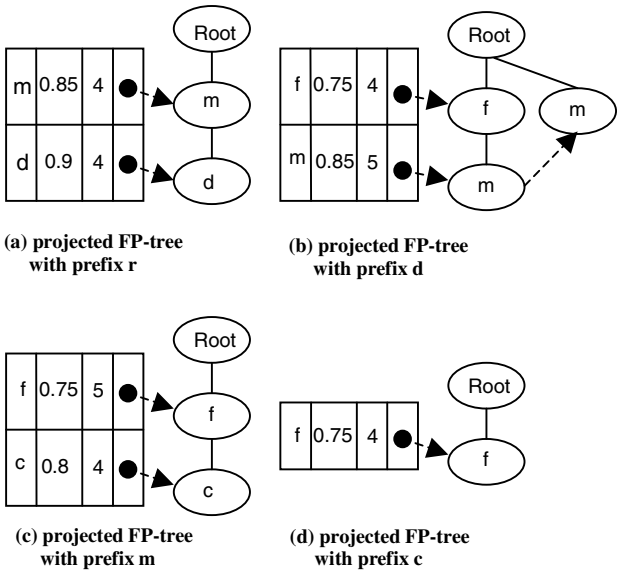

Figure 2. Conditional FP trees in WFIM.

Figure 2 (b) shows a conditional FP-tree for prefix d:5. It's not a single FP-tree, so we divide the conditional FP-tree to mine even smaller conditional FP-trees recursively. As a result, after a recursive process, we obtain weighted frequent itemsets based on conditional pattern "d": "d:5", "dm:5", "df:4" and "dmf:4". Similarly, we can build projected FP-trees from the global FP-tree and mine weighted frequent itemsets from them recursively. These FP-trees are shown in Figure 2 (c)-(d). (The FP-tree for prefix f:5 is empty and not shown here).

3.4.3 WFIM algorithm

WFIM can push weight constraints into the pattern growth algorithm and show how to keep the downward closure property. A weight range and a minimum weight are defined and items are given different weights within the weight range. Now, we

summarize the weighted frequent itemset mining process and present the mining algorithm.

ALGORITHM [WFIM]: Weighted Frequent Itemset Mining with a weight range and a minimum weight constraint in a large transaction database.

Input: (1) A transaction database: TDB,

(2) A minimum support threshold: min_sup,

(3) weights of the items within weight range: w_i,

(4) A minimum weight threshold: min_weight

Output: The complete set of weighted frequent itemsets.

Method:

1. Scan TDB once to find the global weighted frequent items satisfying the following definition: An itemset X is a weighted frequent itemset if the following pruning conditions 1.1 and 1.2 are not satisfied.

Condition 1.1: (support < min_sup && weight < min_weight)

Condition 1.2: (support * MaxW < min_sup)

2. Sort items in weight ascending order. The sorted weighted frequent item list forms the weight_order and header of FP tree.

3. Scan the TDB again and build a global FP-tree using weight_order.

4. Mine a global FP-tree for weighted frequent itemset mining in a bottom up manner. Form conditional databases for all remaining items in weight_list and complete local

weighted frequent itemsets for the conditional databases. (An itemset X is a weighted frequent itemset if the following pruning conditions 4.1 and 4.2 are not satisfied).

Condition 4.1: (support < min_sup && weight < min_weight)

Condition 4.2: (support * MinW < min_sup)

5. When all the items in the global header table have been mined, WFIM is finished.

3.5 Performance evaluation

In this section, we present our performance study over various datasets. As an initial work, WFIM (Apriori) is based on Apriori algorithm but it use a weight range and a minimum weight. The WFIM is the first weighted frequent itemset mining algorithm which uses a FP-growth algorithm and the extensive performance studies [21], [26], [96] showed that the performance of the FP-growth based algorithms is superior to that of Apriori-based mining algorithms in terms of runtime and scalability.

First, we compare WFIM with other algorithms such as MINWAL [12], WFIM (Apriori) and FP-growth [34]. We can find previous weight based mining algorithms [12], [72], [82]. However, WAR [82] does not consider a weight measure in mining frequent patterns. After getting these patterns, WAR considers a weight measure to generate weighted association rules. WARM [72] uses a weight measure but does not use a support measure. The main two improvement of WFIM are 1) new weight features, a weight range and a minimum weight and 2) use of a pattern growth method for weighted frequent pattern mining. In order to evaluate performance of two advantages, we compare WFIM with MINWAL, WFIM (Apriori) and FP-growth algorithm.

MINWAL is weighted frequent pattern mining algorithm to use k-support upper-bound to maintain downward closure property and FP-growth is the first algorithm used pattern growth approach in mining frequent patterns.

Second, we analyze the efficiency of the weight range and the minimum weight. We show how the number of weighted frequent itemset and runtime can be reduced by adjusting the weight range and the minimum weight. Finally, we show how WFIM has good scalability against the number of transactions in the datasets.

Table 5. Characteristics of real and synthetic datasets

Data sets	Size	#Trans	#Items	A.(M.) t. l.
Connect	12.14M	67557	150	43 (43)
Mushroom	0.83M	8124	120	23 (23)
Pumsb	14.75M	49046	2113	74 (74)
BMS-WebView-1	1.28M	59601	497	2.51 (267)
T10I4D100K	5.06M	100K	1000	10 (31)
T10I4Dx	10.12- 50.6M	200K-1000K	1000	10 (31)

3.5.1 Test environment and datasets

In our experiments, we used three real datasets and several synthetic datasets which have been used in pervious experiments [34], [46], [49], [56], [58], [79]. Table 5 shows the characteristic of these datasets. The three real datasets used are Connect, Mushroom and BMS-webView-1 respectively. The Connect dataset is very dense and includes game state information. The Mushroom dataset is a little dense and contains

characteristics of various species of mushrooms. The BMS-webView-1 dataset is a very sparse dataset with a web log. These real datasets can be obtained from the UCI machine learning repository (http://www.ics.uci.edu/~mlearn) and the frequent itemset mining dataset repository (http://fimi.cs.helsinki.fi/data/).

The synthetic datasets were generated from the IBM dataset generator (http://www.almaden.ibm.com/software/quest/Resources/datasets/syndata.html). We use T10I4D100k dataset which is very sparse and contains 100,000 transactions. However, the synthetic datasets T10I4Dx contain 100k to 1000k transactions. WFIM and WFIM (Apriori) are written in C++. Experiments were performed on a sparcv9 processor operating at 1062 MHz, with 2048MB of memory. All experiments were performed on a Unix machine. In our experiments, a random generation function generated weights within the weight range for each item. When running algorithms, a weight range and a minimum weight are set up as the cut off values by users for weighted frequent patterns.

3.5.2 Experimental results

3.5.2.1 Comparison with WFIM and other algorithms

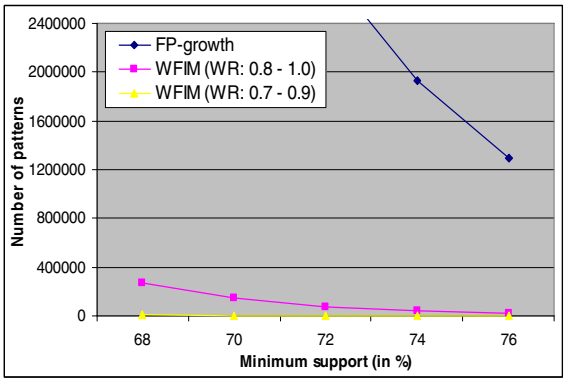

Figure 3. Number of patterns in WFIM (Connect dataset, WR: 0.8 – 1.0).

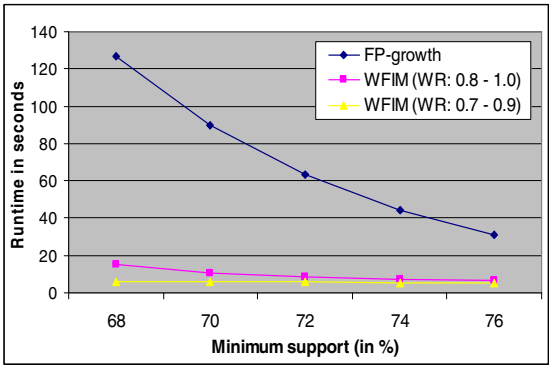

Figure 4. Runtime in WFIM (Connect dataset, WR: 0.8 – 1.0).

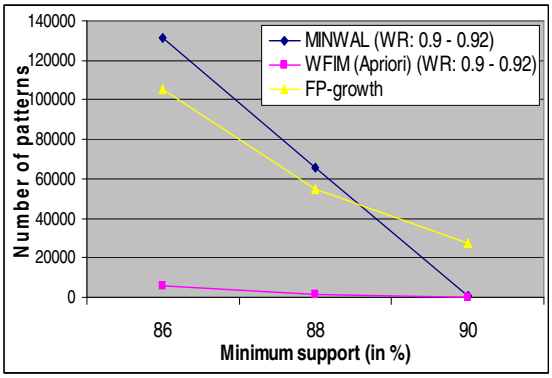

Figure 5. Number of patterns in WFIM (Connect dataset, WR: 0.9 – 0.92).

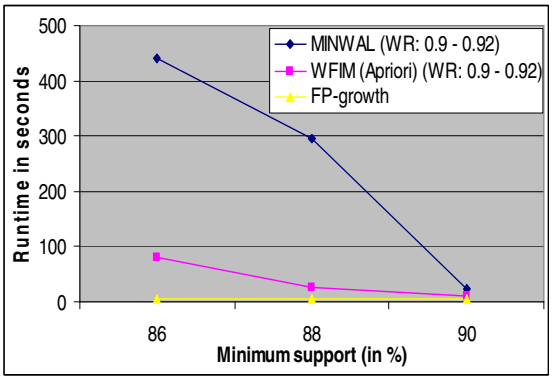

Figure 6. Runtime in WFIM (Connect dataset).

Our experiment shows that in most cases, WFIM outperforms MINWAL, WFIM (Apriori) and FP-growth algorithms. First, we evaluate the performance on the Connect dataset. Figure 3 and Figure 4 show WFIM generate smaller patterns and run faster than FP-growth. Specifically, the smaller patterns are generated as the weight range is increased. In Figure 5, WFIM (Apriori) generates smaller patterns than MINWAL and FP-growth because WFIM (Apriori) can reduce the number of patterns and runtime by adjusting a weight range and a minimum weight but MINWAL does not use weight range effectively. However, in Figure 6, FP-growth is the fastest among three algorithms since the FP-growth algorithm is based on pattern growth approach so it does not need all candidate generation and test. From Figure 3 to Figure 6, we can see that WFIM is the fastest among other algorithms. In addition, the number of patterns discovered by WFIM is several orders of magnitude smaller than the number of patterns found by MINWAL and FP-growth.

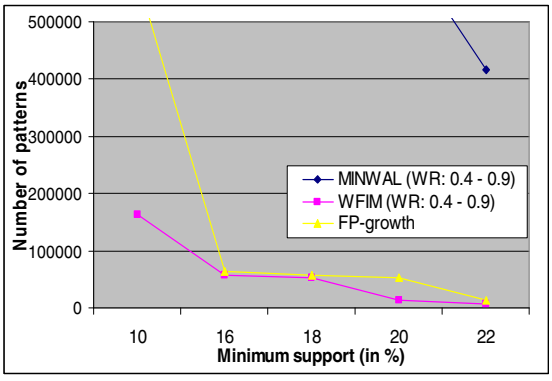

Figure 7. Number of patterns in WFIM (Mushroom dataset, WR: 0.4 – 0.9).

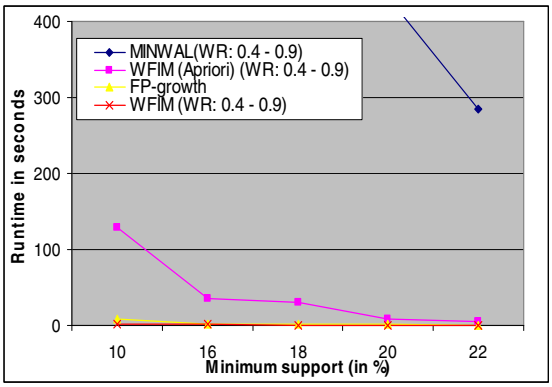

Figure 8. Runtime in WFIM (Mushroom dataset, WR: 0.4 – 0.9).

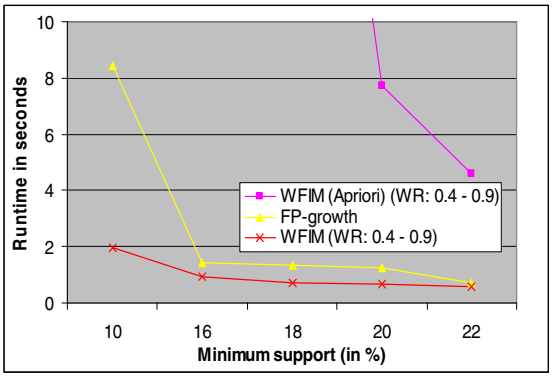

Figure 9. Runtime in WFIM (Mushroom dataset, WR: 0.4 – 0.9).

From Figure 7 to Figure 9 demonstrate the results of performance test using Mushroom dataset by setting weight range from 0.4 to 0.9. WFIM outperforms MINWAL, WFIM (Apriori), and FP-growth. When the support threshold is lowered, the performance difference becomes bigger. Not that in Figure 7, the number of patterns for WFIM is increased as the minimum support is decreased. However, the number of patterns in MINWAL and FP-growth is substantially increased. In Figure 8 and Figure 9, we can see that the runtime of pattern growth based algorithms, WFIM and FP-growth is

faster than Apriori based algorithms, MINWAL and WFIM (Apriori). The difference becomes bigger as a minimum support is lowered.

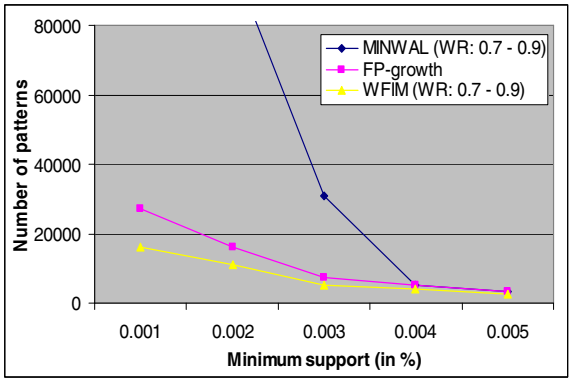

Figure 10. Number of patterns in WFIM (BMW-webview-1, WR: 0.7 – 0.9).

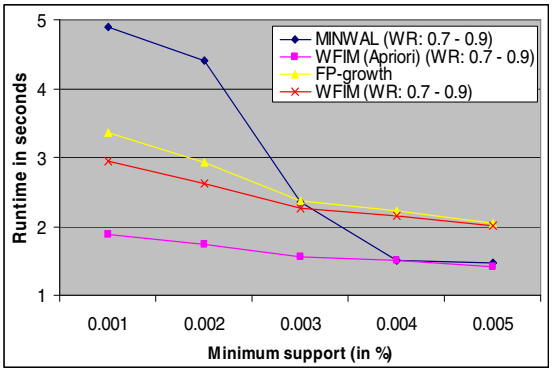

Figure 11. Runtime in WFIM (BMW-webview-1, WR: 0.7 – 0.9).

In Figure 10 and Figure 11, we report the evaluation results for BMW-webview-1dataset. We set up a weight range: 0.7 – 0.9 in Figure 10 and Figure 11. The main performance difference between WFIM and MINWAL is result from not only using a weight range but also applying pattern growth method. By decreasing support threshold, the number of patterns of every algorithm is increased in Figure 10. However, the

number of MINWAL is extremely increased. Note that in Figure 11, WFIM (Apriori) is faster than MINWAL and FP-growth and even faster than WFIM. Additionally, MINWAL is faster than WFIM with greater than a minimum support, 0.003 %. From [96], we already know that in very sparse dataset such as BMW-webview-1, Apriori based algorithms can be faster that pattern growth algorithms due to overhead of constructing FP-tree structure in pattern growth methods. However, the different run time between WFIM (Apriori) and WFIM is less than one second so it's negligible. Our above experiments showed that WFIM and WFIM (Apriori) can generate smaller but important weighted frequent itemset with various WRs.

Table 6. Effectiveness of a minimum weight in WFIM

Minimum Support of Connect dataset	Num of W.F.I WR: 0.5 –1.5 MW: 1.5	Num of W.F.I WR: 0.5 – 1.5 MW: 1.0	Num of W.F.I WR: 0.5 – 1.5 MW: 0.5	Num of F.C.I	Num of F.I
64179 (95%)	125	784	1471	812	2205
60801 (90%)	690	2346	5312	3486	27127
54046 (80%)	2769	2989	3044	15107	533975
47290 (70%)	3997	4089	4093	35875	4129839

The Table 6 lists the number of Weighted Frequent Itemsets (WFI) with various minimum weights, Frequent Itemset (FI) and Frequent Closed Itemsets (FCI). From Table 6, WFIM can generate smaller WFI by using different Minimum Weight (MW) thresholds. For example, in Table 6, the number of WFI at a minimum support: 90%, a WR: 0.5 – 1.5 and a min_weight: 0.5, is 5312. However, the number of WFI can be

reduced to 2346 with a min_weight: 1.0 and can be further reduced to a 690 with a min_weight: 0.5. The numbers of frequent closed itemsets and frequent itemsets are 3486 and 27127 respectively. In this way, the proper number of weighted frequent itemsets can be found by adjusting a minimum weight.

3.5.2.2 Scalability test

To test the scalability with the number of transactions, the T10I4DxK datasets were used. WFIM is compared with MINWAL, WFIM (Apriori) and FP-growth. In Figure 13 and Figure 14, both WFIM and FP-growth show linear scalability with the number of transactions from 100k to 1000k. However, WFIM is much more scalable than the others.

First, we tested scalability of WFIM with regard to the number of transactions from 100K to 1000K. We set different minimum support of 0.1% to 0.5%. and weight range as 0.2 to 0.8 on the T10I4Dx synthetic datasets.

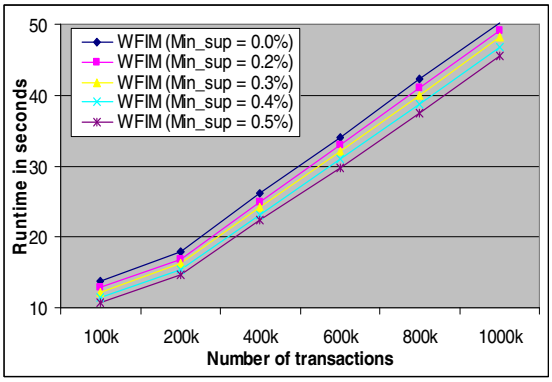

Figure 12. Scalability test in WFIM (T10I4DxK datasets).

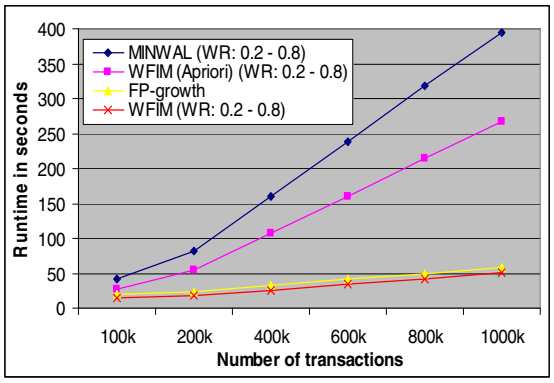

Figure 13. Scalability test in WFIM (T10I4DxK datasets, Min_sup = 0.1%).

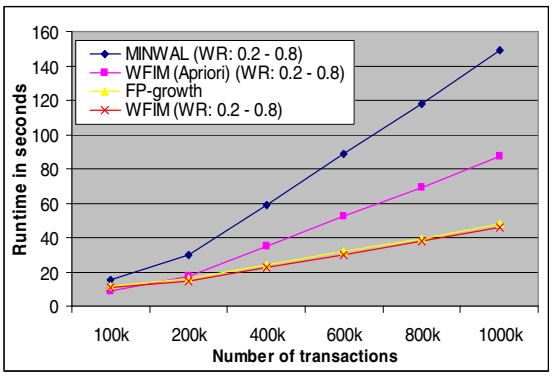

Figure 14. Scalability test in WFIM (T10I4DxK datasets, Min_sup = 0.5%).

In Figure 12, we can see that WFIM has good scalability in terms of number of transactions and becomes better as the minimum support is increased. Second, we compared WFIM with MINWAL, WFIM (Apriori) and FP-growth. We set minimum support as 0.1% in Figure 13 and 0.5% in Figure 14, and a weight range from 0.2 to 0.8 respectively. In Figure 13 and Figure 14, we can see that WFIM has much better scalability in terms of base size. The slope ratio of WFIM in both Figure 13 and Figure 14, is lower than other algorithms. In comparison with other algorithms, WFIM not only runs faster, it also has much better scalability in terms of base size.

3.6. Summary

In summary, WFIM is efficient and scalable in weighted frequent itemset mining. Previous weighted frequent itemset mining algorithms use Aprori based algorithms. However, WFIM uses divide and conquer approaches and maintains downward closure property. WFIM is faster than MINWAL, WFIM (Apriori) and FP-growth. Additionally, it becomes much faster and generates fewer but important frequent itemset for larger database even with a very low minimum support. Various FP-growth based frequent itemset mining algorithms with better performance have been developed such as [46], [49], [58], [77]. The performance of WFIM can be further improved by incorporating these algorithms.

4. WLPMINER: WEIGHTED FREQUENT PATTERN MINING WITH LENGTH-DECREASING SUPPORT CONSTRAINTS*

4.1 Overview of WLPMiner

Two main concerns exist for frequent pattern mining in the real world. First, each item has different importance so researchers have proposed weighted frequent pattern mining algorithms that reflect the importance of items. Second, patterns having only smaller items tend to be interesting if they have high support, while long patterns can still be interesting although their supports are relatively small. Weight and length decreasing support constraints are key factors, but no mining algorithms consider both constraints.

In this approach, we propose weighted frequent pattern mining with length decreasing support constraints [91]. The simple way to use a length decreasing support constraint is to set $\min_{l>0} f(l)$ as a minimum support and remove patterns which do not satisfy the length decreasing constraints. However, it takes a lot of time to generate frequent patterns. WLPMiner takes into account two features, a weight constraint and a length decreasing support constraint instead of only resetting the minimum support. This allows WLPMiner to generate more meaningful patterns. For pruning techniques, we propose 1) the notion of Weighted Smallest Valid Extension (WSVE) to apply to both

the length decreasing support constraints and weight constraints and 2) a weight range is used as a supplement to maintain the downward closure property.

The main contributions of this approach are: 1) incorporation of two key features for real datasets, a weight constraint and a length decreasing support constraint, 2) introduction of the concept of the weighted smallest valid extension property, 3) description of pruning techniques, transaction pruning and node pruning, using the weighted smallest valid extension property, 4) implementation of our algorithm, WLPMiner, and 5) execution of an extensive experimental study to compare the performance of our algorithm with BAMBOO [79] and WFIM.

4.2 Weighted frequent pattern mining with WSVE property

In this section, we suggest an efficient weighted frequent pattern mining algorithm with a length decreasing support constraint, called WLPMiner. Our approach is to push a weight constraint and a length decreasing support constraint into the pattern growth algorithm. We introduce the weighted smallest valid extension (WSVE) property to prune the search space. Based on weighted smallest valid extension property, we define transaction pruning, and node pruning. In addition, we show that the effect of combining a weight constraint and a length decreasing support constraint generates smaller but important patterns.

Definition 4.1 Weighted Smallest Valid Extension (WSVE) property

Given a pattern P and a conditional pattern P` such that (support (P) * weight (P)) < f (|P|), then f^{-1}(support (P) * weight (P)) = min (l| f (l) ≤ (support (P) * weight (P))) is

the minimum length that a super pattern of P must have before it can potentially satisfy the length deceasing support constraint.

Lemma 4.1 As the value, x, of multiplying the support with the weight of pattern P decreases, the value of the inverse function, $f^{-1}(x)$, of a length decreasing support constraint function increases. Meanwhile, as the value of x increases, the value of the inverse function of a length decreasing support constraint function, $f^{-1}(x)$ decreases.

The WSVE property allows us to prune the search space. It considers not only a support measure but also a weight measure. From the WSVE property, if the pattern P is a weighted infrequent pattern, the length of any superset of the pattern P should have at least $f^{-1}(\text{support (P)} * \text{weight (P)})$. If not, the superset is also a weighted infrequent pattern, so it can be pruned.

Lemma 4.2 Given a conditional pattern, X, and a pattern in the conditional database, Y, a weight of the conditional pattern, X is no less than a weight of a pattern, Y within a transaction, t in the conditional database.

In the weighted smallest valid extension property, weight (X) is used as a weight parameter of the inverse function of a length decreasing support constraint. WLPMiner uses an ascending weight ordered prefix tree and the tree is traversed using a bottom-up strategy. Therefore, the weight of a conditional pattern, X, is always greater than or equal to the weight of an item, Y of a transaction, t, within a conditional database.

Lemma 4.3 The following formula is always satisfied: $f^{-1}(\text{support (X)} * \text{weight (X)}) \leq f^{-1}(\text{support (X)} * \text{weight (Y)}) \leq f^{-1}(\text{support (X+Y)} * \text{weight (Y)})$.

As stated in lemma 4.2, weight (X) is always no less than weight (Y) and weight (X+Y) is always no less than weight (Y). From lemma 4.1, we know that f^{-1} (weight (X)) $\leq f^{-1}$(weight (X+Y)) $\leq f^{-1}$(weight (Y)). Finally, f^{-1}(support (X) * weight (X)) is less than or equal to f^{-1}(support (X) * weight (Y)) and f^{-1}(support (X) * weight (Y)) is less than or equal to f^{-1}(support (X+Y) * weight (Y)). In lemma 4.3, we see that f^{-1} (support (X) * weight (X)) is the minimum length for a superset (X+Y) of a conditional pattern (X). Weighted frequent patterns with a length decreasing support constraint should satisfy the WSVE property, although other patterns may also satisfy the WSVE property.

4.3 Pruning by WSVE property

Using the weighted smallest valid extension property, we suggest two pruning techniques, transaction pruning and node pruning. Transaction pruning is applied before constructing FP-tree, while node pruning is used after building FP-tree.

4.3.1 Transaction pruning by WSVE property

Given a length decreasing support constraint f(l), and a conditional database D` with regard to a conditional pattern X, a pattern $Y \in D`$ can be pruned from D` if ((support (X) * weight (X)) < f (|X| + |Y|)).

The transaction pruning method is used to remove candidate transactions of a conditional database. It uses the weighted smallest valid extension property. Separate local FP-trees are built for all patterns that contain the conditional pattern. From the WSVE property, any superset (X+Y) of a conditional pattern (X) must have a length of

at least f^{-1}(support (X) * weight (X)). We can remove any patterns (Y) with a length of less than f^{-1}(support (X) * weight (X)) - |X|. This formula can be rewritten as: |X| + |Y| < f^{-1}(support (X) * weight (X)), which is the same as (support (X) * weight (X)) < f (|X| + |Y|).

4.3.2 Node pruning by WSVE property

Given a length decreasing support constraint f(l), a conditional pattern database D` with regard to a conditional pattern P, and the FP tree, T, built from D`, a node v in T can be pruned from T if ((h (v) + |P|) < f^{-1} (support(I (v)) * weight (I (v)))).

The node pruning method reduces nodes of a conditional local FP-tree. Assume that I(v) is the item stored in this node and h(v) is the height of the longest path from the root to a leaf node in which path, the node, v should be located. From the weighted smallest valid extension property, we can see that a node that contributes to a weighted frequent pattern, should adhere to the following formula: h(v) + |P| ≥ f^{-1} (support (I(v)) * weight (I(v))). Therefore, we can define node pruning by the weighted smallest valid extension in definition 4.1. This formula can be rewritten as: f (h(v) + |P|) ≤ (support (I(v)) * weight (I(v))). We can remove a node if ((h (v) + |P|) < f^{-1} (support (I(v)) * weight (I(v)))). Assume that the transactions of the conditional database are sorted in decreasing transaction length and traverse each transaction in that order. Let t be a transaction and l (t) be its length. For practical considerations, we can use this formula: ((l (t) + |P|) < f^{-1} (support (I (v)) * weight (I (v)))) instead of ((h (v) + |P|) < f^{-1} (support (I (v)) * weight (I (v)))).

4.4 Weighted frequent patterns with length decreasing support constraints

A pattern X is a weighted frequent pattern with a length decreasing constraint if all of the following pruning conditions are satisfied. If pattern X does not satisfy anyone of them, the pattern is called a weighted infrequent pattern with a length decreasing constraint.

Pruning condition 4.1: (support ≥ f (maxLength) || weight ≥ min_weight)

Pruning condition 4.2: (support * MaxW (MinW) ≥ f (maxLength)

Pruning condition 4.3: Transaction pruning by the WSVE property

Pruning condition 4.4: Node pruning by the WSVE property

In a transaction database, the value of multiplying the support of a pattern with a maximum weight (MaxW) among items in the transaction database is less than f (maxLength). In conditional databases, the value of multiplying the support of a pattern with a minimum weight (MinW) of a conditional pattern in the conditional database is less than f (maxLength), a length decreasing minimum support.

In WLPMiner, an ascending weight order method and a bottom-up traversal strategy are used in mining weighted frequent patterns. WLPMiner defines weighted Smallest Valid Extension property and prunes transactions and nodes by the WSVE property. The performance of pruning conditions 4.1 and 4.2 may not be good since the minimum support for the longest pattern of the length decreasing support constraint must be used in order to keep downward closure property. However, performance can be improved by using these pruning conditions with the weighted closure property and the

weight range. The weighted smallest valid extension and the downward closure property are both used to prune the search space.

4.5 WLPMiner algorithm

The WLPMiner algorithm uses a weight range and a minimum weight. Items are given different weights within the weight range. We now the weighted frequent pattern mining process and present the mining algorithm.

WLPMiner algorithm: Mining weighted frequent patterns with a length decreasing support constraint

---

Input: (1) A transaction database: TDB

(2) f(x): a length decreasing support constraint function

(3) Weights of the items within weight range : w_i

(4) A minimum weight threshold : min_weight

Output: (1) WFP: the complete set of Weighted Frequent Patterns that satisfy the length decreasing support constraint.

Begin

1. Let WFP be the set of weighted frequent patterns that satisfy the length decreasing support constraint. Initialize WFP ← 0;

2. Scan TDB once to find the global weighted frequent items satisfying the following definition: A pattern X is a Weighted Frequent Pattern (WFP) if the following pruning conditions 2.1 and 2.2 are not satisfied.

Condition 2.1: (support < f(maxLength) && weight < min_weight)

The support of a pattern is less than a minimum support and the weight of a pattern is less than a minimum weight constraint.

Condition 2.2: (support * MaxW < f(maxLength))

In a transaction database, the value of multiplying the support of a pattern with a maximum weight (MaxW) of each item in the transaction database is less than a minimum support.

3. Sort items of WFP in weight ascending order. The sorted weighted frequent item list forms the weighted frequent list.

4. Scan the TDB again and build a global FP-tree using weight_order.

5. Call WLPMiner (FP-tree, 0, WFP)

Procedure WLPMiner (Tree, α, WFP)

1: for each a_i in the header of Tree do

2: set $\beta = \alpha \cup a_i$;

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) < f(maxLength) && weight (β b_j) < min_weight) delete b_j from I_β;

7: if (sub (β b_j) * MinW < f(maxLength)) delete b_j from I_β;

8: end for

9: $CDB_\beta \leftarrow$ transaction_pruning_by_WSVE (β, CDB_β);

10: $Tree_\beta \leftarrow$ FP_Tree_Construction (I_β, CDB_β)

11: $Tree_\beta \leftarrow$ node_pruning_by_WSVE (β, $Tree_\beta$);

12: if $Tree_\beta \neq 0$ then

13: call WLPMiner ($Tree_\beta$, β, WFP)

14: end if

15: end for

In the WLPMiner algorithm, TDB is scanned once, weighted frequent items satisfying condition 2.1 and 2.2 are found and these items are sorted in weight ascending order. The WLPMiner algorithm then calls the recursive procedure WLPMiner (Tree, α, WFP). Lines 6 and 7 generate weighted frequent patterns with a length decreasing constraint. Line 9 conducts transaction pruning by the WSVE property. If a pattern in a conditional database satisfies the transaction pruning, it is inserted into a local FP-tree. After a local FP-tree is constructed in line 10, node pruning by the WSVE property is carried out in line 11. WLPMiner algorithm adopts the bottom-up divide and conquer paradigm to grow the current the prefix. If the local FP-tree is not empty, the procedure WLPMiner ($Tree_\beta$, β, WFP) is called recursively in line 13.

4.6 Performance evaluation

In this section, we present our performance study over various datasets. WLPMiner is the first weighted frequent pattern mining algorithm that considers both weight constraints and length decreasing support constraints which are characteristics of real datasets. We report our experimental results on the performance of WLPMiner in comparison with recently developed algorithms such as BAMBOO [79] and WFIM. Our

results show that WLPMiner not only generates more concise and important result sets, but also has much better performance than recently developed mining algorithms through incorporating a length decreasing support constraint into weighted frequent pattern mining. Moreover, WLPminer has good scalability of the number of transactions. In our experiments, we compared WLPMiner with BAMBOO [79] which is a frequent pattern mining algorithm with a length decreasing support constraint. We also compared WLPMiner with WFIM that is a weighted frequent pattern mining algorithm developed recently.

4.6.1 Test environment and datasets

Table 5 shows the characteristics of the datasets used for performance evaluation. The three real datasets used are Connect, Mushroom and Pumsb respectively. The Connect dataset is very dense and includes game state information. The Mushroom dataset is a little dense and contains characteristics of various species of mushrooms. The Pumsb dataset is census dataset. We use T10I4D100k which is very sparse and contains 100,000 transactions. The synthetic datasets T10I4Dx contain from 100k to 1000k transactions. WLPMiner was written in C++. Experiments were performed on a sparcv9 processor operating at 1062 MHz, with 2048MB of memory. All experiments were performed on a Unix machine. In our experiments, a random generation function was used to generate weights for each item. When running WLPMiner, the minimum support was determined as the cut off value for the maximum pattern length under the corresponding length decreasing support constraint.

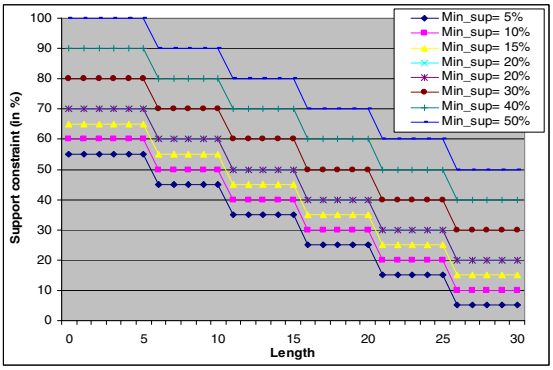

Figure 15. Length decreasing support constraints in WLPMiner (Connect dataset).

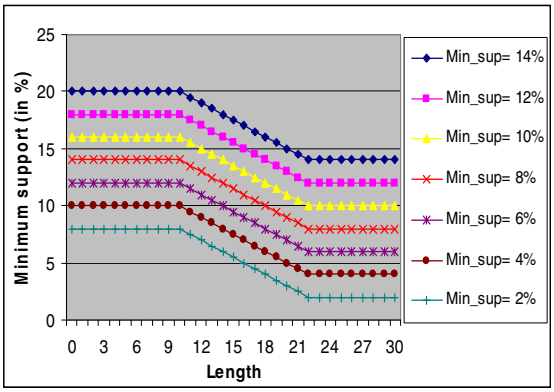

Figure 16. Length decreasing support constraints in WLPMiner (Mushroom dataset).

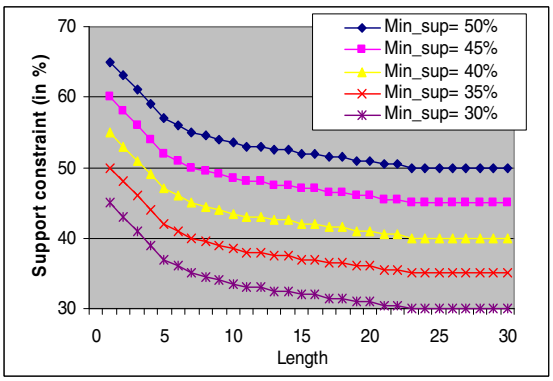

Figure 17. Length decreasing support constraints in WLPMiner (Pumsb dataset).

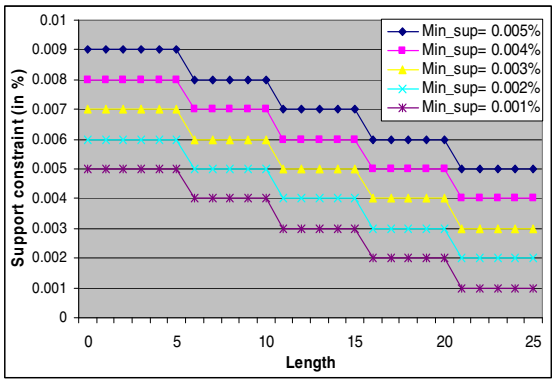

Figure 18. Length decreasing support constraints in WLPMiner (T10I4DnK dataset).

From Figure 15 to Figure 18 show various length decreasing support constraints used in the performance evaluation for different datasets. LPMiner and BAMBOO are recently developed mining algorithms using length decreasing support constraints. These length decreasing support constraints are similar to those used in BAMBOO. When running WLPMiner, the minimum support was determined as the cut off value for the maximum pattern length under the corresponding length decreasing support constraint.

4.6.2 Experimental results

In our experiments, we compared WLPMiner with BAMBOO which is a frequent pattern mining algorithm with a length decreasing support constraint. Bamboo outperforms LPMiner in terms of runtime and the number of frequent patterns. Therefore, we compared WLPMiner with BAMBOO. Additionally, we also compared WLPMiner with WFIM that is a weighted frequent pattern mining algorithm developed recently.

4.6.2.1 Comparison of WLPMiner with WFIM and BAMBOO

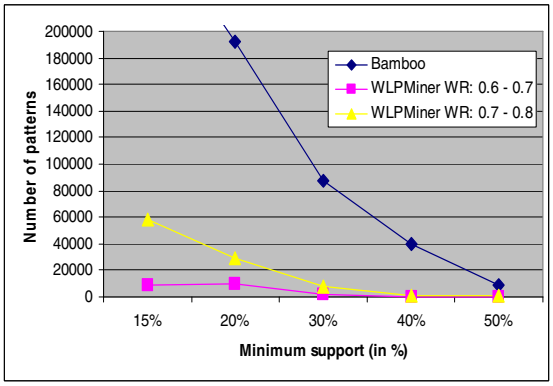

Figure 19. Number of patterns in WLPMiner (Connect dataset).

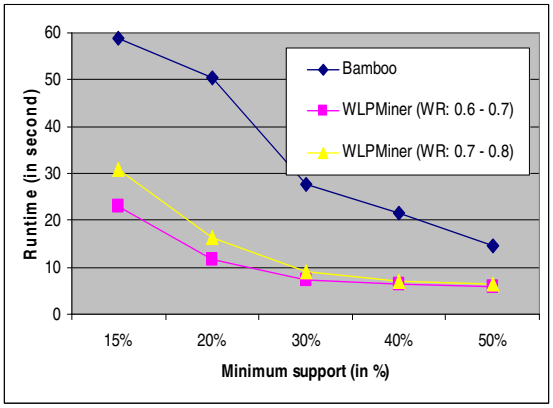

Figure 20. Runtime in WLPMiner (Connect dataset).

In this test, we focused on the efficiency of incorporation of weight constraint and length decreasing support constraints. First, we evaluated the performance on the Connect dataset. We use two weight ranges, 0.6 – 0.7 and 0.7 – 0.8 respectively. Figure 19 and Figure 20 show results of performance test for the Connect dataset. Figure 19 compares the number of weighted frequent patterns of WLPMiner with those of WFIM

and BAMBOO. Figure 20 shows the runtime of the algorithms under the same weight range. In Figure 19, we can see WLPMiner generates smaller frequent patterns than that of BAMBOO. From Figure 20, we see that WLPMiner is much faster than BAMBOO. Although WLPMiner and BAMBOO algorithms use a length decreasing support constraint, WLPMiner outperforms BAMBOO because WLPMiner uses the Weighted Smallest Valid Extension property and incorporates weight constraints into length decreasing support constraints.

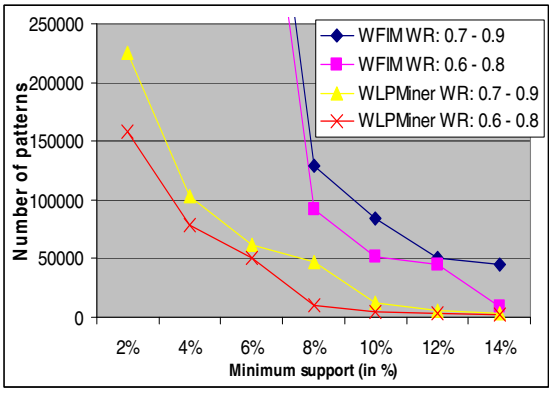

Figure 21. Number of patterns in WLPMiner (Mushroom dataset).

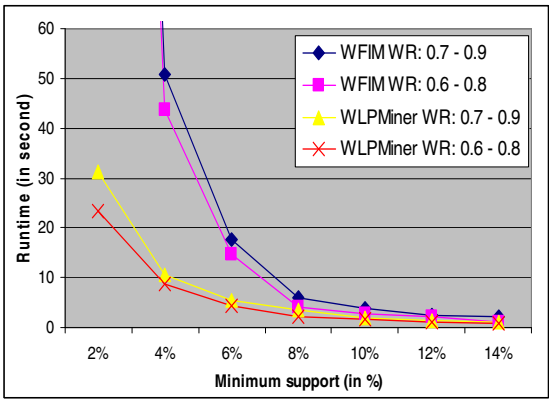

Figure 22. Runtime in WLPMiner (Mushroom dataset).

Figure 21 and Figure 22 demonstrate the results of performance test using the Mushroom dataset with weight ranges, 0.6 – 0.8 and 0.7 – 0.9. WLPMiner outperforms WFIM. When a weight range is lowered, the performance becomes better. Although WLPMiner and WFIM use same weight range, WLPMiner have better performance since WLPMiner use pruning techniques based on Weighted Smallest Valid Extension property.

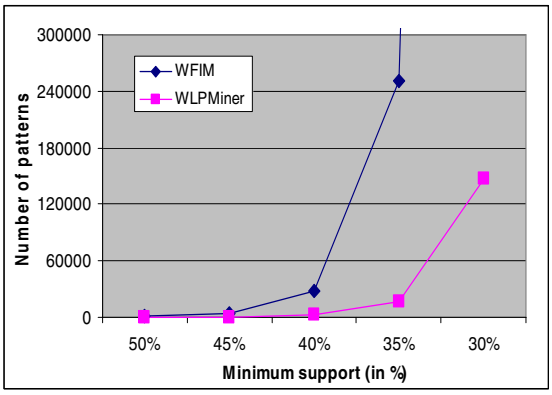

Figure 23. Number of patterns in WLPMiner (Pumsb dataset).

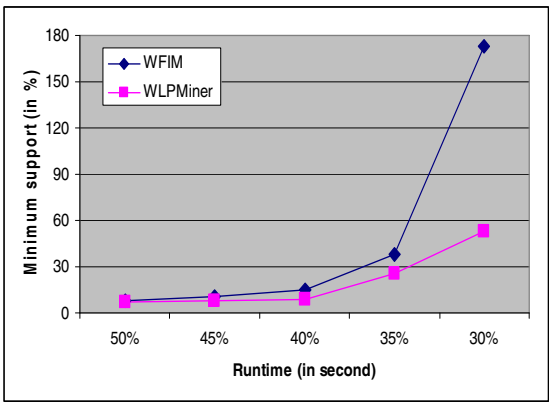

Figure 24. Runtime in WLPMiner (Pumsb dataset).

From Figure 23 to Figure 24, we report the evaluation results for the Pumsb dataset. We set up a weight range: from 0.4 – 0.8. In most cases, WLPMiner is faster than WFIM and generates fewer patterns. Users may increase a minimum support to find fewer patterns with higher frequency. However, the number of patterns at minimum support 35% in WFIM is dramatically increased. Meanwhile, the number of patterns in WLPMiner is controllable.

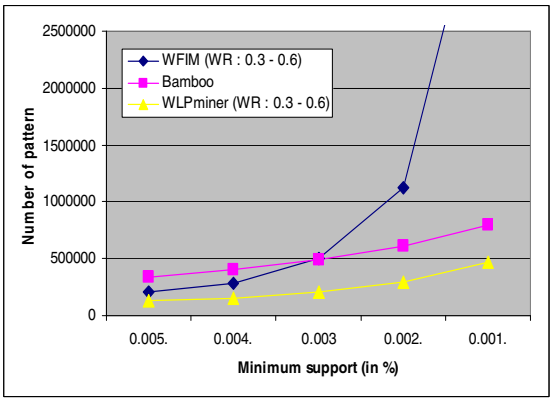

Figure 25. Number of patterns in WLPMiner (T10I4D100K dataset).

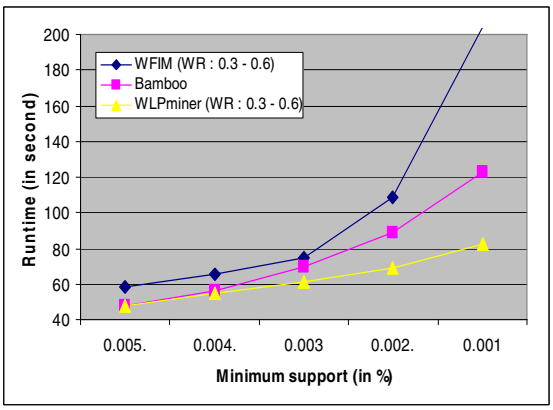

Figure 26. Runtime in WLPMiner (T10I4D100K dataset).

In Figure 25 and Figure 26, we report the evaluation results for T10I4D100K dataset. We set up a weight range: 0.3 – 0.6 in Figure 25 and Figure 26. The main performance difference between WLPMiner and WFIM is result from using length decreasing support constraints. Additionally, WLPMiner outperforms BAMBOO since WLPMIner applies weight constraint. By decreasing a support threshold, the number of patterns of every algorithm is increased in Figure 25. However, the number of WFIM is extremely increased. From Figure 25 to Figure 26, we can see that WLPMiner generates fewer but important patterns. In addition, it can be several orders of magnitude faster than WFIM since the Weighted Smallest Valid Extension property for the length decreasing support constraint is effective and efficient in pruning the result set in the datasets.

4.6.2.2 Quality of patterns in WLPMiner

We showed that WLPMiner not only generates more concise and important result patterns, but also has much better performance than recently developed algorithms, BAMBOO and WFIM. In this section, we show the quality of weighted frequent patterns with length decreasing support constraints. In most of test datasets, items are expressed as integer values so it is not easy to understand the meaning of items and discovered patterns. In this analysis, the T10I4D100K dataset is used to test the quality of weighted frequent pattern mining. A minimum support is set to 0.002% and a weight range is set as 0.3 – 0.6. We analyzed the frequent patterns to show effectiveness of mining weighted frequent pattern mining with a length decreasing support constraints. We compared the patterns mined by WLPMiner with those of WFIM. First, BAMBOO generate following

patterns: <110, 282, 810, 956>: 14, <48, 55, 185, 637, 877, 989>: 19 and <185, 234, 631, 877, 958, 989>: 19. However, these patterns are pruned by WLPMiner because these patterns have low weights although the supports of these patterns are no less than length decreasing support constraints. Second, <529, 541, 829, 937>:12, <295, 529, 541, 829, 937>: 12 and <41, 208, 529, 590, 606>:12 are discovered by WFIM. Meanwhile, these patterns are all pruned by WLPMiner since WLPMiner consider length decreasing support constraints. Note that WLPMiner apply not only a weight constraint but also length decreasing support constraints so more important and reasonable patterns can be mined.

4.6.2.3 Scalability test

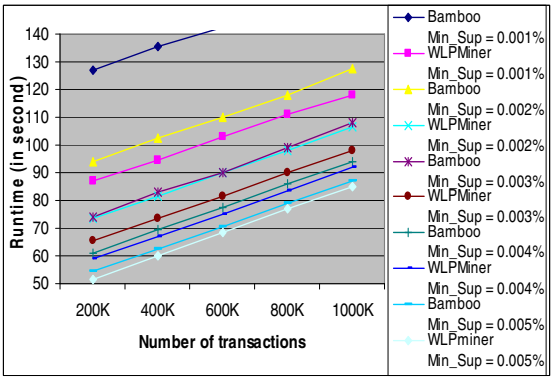

Figure 27. Scalability test in WLPMiner (T10I4Dx dataset, WR: 0.3 - 0.6).

To test the scalability with the number of transactions, T10I4DxK datasets are used. We set a weight range as 0.3 to 0.6 and changed the minimum support from 0.001% to 0.005%**.** In this scalability test, WLPMiner is compared with BAMBOO. BAMBOO shows linear scalability with the number of transactions from 200k to 1000k.

However, WLPMiner is much more scalable than BAMBOO. From Figure 27, the difference between the two algorithms becomes clear. We first tested the scalability in terms of base size from 200K tuples to 1000K tuples and different minimum support of 0.001% to 0.005%. From Figure 27, we can see that WLPMiner has much better scalability in terms of base size. The slope ratio for each different minimum support is almost similar.

4.7. Summary

We developed the WLPMiner algorithm that integrates a weight constraint with a length decreasing support constraint for mining frequent patterns. The key insights achieved in this approach are the high performance of the WSVE property and the use of a weight range in the weight constraint. We show that combining a weight constraint with a length decreasing support constraint improves performance in terms of the number of patterns and runtime. The extensive performance analysis shows that WLPMiner is efficient and scalable in weighted frequent pattern mining. In future work, the WSVE property will be used with different pruning techniques suggested in other algorithms using length decreasing support constraints.

5. WIP: WEIGHTED INTERESTING PATTERN MINING WITH A STRONG WEIGHT AND/OR SUPPORT AFFINITY

5.1 Overview of WIP

In this approach, we propose an efficient mining algorithm called WIP (Weighted Interesting Pattern mining) based on the pattern growth approach. We define the concept of a weighted hyperclique pattern that uses a new measure, called weight-confidence, to consider weight affinity and prevent the generation of patterns with substantially different weight levels. The weight confidence is used to generate patterns with similar levels of weights and the actual h-confidence serves to identify strong support affinity patterns. The Hyperclique Miner [86] which first used an h-confidence, adopted an Apriori algorithm. In addition, Hyperclique Miner used an upper bound on h-confidence. However, the actual h-confidence is more accurate than the upper bound of h-confidence. WIP uses the actual h-confidence instead of the upper bound of h-confidence so weak support affinity patterns can be pruned earlier. The actual h-confidence can be calculated without additional computation cost in WIP based on the FP-tree structure (section 5.5). An extensive performance analysis shows that weighted interesting patterns are extremely valuable patterns, since they have strong affinity in terms of a weight and/or support. Users can adjust the number of frequent patterns or find more valuable patterns by using weighted frequent patterns or weighted frequent patterns with weight affinity and/or support affinity, instead of only obtaining frequent patterns by changing the support threshold. Weighting applications exist in which items

have different importance and patterns with a similar level of support and/or weight are more meaningful. For example, strong support and/or weight affinity measures can be applied in DNA analysis and web log mining by giving more weights to specific DNA patterns or web log sequences.

The main contributions of this approach are: 1) introduction of the weight-confidence measure and weighted hyperclique patterns, 2) integration of two key features, a weight and a support, 3) improvement in performance by applying actual h-confidences without additional computation cost, 4) implementation of our algorithm, WIP, and 5) an extensive experimental study that compared the performance of our algorithm with WFIM and the Hyperclique Miner [86].

5.2 Analysis of previous algorithms

Previous weighted frequent pattern mining algorithms [12], [72], [82] are based on the Apriori algorithm which uses a candidate set generation and test mechanism. WFIM is the weighted frequent itemset mining algorithm to use a pattern growth algorithm. WFIM focused on the downward closure property while maintaining algorithm efficiency. Patterns generated by WFIM have weak support and/or weight affinity patterns. WFIM uses a weight range to adjust the number of patterns. However, WFIM does not provide ways to remove patterns that include items with different support and/or weight levels. It would be better if the weak affinity patterns could be pruned first, resulting in fewer patterns after mining.

To mine correlated patterns, interesting measures [11], [40], [53], [71], [73], [84] have been suggested. In Hyperclique Miner [86], a hyperclique pattern was defined that

generates patterns involving items with a similar level of support. The Hyperclique Miner has the following limitations. First, Hyperclique Miner has adopted an Apriori algorithm which uses candidate set generation and test approaches. It is very costly to generate and test all the candidates. Second, Hyperclique Miner uses an upper bound on h-confidence which is defined as $\text{Min}_{1 \le j \le m}$ {support ($\{i_j\}$)} / $\text{Max}_{1 \le k \le m}$ {support ($\{i_k\}$)} when a pattern $P = \{i_1, i_2, \ldots, i_m\}$ is given. Because Hyperclique Miner is based on the Apriori algorithm, the upper bound can reduce the computation cost. However, the upper bound on h-confidence is no more accurate than the original h-confidence. In Apriori based algorithms, the cost for computing support for patterns with size greater than or equal to two is higher than the cost for computing support for single items. However, in pattern growth methods which use the FP-tree structure, the real support value of a pattern with any size (instead of the minimum support of items within patterns) can be calculated without additional computation cost. Third, our performance tests show that many cases exist in which the number of hyperclique patterns is not increased or decreased, even though the minimum support is changed. Additionally, the minimum h-confidence must be set very high. Otherwise, the number of hyperclique patterns becomes very large or is unchanged even if the minimum h-confidence is increased. The h-confidence can be an effective measure, but the effect is not always present in real datasets.

5.3 Weighted hyperclique patterns

In this section, we define the weight confidence measure, explain the concept of weighted hyperclique patterns, and show important properties.

Definition 5.1 Weight confidence (w-confidence)

Weight confidence of a pattern $P = \{i_1, i_2, \ldots, i_m\}$, denoted as wconf (P), is a measure that reflects the overall weight affinity among items within the pattern. It is the ratio of the minimum weight of items within the pattern to the maximum weight of items within the pattern. That is, this measure is defined as

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

Definition 5.2 Weighted Hyperclique Pattern (WHP)

A pattern is a weighted hyperclique pattern if the weight confidence of a pattern is greater than or equal to a minimum weight confidence. In other words, given a set of items $I = \{i_1, i_2, \ldots, i_m\}$, a pattern P is a hyperclique pattern if and only if $|P| > 0$ and wconf (P) □ min_wconf, where min_wconf is a minimum weight confidence.

Example 5.1: consider a pattern P = {A, B, C} and P` = {D, E, F}. Assume that a minimum weight confidence is 0.5, weight ({A}) = 0.2, weight ({B}) = 0.5, weight ({C}) = 0.8, weight ({D}) = 0.4, weight ({E}) = 0.5, and weight ({F}) = 0.6, where weight (X) is the weight value of a pattern X. Then, the average weight of pattern P and pattern P` are both 0.5, wconf (P) = 0.25 and wconf (P`) = 0.67. Therefore, pattern P is not a weighted hyperclique pattern but pattern P` is a weighted hyperclique pattern.

Property 5.1 Anti-monotone property of w-confidence

The anti-monotone property of w-confidence is similar to that of the support measure used in frequent pattern mining. If the w-confidence of a pattern P is no less

than a minimum w-confidence, every subset of pattern P is also no less than the minimum w- confidence.

Lemma 5.1 The w-confidence measure has the anti-monotonic property.

From definition 5.1, we can see that Max $_{1 \le k \le m}$ {weight ({i_k})} of the pattern P is always greater than or equal to that of a subset of pattern P and Min$_{1 \le j \le m}$ {weight ({i_j})} of the pattern P is always less than or equal to that of subset of pattern P. Therefore, we know that

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

That is, if the w-confidence of the pattern {i_1, i_2, …, i_m} is greater than or equal to a minimum w-confidence, so is every subset of size m - 1. Therefore, the w-confidence can be used to prune the exponential search space.

Property 5.2 Cross weight property

A pattern is called a cross weight pattern if the pattern contains items which have different levels of weights. Given a min_wconf (minimum weight confidence) as a threshold, if w-confidence has the cross weight property, for any cross weight pattern P with regard to a min_wconf, the value of the w-confidence is less than the min_wconf.

Lemma 5.2 The w-confidence has the cross weight property.

Given definition 5.1, assume that there is a cross weight pattern P = {i_1, i_2, ..., i_m} that contains at least two items x and y such that weight ({x}) / weight ({y}) < t. where 0 < t < 1.

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

$$\le \frac{\text{Min}_{1 \le j \le m}\{..., \text{weight}(\{x\}), ..., \text{weight}(\{y\}), ...\}}{\text{Max}_{1 \le k \le m}\{..., \text{weight}(\{x\}), ..., \text{weight}(\{y\}), ...\}}$$

$$\le \frac{\text{weight}(\{x\})}{\text{Max}_{1 \le k \le m}\{..., \text{weight}(\{x\}), ..., \text{weight}(\{y\}), ...\}}$$

$$\le \frac{\text{weight}(\{x\})}{\text{weight}(\{y\})} < t$$

Therefore, we know that the value of the w-confidence is less than the min_wconf for any cross weight pattern P with regard to a w-confidence threshold, t.

5.4 Effectiveness of w-confidence

Let us give a concrete example to show the effectiveness of w-confidence and weighted hyperclique patterns generated by using w-confidence. Given six items, their weights {"gold ring:5", "silver ring:2", "bronze ring:1.5", "gold necklace:4", "silver necklace:1.5", and "bronze necklace:1"} and a minimum w-confidence, 60%, we see that gold has a high weight, and silver and bronze have lower weights, even though the weight of silver is higher than that of bronze. Using a w-confidence measure, patterns that consist of items with similar levels of weight can be generated. For instance, "gold ring" and "gold necklace" are more important than other rings and necklaces so they have high weights. Meanwhile, "bronze ring" and "bronze necklace" are less important, so they have low weights. However, the item "gold ring" has lower support (frequency) than "silver ring" and "bronze ring" because "gold ring" is so expensive that customers do not usually buy it. We can think of the "necklace" items in a similar way. We know

that some customers who buy item “gold ring” are usually interested in other costly items such as “gold necklace”.

In contrast, other buyers who want to get a cheap “bronze ring” would like to find cheaper items such as “silver necklace” and “bronze necklace”. Patterns such as “gold ring, gold necklace” can not be mined by using a support measure and patterns such as “bronze ring, bronze necklace” or “silver ring, bronze necklace” can not be found by using a weight measure. Although the pattern “bronze ring, bronze necklace” and “bronze ring, silver necklace” have low weights, the w-confidences of the patterns are 67% (1/1.5) and 100% (1.5/1.5) respectively. Therefore, it is certain that they have more w-confidence than the pattern “gold ring, bronze necklace” (wconf = 20%) or “gold ring, silver necklace” (wconf = 30%). Using w-confidence, we can generate strong weight affinity patterns such as “gold ring, gold necklace” (wconf = 80%), “bronze ring, bronze necklace” (wconf = 67%) and “bronze ring, silver necklace” (wconf = 100%).

5.5 Mining weighted interesting patterns

In this section, we define weighted interesting pattern mining and show the mining process.

Definition 5.3 Weight Range (WR)

The weight of each item is assigned to reflect the importance of each item in the transaction database. A weight is given to an item with a weight range, $W_{min} \leq W \leq W_{max}$.

Definition 5.4 Weighted Interesting Pattern (WIP)

A pattern is a weighted interesting pattern if the following pruning conditions are satisfied. W-confidence can also be used independently.

Pruning condition 5.1: (support * MaxW (MinW) ≥ min_sup)

In a transaction database, the value of multiplying the support of itemsets with a maximum weight (MaxW) among items in the transaction database is greater than or equal to a minimum support. In conditional databases, the value of multiplying the support of an itemset with a minimum weight (MinW) of a conditional pattern in the FP-trees is greater than or equal to a minimum support.

Pruning condition 5.2: (w-confidence ≥ min_wconf)

The pattern is a weighted hyperclique pattern if the w-confidence of a pattern is no less than a minimum w-confidence.

Lemma 5.3 The w-confidence measure can be applied irrespective of different weight ranges.

WIP uses the weight range to maintain the downward closure property. For example, the WR_k of a pattern K = {A, B, C} is from 1 to 3 and the $WR_{k`}$ of a pattern K` = {D, E, F} is from 0.1 to 0.3. Assume that weight ({A}) = 1, weight ({B}) = 2, weight ({C}) = 3, weight ({D}) = 0.1, weight ({E}) = 0.2, and weight ({F}) = 0.3, where weight is the weight value of a pattern. Then, wconf (K) = 0.33 and wconf (K`) = 0.33. Using $WR_{k`}$ rather than WR_k generates fewer patterns by pruning condition 5.1. However, the weight confidences (0.33) of patterns K and K` are the same in spite of different weight ranges. We know that w-confidence is defined as the ratio of the minimum weight of items within the pattern to the maximum weight of items within the pattern.

Therefore, if ratios of the minimum weight to the maximum weight of different weight ranges are the same, the effect is the same. In other words, the w-confidence of a

pattern is only decided by a level of weight affinity between items of a pattern, not by a weight range.

Pruning condition 5.3: (actual h-confidence ≥ min_hconf)

The pattern is a hyperclique pattern if the h-confidence of a pattern is no less than a minimum h-confidence. Given a pattern $P = \{i_1, i_2, ..., i_m\}$, h-confidence of the pattern P is defined as support $(\{i_1, i_2, \ldots, i_m\})$ / $Max_{1 \le k \le m}$ {support $(\{i_k\})$}. The h-confidence for a pattern P has the following upper bound: upper (hconf (P)) = $Min_{1 \le j \le m}$ {support $(\{i_j\})$} / $Max_{1 \le k \le m}$ {support $(\{i_k\})$}. WIP uses actual h-confidence instead of the upper bound of h-confidence.

Lemma 5.4 The actual h-confidence can be used in the pattern growth method instead of the upper bound of h-confidence without additional computation cost.

Pattern growth approaches compute local frequent items of a prefix by scanning a projected database and making local FP-trees for the next step. In local FP-trees, the support of a local frequent item means the support of a pattern that includes a conditional pattern and a local frequent item.

Therefore, we can obtain the supports of patterns without additional computation cost by using the local FP-tree structure. In section 5.5.1, we show examples in which actual h-confidence is more accurate then upper bound of h-confidence.

Table 7. A transaction database TDB as a running example in WIP

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

5.5.1 FP-tree structure and bottom up divide and conquer

We use the transaction database TDB in Table 7. Assume that min_sup is 2, min_wconf is 0.6, min_hconf is 0.8 and use 0.2 □ WR □ 0.6 as the weight range in which the weight list is <a:0.6, b:0.5, c:0.3, d:0.3, f:0.2, g:0.5, h:0.4>. After the first scan of the transaction database, we know that the frequent list is <a:4, b:4, c:4, d:5, f:3, g:3. h:2> and the MaxW is 0.6.

When an item is inserted in the FP-tree, weighted infrequent items are removed and the rest are sorted by weight ascending order. From pruning by weighted support constraint, items “f”, “g” and “h” are pruned because the values of multiplying the supports of items “f”, “g” and “h” with a MaxW (0.6) is less than a minimum support (2).

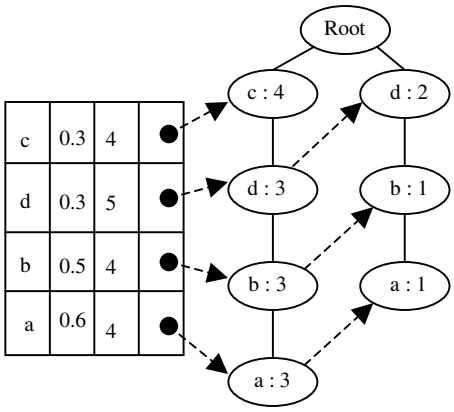

Figure 28. The global FP tree in WIP.

Figure 28 presents the global FP-tree and the related header table. After the global FP-tree is generated from the transaction database, WIP mines weighted interesting patterns from the FP-tree. The weighted interesting patterns are generated by adding items one by one. WIP adapts the divide and conquer approach for the mining process. WIP divides mining the FP-tree into mining smaller FP trees. From the global FP-tree, WIP 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”. For node “a”, we generate a conditional database by starting from a’s head and following a’s node link. The conditional database for prefix “a:3” contains two transactions: <bdc:3> and <bd:1>. In WIP, item “c:3” is pruned by the weighted support constraint of condition 5.1 in definition 5.4 because the value (1.8) of multiplying item c’s support (3) with a MinW (0.6) of the conditional pattern is less than the minimum support (2). In addition, a local item “d:4” is pruned by weight confidence. The candidate pattern, from a local item

“d:4” and a conditional pattern “a” is “ad:4” and the weight confidence (0.5) of the candidate pattern “ad:4” is less than the minimum weight confidence (0.6). Note that there is only one item “a” in the conditional pattern, so the MinW of the conditional pattern is 0.6. After that, Figure 29 (a) shows a FP- tree with prefix “a”. For node “b”, WIP derives a conditional pattern (b:3) and two paths in the FP-tree: <dc:3>, and <d:1>. A local item “c:3” is pruned by the weighted support constraint of condition 5.1 in definition 5.4 because the value (1.8) of multiplying the support (3) of an item “c” with MinW (0.6) of a conditional pattern “b” is less than the minimum support (2). After pruning weighted uninteresting patterns in the conditional database, the projected FP-tree for the prefix “b:3” is constructed. Figure 29 (b) shows a conditional FP-tree for prefix b:3. For node “d”, the conditional database for prefix “d:3” contains a transaction: <c:3>. Note that no item is pruned in this step when the upper bound of h-confidence is used in Hyperclique Miner [86], [87].

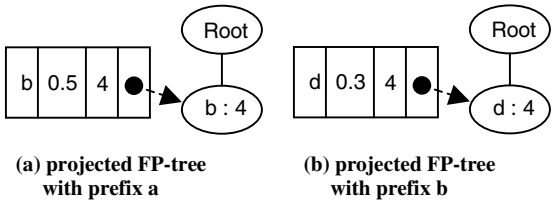

Figure 29. Conditional FP trees in WIP.

However, WIP uses actual h-confidence instead of the upper bound of h-confidence to prune weighted uninteresting patterns, so item “c:3” is pruned by actual h-confidence of condition 5.3 in definition 5.4 because the upper bound of h-confidence of

the pattern ”bc” is 1 (shown in Table 8) but the real h-confidence of the pattern is 0.75. The h-confidence (0.75) of the candidate pattern, “bc:4”, is less than the minimum h-confidence (0.8). In this way, we can build local projected FP-trees from the global FP-tree and mine weighted interesting patterns from them recursively. (The FP-trees for prefix d:3 and c:4 are empty and not shown in Figure 29). Note that the candidate pattern “bcd” is pruned by the anti-monotone property (downward closure property) of w-confidence. That is, if a pattern “bd” is not a weighted hyperclique pattern, the superset “bcd” of the pattern “bd” is also not a weighted hyperclique pattern and the candidate patterns are not generated. Table 8 shows examples of pruning candidate patterns by weighted support, w-confidence, upper bound of h-confidence and actual h-confidence.

Table 8. Pruning candidate patterns in WIP

Candidate patterns	Weighted support	Weight confidence	Upper bound of h-confidence	Actual h-confidence
ab : 4	(0.6 * 4)	0.83	1.0 (4/4)	1.0 (4/4)
ac : 3	Pruned (0.6 * 3)	Pruned (0.5)	1.0 (4/4)	Pruned 0.75 (3/4)
ad : 4	(0.6 * 4)	Pruned (0.5)	0.8 (4/5)	0.8 (4/5)
bd : 4	(0.6 * 4)	0.6	0.8 (4/5)	0.8 (4/5)
bc : 4	(0.6 * 4)	0.6	1.0 (4/4)	Pruned 0.75 (3/4)
dc : 3	Pruned (0.6 * 3)	1.0	0.8 (4/5)	Pruned 0.6 (3/5)

5.5.2 Comparison of w-confidence and h-confidence

In this section, we compare w-confidence with h-confidence. W-confidence is a weight measure that considers the weight affinity of items within a pattern. However, h-confidence is a support measure which is used to identify strong affinity patterns in terms of support. The previous use of a weight constraint [12], [82], [72] can generate many patterns containing different levels of weights or miss interesting low weight patterns. W-confidence is applied to find weighted frequent patterns with similar levels of weight. We defined w-confidence as $Min_{1\le j\le m}\{weight(\{i_j\})\} / Max_{1\le k\le m}\{weight(\{i_k\})\}$ instead of $weight(\{i_1, i_2, \ldots, i_m\}) / Max_{1\le k\le m}\{weight(\{i_k\})\}$ because the latter does not satisfy the anti-monotone property (downward closure property). Meanwhile, $Min_{1\le j\le m}\{weight(\{i_j\})\} / Max_{1\le k\le m}\{weight(\{i_k\})\}$ satisfies both the anti-monotone and cross weight properties. The computation of an upper bound on h-confidence is less accurate than the computation of the exact h-confidence value.

5.5.3 Comparison of w-confidence and weighted support constraint

W-confidence and weighted support constraints both use a weight feature. As shown in section 5.4, w-confidence can avoid generating patterns with different levels of weights. Although the weighted support constraint considers weight and support measures, it can generate weak weight affinity patterns. In other words, patterns satisfying weighted support constraint can not satisfy the w-confidence if they contain different levels of weights.

5.6 WIP algorithm

In WIP, an ascending weight order method and a bottom-up traversal strategy are used in mining weighted interesting patterns. WIP defines a w-confidence measure and generates weighted hyperclique patterns. Items are given different weights within the weight range. We now show the weighted interesting pattern mining process and present the mining algorithm.

WIP algorithm: Weighted Interesting Pattern mining with a weight confidence and/or an h-confidence

---

Input: (1) A transaction database: TDB,

(2) A minimum support: min_sup,

(3) Weights of the items within weight range: w_i,

(4) A minimum w-confidence: min_wconf

(5) A minimum h-confidence: min_hconf

Output: The complete set of weighted hyperclique patterns.

Begin

1. Let WIP be the set of weighted interesting patterns that satisfy the constraints.

Initialize WIP ← {};

2. Scan TDB once to find the global weighted frequent items satisfying the following definition: A pattern is a weighted frequent pattern if the following pruning condition is not satisfied.

Pruning condition: (support * MaxW < min_support)

In a transaction database, the value of multiplying the support of a pattern with a MaxW of each item in the transaction database is less than a minimum support.

3. Sort items of WIP in weight ascending order. The sorted weighted frequent item list forms the weighted frequent list.

4. Scan the TDB again and build a global FP-tree using weight_order.

5. Call WIP (FP-tree, { }, WIP)

Procedure WIP (Tree, α, WIP)

1: For each a_i in the header of Tree do

2: set $\beta = \alpha \cup a_i$;

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_support) delete b_j from I_β;

7: If (wconf (β b_j) < min_wconf) delete b_j from I_β;

8: If (hconf (β b_j) < min_hconf) delete b_j from I_β;

9: End for

10: $Tree_\beta \leftarrow$ FP_Tree_Construction (I_β, CDB_β)

11: If $Tree_\beta \neq 0$ then

12: Call WIP ($Tree_\beta$, β, WIP)

13: End if

14: End for

In the WIP algorithm, TDB is scanned once, weighted frequent items satisfying the pruning conditions are found and these items are sorted in weight ascending order. The WIP algorithm then calls the recursive procedure WIP (Tree, α, WFP). Line 6 generates weighted frequent patterns. Line 7 generates weighted hyperclique patterns with weight confidence. Line 8 generates hyperclique patterns with h-confidence. If a pattern in a conditional database satisfies the above constraints, it is inserted into a local FP-tree in line 10. The WIP algorithm adopts the bottom-up divide and conquer paradigm to grow the current prefix. If the local FP-tree is not empty, the procedure WIP ($Tree_\beta$, β, WFP) is called recursively in line 12. Since WIP is tail recursive, it is amenable to efficient iterative implementation. Note that the minimum thresholds such as a minimum support, a minimum w-confidence and a minimum h-confidence can be used selectively under user control.

5.7 Performance evaluation

In this section, we report our experimental results on the performance of WIP in comparison with two recently developed algorithms, WFIM and Hyperclique Miner [86]. WFIM is a weighted frequent pattern mining algorithm based on the pattern growth approach and Hyperclique Miner is a hyperclique pattern mining algorithm based on the Apriori algorithm. The main purpose of this experiment is not only to show the effectiveness of weight confidence in generating weighted hyperclique patterns but also

to demonstrate how effectively the weighted interesting patterns can be generated by incorporating a weight measure with a support measure.

First, we show how the number of weighted interesting patterns in real datasets and synthetic datasets can be adjusted through user feedback and the efficiency of the WIP algorithm. Second, we analyze quality of patterns discovered in WIP. Third, we show that WIP has good scalability against the number of transactions and attributes in the datasets.

5.7.1 Test environment and datasets

The three real datasets used are Connect, Pumsb and Mushroom respectively. The synthetic datasets T10I4Dx contain from 100k to 1000k transactions. We used synthetic datasets with different parameter settings for the scalability test.

Table 9. Parameter settings for the TaLbNc datasets in WIP

Data sets	|T|	|L|	N	Size (Mbytes)
T10.L1000.N10000	10	1000	10000	4.96
T20.L2000.N20000	20	2000	20000	10.73
T30.L3000.N30000	30	3000	30000	16.43
T40.L4000.N40000	40	4000	40000	22.13

A summary of the parameter settings is shown in Table 9, where |T| means the average size of a transaction, N means the number of items and |L| means the maximum number of potential frequent patterns. Each dataset in Table 9 contains 100000

transactions, with an average frequent pattern length equal to 4. We will call these synthetic datasets the TaLbNc datasets. WIP was written in C++. Experiments were performed on a sparcv9 processor operating at 1062 MHz, with 2048MB of memory. All experiments were performed on a Unix machine. In our experiments, a random generation function was used to generate weights for each item.

5.7.2 Experimental results

5.7.2.1 Comparison of WLPMiner with BAMBOO and WFIM

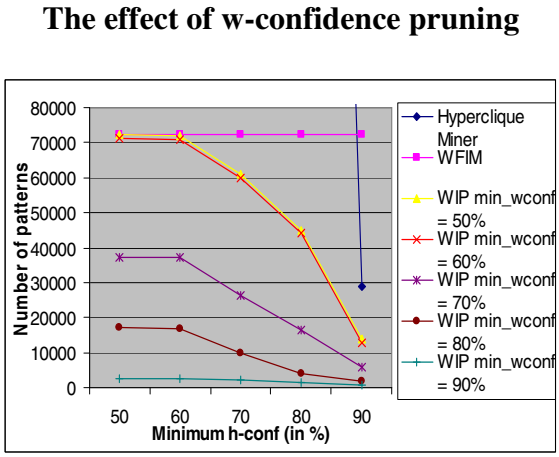

Figure 30. Number of patterns in WIP (Connect dataset, Min_sup = 10%).

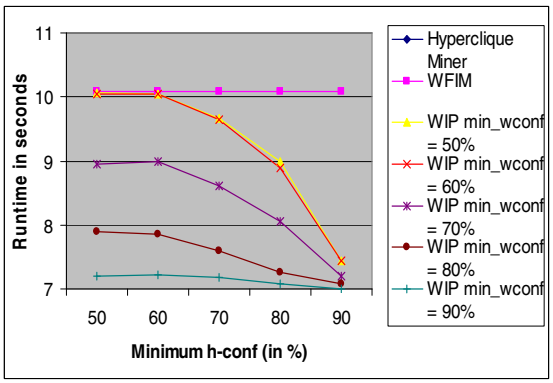

Figure 31. Runtime in WIP (Connect dataset, Min_sup = 10%).

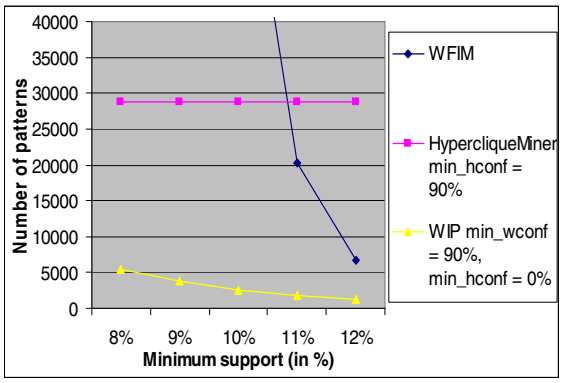

Figure 32. Number of patterns in WIP (Connect dataset, WR: 0.1 – 0.2).

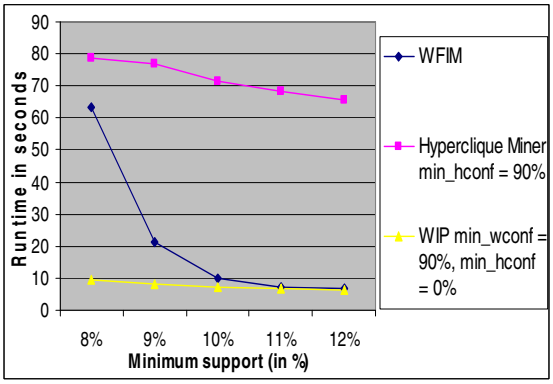

Figure 33. Runtime in WIP (Connect dataset, WR: 0.1 – 0.2).

In this test, we focused on the efficiency of w-confidence. First, we evaluated the performance on the Connect dataset. We set up a weight range from 0.1 to 0.2. Figure. 30 and Figure 32 show that WIP generates fewer patterns than WFIM and Hyperclique Miner. Specifically, fewer patterns are generated as the w-confidence is increased. In Figure 31 and Figure 33, we can see that WIP is faster than WFIM and Hyperclique Miner. In Figure. 32 and Figure 33, WIP did not use an h-confidence but only a w-confidence for the performance test of the w-confidence. The number of patterns discovered by WIP is several orders of magnitude fewer than the number of patterns found by Hyperclique Miner. Note that, in Figure 30, Hyperclique Miner generates a huge number of patterns with h-confidence of less than 90%. From Figure 32, we see that the number of patterns generated by Hyperclique Miner is unchanged although the minimum support is reduced. Hyperclique Miner uses two thresholds, a minimum h-confidence and a minimum support. However, from Figure 30, we see that the number of patterns increases quickly for the Hyperclique Miner as a minimum h-confidence is decreased. In Figure 31, we could not show the runtime of Hyperclique Miner because the runtime becomes so much larger as the minimum h-confidence is decreased. For example, the runtimes of Hyperclique Miner are 71 seconds with a minimum h-confidence of 90%, 146.27 seconds with a minimum h-confidence of 80% and 814.15 seconds with a minimum h-confidence of 70%. As a reverse case, the number of patterns in Hyperclique Miner is not reduced although a minimum support is increased in Figure 32. Meanwhile, runtime is increased when the minimum support becomes lower showing that the h-confidence is not always effective with different parameter settings in

real datasets. In Figure 30 and Figure 31, the number of patterns and runtime is not changed in WFIM because WFIM does not use an h-confidence measure.

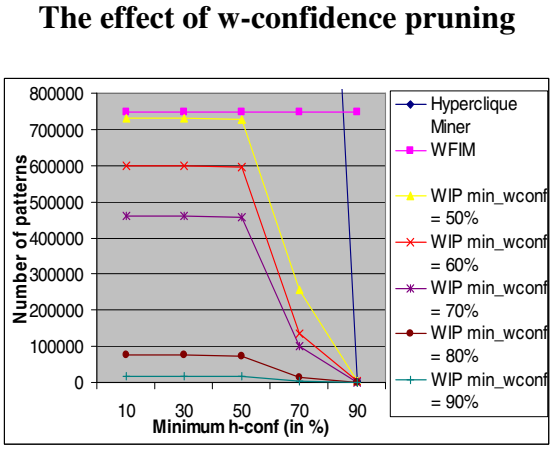

Figure 34. Number of patterns in WIP (Pumsb, Min_sup = 15%).

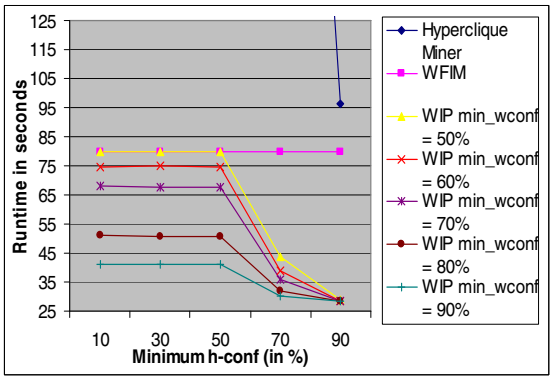

Figure 35. Runtime in WIP (Pumsb dataset, Min_sup = 15%).

The effect of combination of w-confidence and h-confidence pruning

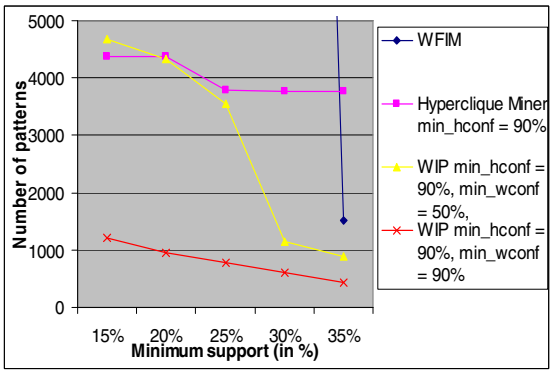

Figure 36. Number of patterns in WIP (Pumsb dataset, WR: 0.2 – 0.4).

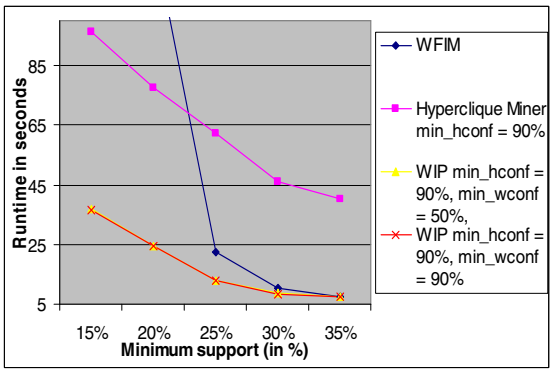

Figure 37. Runtime in WIP (Pumsb dataset, WR: 0.2 – 0.4).

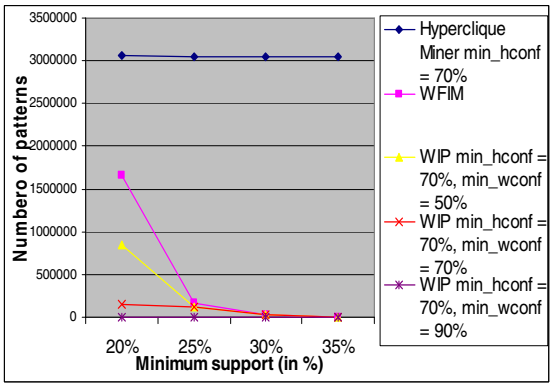

Figure 38. Number of patterns in WIP (Pumsb dataset, WR: 0.2 – 0.5).

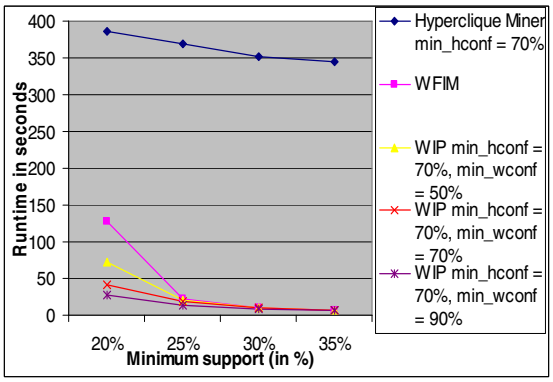

Figure 39. Runtime in WIP (Pumsb dataset, WR: 0.2 – 0.5).

From Figure 34 to Figure 39, we report the evaluation results for the Pumsb dataset. We set up a weight range: 0.2 – 0.4 from Figure 34 to Figure 37 and 0.2 – 0.5 in Figure 38 and Figure 39. The main performance difference between WIP and Hyperclique Miner in the real datasets results from not only using a weight range but also applying w-confidence. By increasing the w-confidence threshold, fewer patterns with a higher level of affinity in terms of weight can be generated. However, the performance gap between WIP and WFIM results from using w-confidence and h-confidence. Note that in Figure 34 and Figure 35, the number of patterns and runtime of WIP at different h-confidences from 0% to 50% do not change. From Figure 34 and Figure 35, we see that there are cases where the h-confidence measure is not effective with different parameter settings in this dataset. That is, the number of patterns and runtime of Hyperclique Miner is not reduced, although the minimum h-confidence is increased up to 50%. We fixed a minimum h-confidence at 90% in Figure 36 and Figure 37, and 70% in Figure 38 and Figure 39, and used several minimum weight confidences

as parameter values to test the effect of weighted interesting patterns as the minimum support threshold is changed. In most cases, WIP is the fastest among the three algorithms and generates fewer patterns.

We can also see that, with w-confidence pruning, the number of patterns can be adjusted. In other words, fewer interesting patterns with more weight affinity can be found. Users may increase a minimum support to find fewer patterns with higher frequency. However, the number of patterns at a minimum h-confidence 70% in Hyperclique Miner is unchanged, although the minimum support is decreased in Figure 38.

The effect of w-confidence pruning

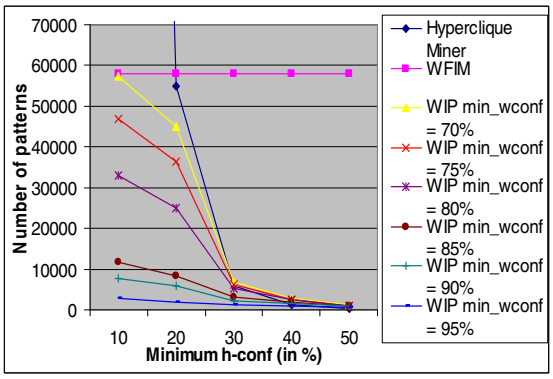

Figure 40. Number of patterns in WIP (Mushroom dataset, Min_sup = 10%).

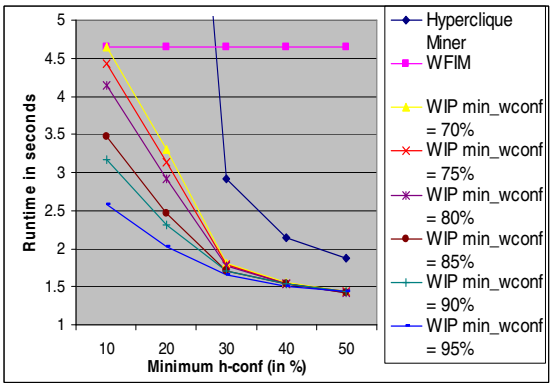

Figure 41. Runtime in WIP (Mushroom dataset, Min_sup = 10%).

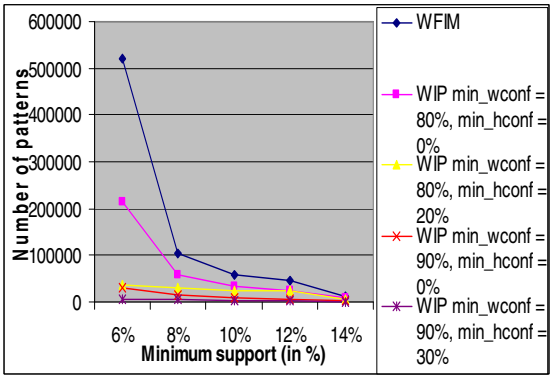

Figure 42. Number of patterns in WIP (Mushroom dataset, WR: 0.6 – 0.9).

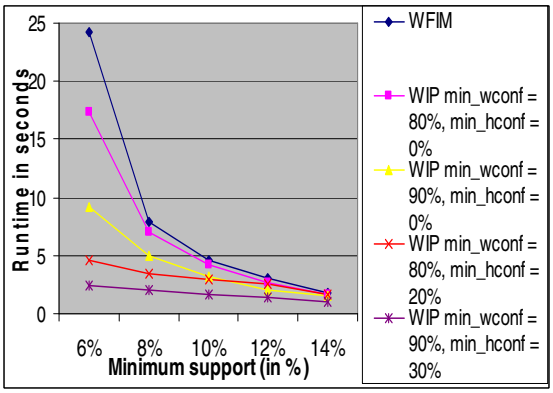

Figure 43. Runtime in WIP (Mushroom dataset, WR: 0.6 – 0.9).

Figure 40 to Figure 43 demonstrates the results of performance test using the Mushroom dataset with a weight range from 0.6 to 0.9. WIP outperforms WFIM and Hyperclique Miner. When the w-confidence threshold is lowered, the performance difference becomes larger. At higher w-confidences such as 90%, the performance of WIP becomes better. We can see that the number of patterns for WIP is decreased as the w-confidence and h-confidence are increased. Recall that WFIM can also adjust the number of patterns by resetting the weight range, although we fixed the weight range in these tests. However, WFIM generate weak affinity patterns even if a weight range is changed.

5.7.2.2 Quality of patterns in WIP

We have showed that the w-confidence can be used to prune patterns with weak weight affinity from real datasets. In most test datasets, items are expressed as integer values, so it is not easy to understand the meaning of items and discovered patterns. For example, in this analysis, the Pumsb dataset is used to test the quality of weighted frequent pattern mining. A minimum support is set to 20% and a weight range is set as 0.2 – 0.4. We analyzed the affinity patterns to show effectiveness of weighted frequent pattern mining with weight affinity. We compared the patterns mined by WIP with those of WFIM. For instance, patterns <260, 4426, 4436, 4438, 4940>: 40791 and <180, 260, 4404, 4436, 7092, 7112>: 37757 are mined by WFIM and patterns <170, 180, 4432, 4436, 4438, 7062>:44373 and <180, 4432, 4438, 4940, 7092>: 44217 are discovered by Hyperclique Miner with a min_hconf, 90%. However, these patterns are all pruned by w-confidence in WIP (min_wconf = 90%). In other words, these patterns contain items

with weak weight affinity although they have strong support affinity. In WFIM, although the minimum support is increased or the weight range is changed, weak affinity patterns are still found. The weak affinity patterns can be effectively pruned by w-confidence while mining strong weight affinity patterns.

5.7.2.3 Scalability test

The TaLbNc dataset is used to test scalability with the number of attributes and the T10I4DxK dataset is used to test scalability with the number of transactions. From the performance test, WIP scales much better than WFM and becomes better by weight and h-confidence thresholds. WIP and WFIM show linear scalability with the number of attributes from 100000 to 400000 and the number of transactions from 100k to 1000k. However, WIP is much more scalable than WFIM. From Figure 44 to Figure 47, the difference between WIP and WFIM becomes clear.

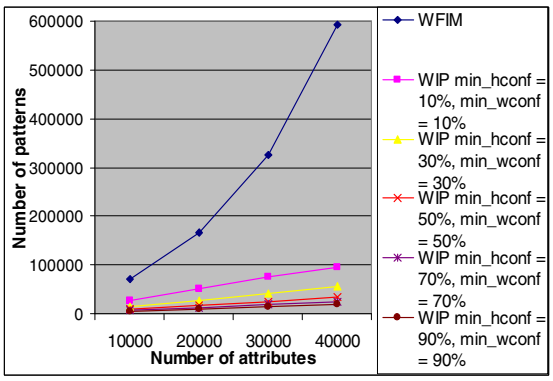

Figure 44. Scalability test (Number of patterns) in WIP (TaLbNc dataset).

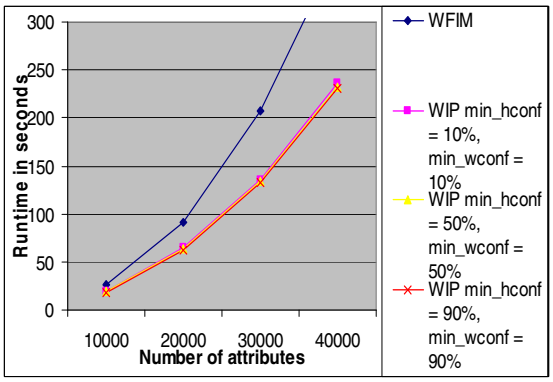

Figure 45. Scalability test (Runtime) in WIP (TaLbNc dataset).

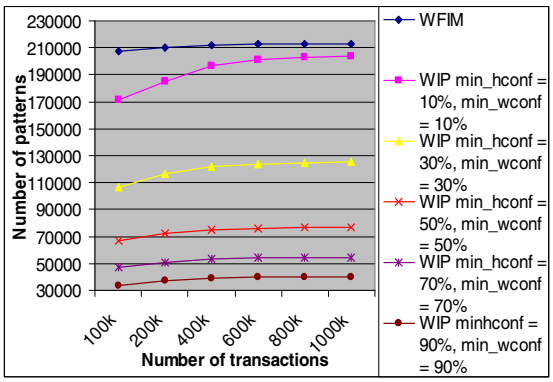

Figure 46. Scalability test (Number of patterns) in WIP (T10I4Dx dataset).

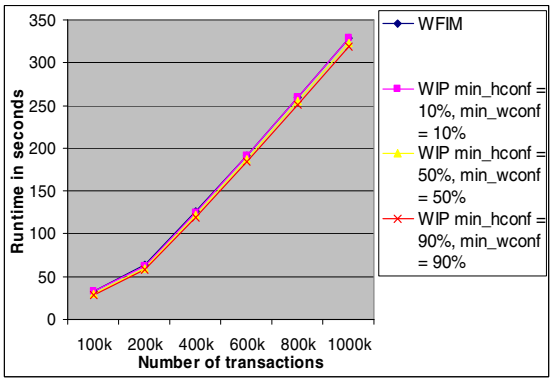

Figure 47. Scalability test (Runtime) in WIP (T10I4Dx dataset).

First, we tested the scalability in terms of the number of attributes from 10000 to 40000 using TaLbNc datasets which are used in [86]. In this test, we set a support threshold to 0.01% and a weight range from 0.3 to 0.9. From Figure 44 and Figure 45, we can see that WIP has much better scalability in terms of base size and WIP is much faster than WFIM. The slope ratio becomes higher as a w-confidence threshold and an h-confidence threshold are lower. With higher w-confidence and h-confidence, the number of patterns is more controllable and the runtime becomes faster.

Second, we performed a scalability test with regard to the number of transactions from 100K to 1000K on the T10I4Dx synthetic datasets. We set a minimum support as 0.003% and a weight range as 0.1 to 0.9. In Figure 46 and Figure 47, we can see that WIP has much better scalability in terms of number of transactions and becomes faster as the weight and h-confidence threshold is increased. Without w-confidence and h-confidence pruning, in WFIM, the number of patterns increases dramatically. The interesting result from Figure 47 is that the run time of WIP and WFIM is similar even if the runtime of WIP is a little faster than that of WFIM. We believe that the overhead of computing w-confidence of a pattern is negligible.

5.8 Summary

In summary, WIP is efficient and scalable in weighted interesting pattern mining. Although WFIM used a weight range and a minimum weight to reduce number of patterns, these patterns included items with different support levels and/or weight levels. For Hyperclique Miner, according to our performance test, there are many cases where the number of patterns and runtime did not reduce, although the minimum support or

minimum h-confidence was increased. As a result, WIP is faster than WFIM and Hypercliqe Miner and generates fewer but more interesting patterns for users.

6. WEIGHTED SEQUENTIAL PATTERN MINING

Sequential pattern mining is an important data mining area with broad applications. Researchers have suggested sequential pattern mining algorithms which mine the set of frequent subsequences satisfying a minimum support constraint in a sequence database. However, previous sequential mining algorithms treat sequential patterns uniformly while real sequential patterns have different importance. Another main problem in most of the sequence mining algorithms is that they still generate an exponentially large number of sequential patterns when a minimum support is lowered and they do not provide alternative ways to adjust the number of sequential patterns other than increasing the minimum support. In this chaper, we propose a Weighted Sequential pattern mining algorithms. Our main approach is to push the weight constraints into the sequential pattern growth approach while maintaining the downward closure property. We develop two approaches using weight constraints.

6.1 Problem definition and related work

6.1.1 Problem definition

Let $I = \{i_1, i_2 ... i_n\}$ be a unique set of items. A sequence S is an ordered list of itemsets, denoted as $\langle s_1, s_2, .., s_m \rangle$, where s_j is an itemset which is also called an element of the sequence, and $s_j \subseteq I$. That is, $S = \langle s_1, s_2, \ldots, s_m \rangle$ and s_j is $(x_1 x_2 \ldots x_k)$,where x_t is an item. The brackets are omitted if an itemset has only one item. An item can occur at most one time in an element of a sequence but it can occur multiple times in different

elements of a sequence. The size |s| of a sequence is the number of elements in the sequence. The length, l(s), is the total number of items in the sequence. A sequence with length l is called an l-sequence. A sequence database, D = {S_1, S_2, .., S_n}, is a set of tuples ⟨sid, s⟩, where sid is a sequence identifier and S_k is an input sequence. A sequence $\alpha = \langle X_1, X_2, .., X_n \rangle$ is called a subsequence of another sequence $\beta = \langle Y_1, Y_2, .., Y_m \rangle$ ($X \leq Y$), and β is called a super sequence of α if there exist an integer $1 \leq i_1 < \ldots < i_1 < m$ such that $X_1 \subseteq Y_{i\,1}, \ldots, \subseteq X_n \subseteq Y_{i\,n.}$ A tuple (sid, S) is said to contain a sequence S_a if S is a super sequence of S_a. The support of a sequence S_a in a sequence database D is the number of tuples in SDB that contains S_a. Given a support threshold, min_sup, a sequence S_a is a frequent sequence in the sequence database if the support of the sequence S_a is no less than a minimum support threshold. The problem of sequential pattern mining is to find the complete set of sequential patterns in the database with a support constraint.

Table 10. A sequence database as a running example in WSpan

Sequence ID	**Sequence**
10	<a (abc) (ac) d (cf)>
20	<(ad) c (bc) (ae) bc>
30	<(ef) (ab) (df) cb>
40	<eg (af) cbc>
50	<a (ab) (cd) egh>
60	<a (abd) bc>

Example 6.1: Table 10 shows the input sequence database SDB in our running example. Assume that a minimum support is 2. The SDB has eight unique items, and six input sequences. A sequence ⟨a (abc) (ac) d (cf)⟩ in SDB has five elements: a, (abc), (ac), d, (cf) where items "a" and "c" appear three times in different elements of the sequence. The size of ⟨a (abc) (ac) d (cf)⟩ is 5 and the length of this sequence is 9. Sequence ⟨a (bc) d⟩ is a sub sequence of ⟨a (abc) (ac) d (cf)⟩ since sequences 10 and 20 contain sub sequence s =⟨a (bc) d⟩ and the support of the sequence is no less than 2. Therefore, s is a sequential pattern of length 4.

6.1.2 Related work

No sequential pattern mining algorithms, to date, have considered weighted sequences. Instead, all the sequential pattern mining algorithms suggested so far have given the same importance to the sequences and the elements in a sequence. However, it is important to distinguish important sequences from a large number of sequence patterns. Previous studies have suggested efficient algorithms [1], [4], [14], [16], [25], [33], [50], [59], [60], [62], [69], [74], [75], [93] for mining sequential patterns.

GSP [69] mines sequential patterns based on an Apriori-like approach by generating all candidate sequences. This is inefficient and ineffective. To overcome this problem, the database projection growth based approach, FreeSpan [33], was developed. Although FreeSpan outperforms the Apriori based GSP algorithm, FreeSpan may generate any substring combination in a sequence. The projection in FreeSpan must keep all sequences in the original sequence database without length reduction.

PrefixSpan [59], a more efficient pattern growth algorithm was proposed which improves the mining process. The main idea of PrefixSpan is to examine only the prefix subsequences and project only their corresponding suffix subsequences into projected databases. In each projected database, sequential patterns are grown by exploring only local frequent patterns.

In SPADE [93], a vertical id-list data format was presented and the frequent sequence enumeration was performed by a simple join on id lists. SPADE can be considered as an extension of vertical format based frequent pattern mining.

SPAM [1] utilizes depth first traversal of the search space combined with a vertical bitmap representation of each sequence. Before SPAM, SPADE and PrefixSpan were two of the fastest algorithms. According to performance evaluations [1], SPAM outperforms SPADE on most datasets and PrefixSpan outperforms SPAM slightly on very small datasets. Except for this case, SPAM outperforms PrefixSpan. Therefore, WSpan will be compared with SPAM for performance evaluation.

6.2 WSpan: Weighted sequential pattern mining with a weight range and a minimum weight

6.2.1 Overview of WSpan

We propose an efficient method for weighted sequential pattern mining to tackle these problems of previous sequential pattern mining. Our main goal in this framework is to push weight constraints into the sequential pattern growth algorithm while keeping the downward closure property. In WSpan, a weight range is applied, items are given different weights within the weight range to reflect characteristics of the real dataset, and the weights of sequences are calculated. The weight range is utilized to generate a reasonable number of weighted sequential frequent patterns even in a dense database with a low minimum support. In addition, both the weight and support of each item are considered separately for pruning the search space. An extensive performance study shows that the number of weighted sequential patterns can be easily adjusted by setting a weight range and the runtime is efficient.

The main contributions of this approach are: 1) introduction of the concept of weighted sequential patterns, 2) classification and incorporation of two key features, a weight and a support, 3) description of weighted sequential pattern mining by using a weight range and a minimum weight, 4) analysis of applications of weighted sequential patterns, 5) implementation of our algorithm, WSpan, and 6) execution of an extensive experimental study to compare the performance of our algorithm with a recently developed algorithm, SPAM [1].

6.2.2 Preliminaries

The problem of weighted sequential pattern mining is to find the complete set of weighted sequential patterns in the database with a support constraint and a weight constraint. The main consideration in applying a weight constraint to sequential pattern mining is that the downward closure property may be broken by simply applying a weight constraint. A sequence with lower weight can be a frequent sequential pattern by combining items with higher weight in the sequence. To solve problems, a weight range is applied to give weight boundaries.

Definition 6.1 Weight Range (WR)

A weight of an item is a non-negative real number that shows the importance of each item. The weight of each item is assigned to reflect the importance of each item in the sequence database. A weight is given to an item within a weight range, $W_{min} \leq WR \leq W_{max}$.

Definition 6.2 Minimum weight constraint (min_weight)

In WSpan, we want to give a balance between the two measures of weight and support. Therefore, we define a minimum weight constraint like a minimum support (min_sup) in order to prune items which have lower weights.

Definition 6.3 Weighted sequence and maximum weight

We can use the term, weighted sequence to represent a set of weighted sequential patterns. Weights of weighted sequences are obtained by calculating the average value of the weights in items of a sequence. A maximum weight (MaxW) is defined as a value of the maximum weight of items in a sequence database or a projected database. In mining

weighted sequential patterns, the number of weighted sequential patterns can be decreased if MaxW is less than one, otherwise, the number of weighted sequential patterns will be increased.

Definition 6.4 Useless sequence pattern

A sequential pattern is called a useless sequential pattern if the support of the sequential pattern is less than a minimum support (min_sup) and its weight is also less than a minimum weight (min_weight).

Table 11. Example sets of items with different weight ranges

Item	<a>	<b>	<c>	<d>	<e>	<f>	<g>	<h>
Support	6	6	6	5	4	3	2	1
Weight ($0.7 \leq WR_1 \leq 1.3$)	1.1	1.0	0.9	1.0	0.7	0.9	1.3	1.2
Weight ($0.7 \leq WR_2 \leq 0.9$)	0.9	0.75	0.8	0.85	0.75	0.7	0.85	0.8
Weight ($0.2 \leq WR_3 \leq 0.6$)	0.5	0.2	0.6	0.4	0.6	0.3	0.5	0.3

Example 6.2: Given the sequence database in Table 10, the examples of sets of items with different weights in Table 11 and minimum support = 2, the set of items in the database, i.e., length-1 subsequences in the form of "item:support" is <a> : 6, <b> : 6, <c> : 6, <d> : 5, <e> : 4, <f> : 3, <g> : 2, <h> : 1. When WR_2 is used, the weight of a sequence <a (abc) (ac) d (cf)> is 0.8 ((0.9 + 0.75 + 0.8 + 0.85 + 0.7) / 5). MaxW of the weight range is the maximum value of a weight range. For example, the MaxW of WR_1, WR_2, and WR_3 is 1.3, 0.9 and 0.6 respectively.

6.2.3 Weighted sequential patterns

In this section, we suggest the concept of weighted sequential patterns, and show their properties. A sequential pattern is a weighted infrequent sequential pattern if, following pruning, condition 6.1 or condition 6.2 is satisfied. If the sequential pattern does not satisfy both of these, the sequential pattern is called a weighted frequent sequential pattern.

Pruning condition 6.1 (support < min_sup && weight < min_weight)

The support of a sequential pattern is less than a minimum support and the sequence weight is less than a minimum weight constraint.

Pruning condition 6.2 (support * MaxW < min_sup)

In a sequence database, the value of multiplying the support of a sequential pattern with a maximum weight among items in the sequence database is less than a minimum support. In projected sequential databases, the value of multiplying the support of a sequence with a maximum weight of items in the projected sequential databases is less than a minimum support. Note that MaxW is used to maintain downward closure property.

Lemma 6.1 When two conditions are applied to prune weighted infrequent sequential patterns, the case in which only pruning condition 6.1, but not pruning condition 6.2, is satisfied for pruning weighted infrequent sequential patterns, is that a MaxW of a sequence database should be greater than one.

Proof: In this case, pruning condition 6.1, but not pruning condition 6.2 in the above, should be satisfied in order to prune a sequential pattern. That is, in condition 6.1, the

support of a sequence is less than a min_sup and the weight of the sequential pattern is less than a min_weight. However, the value of multiplying the support with a MaxW of a sequential pattern should be no less than a minimum support. We can see that the following two formulas should be satisfied.

Formula 1: support < min_sup

Formula 2: support * MaxW ≥ min_sup

We know that the MaxW of a sequence database or projected databases must be no less than one in order to satisfy both of the formulas. For example, assume that a minimum support is 5, a minimum weight is 0.8, a support of a sequence is 4, the weight of the sequential pattern is 0.7 and the MaxW of a sequential pattern in SDB is 1.3. We know that pruning condition 6.1 is satisfied but pruning condition 6.2 is not satisfied. Therefore, this sequential pattern is pruned by condition 6.1.

Lemma 6.2 There is no limitation on using pruning condition 6.2. That is, pruning condition 6.2 (support * MaxW < min_sup) can be applied without limitation.

When only pruning condition 6.2, but not pruning condition 6.1 is satisfied to prune weighted infrequent sequential patterns, a MaxW (Maximum Weight) of a sequence database or projected databases can be any value.

Proof: In this case, a sequential pattern is pruned since pruning condition 6.2 is satisfied although condition 6.1 is not satisfied. We see that the following two formulas should be satisfied.

Formula 3: (support ≥ min_sup || weight ≥ min_weight)

Formula 4: (support * MaxW < min_sup)

If a support of a sequential pattern is no less than a minimum support in Formula 3, MaxW should be less than one to satisfy Formula 4. However, if a weight of a sequential pattern is greater than or equal to a minimum weight threshold in formula 3 and the support of a sequential pattern is less than a minimum support, there is no relationship between Formula 3 and Formula 4. In other words, pruning condition 6.2 (support * MaxW < min_sup) can be applied without limitation.

Lemma 6.3 When two pruning conditions are applied to prune weighted infrequent sequential patterns, the method always prunes more than the approach of using only a minimum support when a MaxW of the transaction database or projected databases is less than one.

Proof: In normal frequent sequential pattern mining, every item has the same priority. That is, their weights are 1.0. If pruning condition 6.2 is only considered, we can see that more sequential patterns will be pruned when weights of items are set to less than one. For example, assume that a minimum support is 4 and the support of a sequential pattern is 5. In normal sequential pattern mining, the sequential pattern is not pruned since weights of items in the sequence are 1.0 and the support (5) of the sequential pattern is greater than a minimum support (4). However, the sequential pattern is pruned when the weight of the sequential pattern is 0.7 by condition 6.2 in section 6.2.3.

Example 6.3: The columns in Table 12 show the set of weighted sequential patterns after pruning weighted infrequent sequential patterns using pruning condition 6.2 by applying different WRs. For example, when WR_3 is applied and a minimum support is 2, pattern f's support is 3, MaxW is 0.6 and the value (1.8) of multiplying the sequential

pattern's support (3) with a MaxW (0.6) in the SDB is less than minimum support (2), so a pattern "f" in each sequence in SDB can be removed. Meanwhile, the number of weighted sequential patterns can be increased when WR_1 is used as a weight range. The support of a pattern "g" in the sequence database is 2. However, a maximum weight is 1.3 so the value (2.6) of multiplying pattern's support (2) with a MaxW (1.3) of a pattern is greater than a minimum support (2), so the pattern "g" is not pruned in the weighted sequence list.

Table 12. Weighted sequential patterns with different weight ranges

SID	**Weighted Sequence List** ($0.7 \leq WR_1 \leq 1.3$)	**Weighted Sequence List** ($0.7 \leq WR_2 \leq 0.9$)	**Weighted Sequence List** ($0.2 \leq WR_3 \leq 0.6$)
10	<a(abc)(ac)d(cf)>	<a(abc)(ac)d(cf)>	<a(abc)(ac)dc>
20	<(ad)c(bc)(ae)bc>	<(ad)c(bc)(ae)bc>	<(ad)c(bc)(ae)bc>
30	<(ef)(ab)(df)cb>	<(ef)(ab)(df)cb>	<e(ab)dcb>
40	<eg(af)cbc>	<e(af)cbc>	<eacbc>
50	<a(ab)(cd)egh>	<a(ab)(cd)e>	<a(ab)(cd)e>
60	<a(abd)bc>	<a(abd)bc>	<a(abd)bc>

Example 6.4: Let us show another example by changing a minimum weight. In this example, Table 10 and Table 11 are used for a sequence database and the weight range respectively. Assume that a weight range is $0.2 \leq WR_3 \leq 0.6$ and a minimum support is 3. Then, the pruning condition 6.1 is applied as follows. If a minimum weight is 0.6, items "g" and "h" in each sequence are pruned. If a minimum weight is 0.4, the item "h" in each sequence is only pruned. Meanwhile, no item in each sequence is pruned if a

minimum weight is less than 0.4. In a similar way, the number of weighted sequential patterns can also be adjusted by using a minimum weight.

6.2.4 Mining weighted sequential patterns

We introduce a weighted sequential prefix pattern growth method called WSpan. It computes a local frequent sequence pattern of a prefix by scanning its projected database. The projection is based on a frequent prefix. By considering the support and weight of sub-sequences, the weighted sequential patterns are mined. In the following, we use the sequence database SDB in Table 10 and apply $0.7 \leq WR_2 \leq 0.9$ as a weight range from Table 11. Assume that min_sup is 2. In WSpan, mining weighted sequential pattern is performed as follows.

Step 1: Find length-1 weighted sequential patterns.

Scan the sequence database once, count the support of each item, check the weight of each item and find all the weighted frequent items in sequences. Although supports of items in sequences are lower than the minimum support and infrequent, the items can not be deleted since infrequent items in sequences may become weighted itemsets in the next step. The weighted infrequent items are found and removed according to pruning condition 6.1 and 6.2. For instance, assume that WR_2 is used as a weight range and a minimum support is 2. After the first scan of the transaction database, items “g”, and “h” are pruned because the value (1.8) of multiplying the support (2) of the sequential patterns with a maximum weight (0.9) is less than a minimum support (2). Therefore, length-1 frequent sequential patterns are <a> : 6, <b> : 6, <c> : 6, <d> : 5, <e> : 4, <f> : 3, the weight list is <a:0.9 b:0.75, c:0.8, d:0.85, e: 0.75,

f:0.7>, and the MaxW is 0.9. Note that an item with a low support can be a weighted frequent item if it has a higher weight.

Step 2: Divide search space.

The complete set of weighted sequential patterns can be partitioned into following seven subsets according to seven prefixes: (1) the ones having prefix <a>; (2) the ones with prefix <b>, (3) the ones with prefix <c>, (4) the ones with prefix <d>, (5) the ones with prefix <e>, and (6) the ones with prefix <f>.

Step 3: Find subsets of sequential patterns.

The subsets of sequential patterns can be mined by constructing the corresponding set of projected databases and mining them recursively. The process of weighted sequential pattern mining in WSpan is explained as follows.

A. Find weighted sequential patterns with prefix <a>

We only collect the sequences which have <a>. Additionally, in a sequence containing <a>, only the subsequence prefixed with the first occurrence of <a> should be considered. For example, in sequence <a (abc) (ac) d (cf)>, only the subsequence <(abc) (ac) d (cf)> are considered and in sequence <(ad) c (bc) (ae) bc>, only the sequence <(~d) c (bc) (ae) bc> are collected. The sequences in sequence database, SDB containing <a> are projected with regards to <a> to form the <a>-projected database, which consists of six suffix sequences: <(abc) (ac) d (cf)>, <(~d) c (bc) (ae) bc>, <(~b) (df) cb>, <(~f) cbc> <(ab) (cd) e> and <(abd) bc>. By scanning <a> projected database once, its local items are a:4, b:6, c:6, d:4, e:2, f:2, (~b):4, (~d):1, (~e):1 and (~f):1.

However, in WSpan, the pruning conditions defined are used. From Table 11, we know that the maximum weight of a sequence database is 0.9 and the locally frequent items which have two as a support are also removed because the value (1.8) of multiplying the support (2) of the sequential patterns with a maximum weight (0.9) is less than a minimum support (2). Therefore, in WSpan, not only (~d):1 (~e):1, and (~f):1 but also e:2 and f:2 are removed. All the length-2 weighted sequential patterns prefixed with <a> are: **<aa>:4, <ab>:6, <ac>:6, <ad>:4, and <(ab)>:4.** Previous sequential pattern mining algorithms only consider a support as a measure in each projected database, so sequential patterns with one as the support are only pruned. However, in WSpan, before constructing the next projected database, pruning conditions are applied in order to check whether it is a weighed sequential pattern. Although the supports of the sequential patterns are equal to the minimum support, the sequential patterns with a lower weight can be deleted. Recursively, all the sequential patterns with prefix <a> can be partitioned into four subsets: 1) those prefixed with <aa>, 2) those prefixed with <ab>, 3) those prefixed with <ac>, 4) those prefixed with <ad>, and finally, 5) those prefixed with <(ab)>. These subsets can be mined by constructing respective projected databases and mining each recursively as follows.

1) The **<aa>** projected database consists of four suffix subsequences prefixed with <aa>: <(~bc) (ac) dc)>, <bc>, <(~b) (cd)>, and <(~bd) bc>. By scanning <aa> projected database once, its local items are a:1, b:2, c:4, d:2, (~b):3, and (~c):1. In WSpan, items a:1, b:2, d:2 and (~c):1 are pruned according to the pruning conditions because the values (0.9/1.8) of multiplying the supports (1/2) of the items with a

maximum weight (0.9) are less than a minimum support (2). Therefore, <aa> projected database returns four sequential patterns: **<aac>:4, and <a(ab)>:3**.

2) The **<ab>** projected database consists of five suffix subsequences prefixed with <ab>: < (~c) (ac) dc >, <(~c) a bc>, <c>, <(cd)> and < (~d) bc>. By scanning <ab> projected database once, its local items are a:2, b:2, c:5, d:2, (~c):2, and (~d):1. Items a:2, b:2, d:2, (~c):2 and (~d):1 are not weighted frequent so the items are pruned according to the pruning conditions. Therefore, <ab> projected database returns one sequential pattern: **<abc>:5.**

3) The **<ac>** projected database consists of five suffix subsequences prefixed with <(ac) dc>, <(bc) abc>, <b>, <bc>, <(~d) >. By scanning <ac> projected database once, its local items are a:2, b:3, c:3, d:1 and (~d):1. Not that the items, d:1 and (~d):1 are infrequent and the item, a:2 is frequent. Previous sequential pattern mining algorithms prune only items d:1 and (~d):1 but WSpan prunes the item, a:2 because it is a weighted infrequent item according to pruning conditions. Therefore, in WSpan, weighted sequential patterns for <ac> projected database are **<acb>:3, and <acc>:3**.

4) The **<ad>** projected database consists of three suffix subsequences prefixed with <ad>: <c>, <cb> and <bc>. By scanning <ad> projected database once, its locally frequent items are b:2, and c:3. In WSpan, after pruning b:2 by pruning conditions, <ad> projected database returns only **<adc>:3** as a weighted sequential pattern.

5) The **<(ab)>** projected database consists of four suffix subsequences prefixed with <ad>: <(~c) (ac) dc>, <dcb>, <(cd)> and <(~d) bc>. By scanning <(ab)> projected database once, its local items are a:1, b:2, c:4, d:3, (~c):1 and (~d):1. In WSpan, items

a:1, b:2, (~c):1 and (~d):1 are pruned from pruning conditions so <(ab)> projected database returns only **<(ab)c>:4** and **<(ab)d>:3** as weighted sequential patterns.

B. Find weighted sequential patterns with prefix <b>, <c>, <d>, <e>, and <f> respectively. This can be done by constructing the <b>, <c>, <d>, <e>, and <f> projected databases and mining them, respectively.

Step 4: The set of sequential patterns is the collection of patterns found in the above recursive mining process. In this example, the number of weighted sequential patterns becomes fewer than the number of sequential patterns generated in the previous sequential pattern mining algorithms because WSpan consider a weight constraint as well as a support constraint. We can see that WSpan can reduce the number of sequential patterns by adjusting a weight range and a minimum weight.

6.2.5 WSpan algorithm

WSpan pushes weight constraints into the projection based sequential pattern mining approach. A weight range and a minimum weight are used and items are given different weights within the weight range. We now show the weighted sequential pattern mining process and present the mining algorithm.

WSpan algorithm: Weighted Sequential pattern mining with a weight range and a minimum weight in large sequence databases.

Input: (1) A sequence database: SDB,

(2) The minimum support threshold: min_sup,

(3) The minimum weight threshold: min_weight,

(4) The weights of the items within weight range: w_i,

Output: The complete set of weighted sequential patterns.

Begin

1. Let WSP be the set of Weighted Sequential Patterns that satisfy the constraints.

Initialize WSP ← {};

2. Scan SDB once, count the support of each item, check the weight of each item and find each weighted frequent item, β, in sequences satisfying the following pruning conditions: β is a weighted sequential item if the following pruning conditions are not satisfied.

Condition 1: (support < min_sup && weight < min_weight)

Condition 2: (support * MaxW < min_sup)

3. For each weighted frequent item, β, in SDB

Call WSPan (WSP, <β>, 0, SDB)

End for

End

Procedure WSpan (WSP, α, L, S|α)

Parameter:

(1) α is a weighted sequential pattern that satisfies the above pruning conditions,

(2) L is the length of α,

(3) $S|_\alpha$ is the sequence database, SDB if α is null, otherwise, it is the α-projected database.

1. Scan S | $_\alpha$ once, count the support of each item, and find each weighted frequent item, β in sequences satisfying the following pruning conditions: β is a weighted sequential item if the following pruning conditions are not satisfied.

Condition 1: (support < min_sup && weight < min_weight)

Condition 2: (support * MaxW < min_sup)

(a) β can be assembled to the last element of α to form a sequential pattern or

(b) <β> can be appended to α to form a sequential pattern.

2. For each weighted frequent item β,

Add it to α to form a sequential pattern α`, and output α`.

3. For each α`,

Construct α`-projected database S| α`;

Call WSpan (α`, L+1, S | α`)

After WSpan algorithm calls the procedure WSpan (WSP, <β>, 0, SDB), WSpan (α`, L+1, S | α`) is called recursively after α` projected database S|a` is constructed.

6.2.6 Performance evaluation

In this section, we present our performance study over various datasets. The WSpan is the first sequential pattern mining algorithm to consider weight of items within sequences and sequential patterns. We report our experimental results on the performance of WSpan in comparison with a recently developed algorithm, SPAM [1], which is the fastest algorithm for mining sequential patterns. The main purpose of this experiment is to demonstrate how effectively the weighted sequential patterns can be

generated by incorporating a weight measure with a support measure. First, we show how the number of weighted sequential patterns can be adjusted through user feedback, the efficiency in terms of runtime of the WSpan algorithm, and the quality of weighted sequential patterns. Second, we show that WSpan has good scalability against the number of sequence transactions in the datasets.

Table 13. Parameters for IBM Quest sequence data generator

Symbol	Meaning
D	Number of customers in 000s in the dataset
C	Average number of transactions per customer
T	Average number of items per transactions
S	Average length of maximal sequences
I	Average length of transactions within the maximal sequences
N	Number of different items in 000s

6.2.6.1 Test environment and datasets and datasets

We used synthetic datasets generated by the IBM dataset generator. Table 13 shows parameters and their meanings in this synthetic sequential dataset generation. More detail information can be found in [4]. WSpan was written in C++ and experiments were performed on a sparcv9 processor operating at 1062 MHz, with 2048MB of memory. All experiments were performed on a Unix machine. In our experiments, a random generation function was used to generate weights for each item.

6.2.6.2 Experimental results

6.2.6.2.1 Comparison of WSpan and SPAM

In this performance test, we focused on the efficiency of using a weight range and a minimum weight. Our experiment shows that in most cases, WSpan outperforms SPAM. First, we evaluate the performance on the D1C10T5S8I5 dataset. We set up different weight ranges but the minimum weight is fixed as a minimum value within weight ranges for testing the effect of the WR.

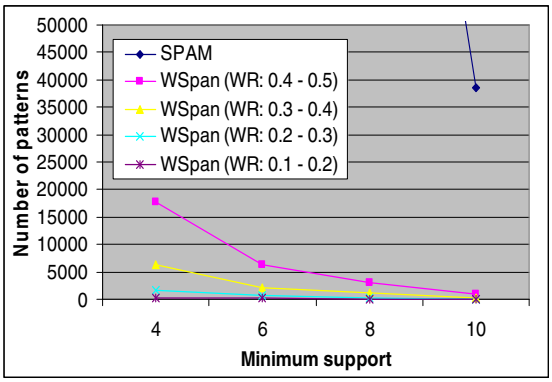

Figure 48. Number of patterns in WSpan (WR: 0.1 – 0.2).

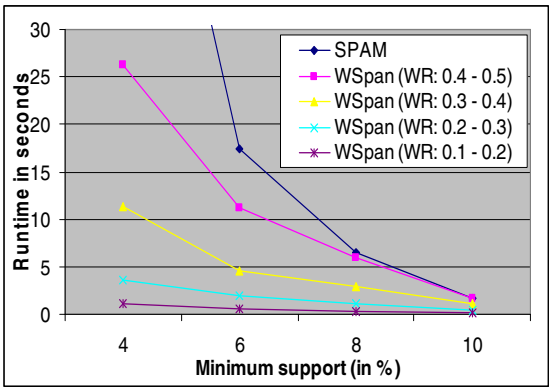

Figure 49. Runtime in WSpan (WR: 0.1 – 0.2).

D1C10T5S8I5 dataset

Figure 48 and Figure 49 show that WSpan generates fewer sequential patterns and runs faster than SPAM. Specifically, fewer sequential patterns are generated as the weight range is decreased. Note that SPAM generates huge sequential patterns and it is much slower as the minimum support is decreased. In SPAM, the number of sequential patterns increases quickly when a minimum support is less than 10 %. Moreover, the runtime becomes much larger as the minimum support is less than 6%. Meanwhile, WSpan generates fewer patterns than SPAM by adjusting a weight range. We can see that WSPan is faster than SPAM. In addition, the number of patterns discovered by WSpan is several orders of magnitude fewer than the number of sequential patterns found by SPAM.

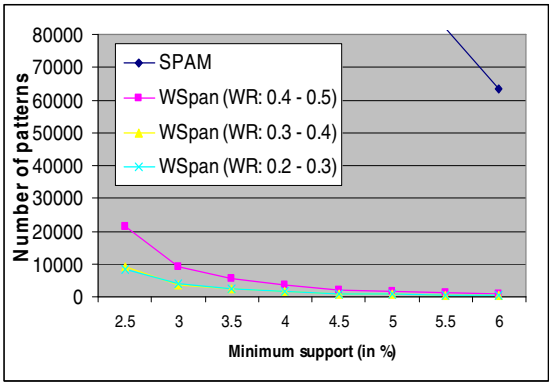

Figure 50. Number of patterns in WSpan (WR: 0.2 – 0.3).

.

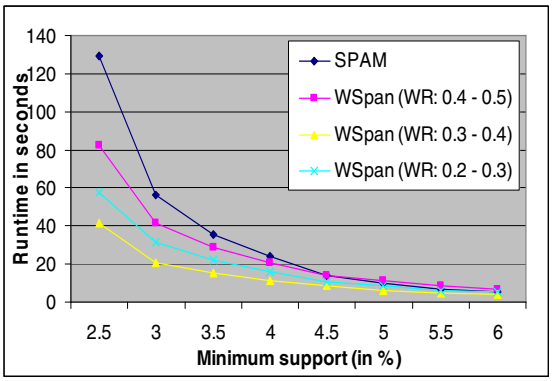

Figure 51. Runtime in WSpan (WR: 0.2 – 0.3).

D7C7T7S7I7 dataset

Figure 50 and Figure 51 demonstrate the results of performance test using the D7C7T7S7I7 dataset by setting different weight ranges from 0.2 to 0.5. WSpan outperforms SPAM and the performance difference becomes larger when the support threshold is lowered. In Figure 50, the number of sequential patterns is increased as the minimum support is decreased, but the number of sequential patterns in SPAM is substantially increased as the minimum support becomes lower. In SPAM, there is no way to adjust the number of sequential patterns except for decreasing the minimum support. Recall that the number of sequential patterns in WSpan can be adjusted by changing the weight ranges. In Figure 51, the runtime of WSpan is faster than SPAM and the difference becomes bigger as a minimum support is lowered.

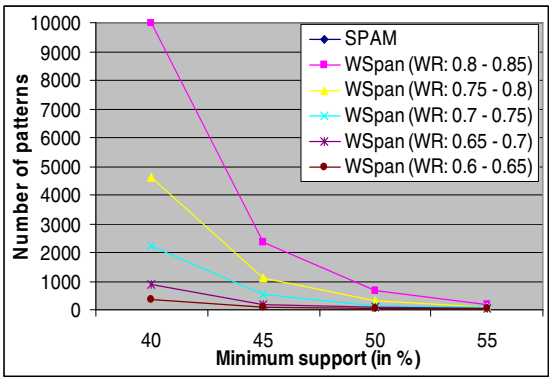

Figure 52. Number of patterns in WSpan (WR: 0.6 – 0.65).

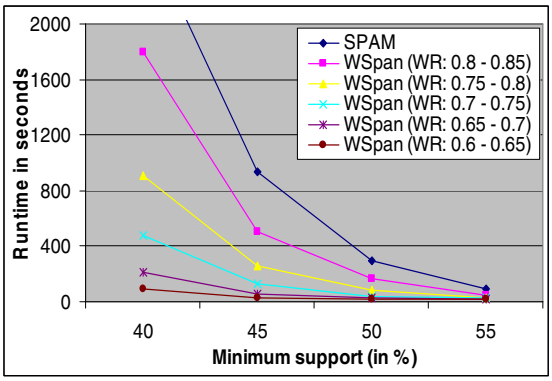

Figure 53. Runtime in WSpan (WR: 0.6 – 0.65).

D15C15T15S15I15 dataset

In Figure 52 and Figure 53, we report the evaluation results for D15C15T15S15I15 dataset. We set up different weight ranges. The main performance difference between WSpan and SPAM results from using a weight range. By decreasing the support threshold, the number of sequential patterns of WSpan is increased but the number of SPAM is extremely increased. In Figure 52, we could not show the number of sequential patterns of SPAM, because the number of sequential patterns mined by

SPAM is huge with the minimum support of less than 55%. For example, the numbers of sequential patterns in SPAM are 449,403 with a minimum support of 55%, 1,365,328 with a minimum support of 50%, 44,062,294 with a minimum support of 45%, and so on. In Figure 53, WSpan is faster than SPAM.

Our above experiments showed that WSpan can generate fewer but important weighted frequent sequential patterns with various weight ranges in several datasets. In previous mining algorithms, it is difficult to reduce frequent sequential patterns without changing the minimum support but WSpan can reduce the number of sequential patterns by adjusting weight ranges when giving weights to each item.

Table 14. Effectiveness of a minimum weight in WSpan
(D7C7T7S7I7 dataset)

Minimum support	Number of W.S.P WR:0.8 – 1.2 MW : 1.2	Number of W.S.P WR:0.8 – 1.2 MW : 1.0	Number of W.S.P WR:0.8 – 1.2 MW : 0.8	Number of S.P WR:0.8 – 1.2
5 %	56897	79320	113712	124728
5.5 %	36475	48356	66825	81792
6%	27826	36156	48827	63207
6.5%	19394	26204	36475	44262

Table 14 lists the number of Weighted Sequential Patterns (WSP) with various minimum weights by WSpan and Sequential Patterns (SP) generated by SPAM. From Table 14, WSpan can generate fewer WSP by using different Minimum Weight (MW)

thresholds. For example, in Table 14, the numbers of sequential patterns are 63,207. Meanwhile, the number of WSP at a minimum support: 6%, a WR: 0.8 – 1.2 and a minimum weight: 0.8 is 48,827, the number of WSP can be reduced to 36,156 with a minimum weight: 1.0 and can be further reduced to 27,826 with a minimum weight, 1.2. In this way, the proper number of weighted sequential patterns can be found by adjusting a minimum weight.

6.2.6.2.2 Quality of patterns in WSpan

In most test datasets, items are expressed as integer values so it is difficult to understand the meaning of items and discovered sequential patterns. We illustrated that weighted sequential patterns can be used in real applications. In this evaluation, the D7C7T5S4I2.5 dataset is used to illustrate the quality of weighted frequent pattern mining. We analyzed the patterns discovered by WSpan to show the effectiveness of mining weighted frequent patterns. We compared the patterns mined by WSpan with those of SPAM. For example, sequential patterns <(17, 45) (91) (70) (91)>:22 and <(45) (27, 91) (70)>:21 are mined by SPAM with a minimum support, 3%. However, these patterns are pruned by WSpan with a weight range, 0.3 – 0.4 because these sequential patterns are weighted infrequent patterns. Although the minimum support is increased from 3% to 5%, weighted infrequent patterns such as <(17, 45) (91) (70) (91)>:22 and <(45) (27, 91) (70)>:21 are found by SPAM. In other words, the supports of these sequential patterns are more than the minimum support but the weights of the patterns are relatively less important. The weighted sequential mining approach can be effectively applied to discover unusual patterns and detect fraudulent patterns.

6.2.6.2.3 Scalability test

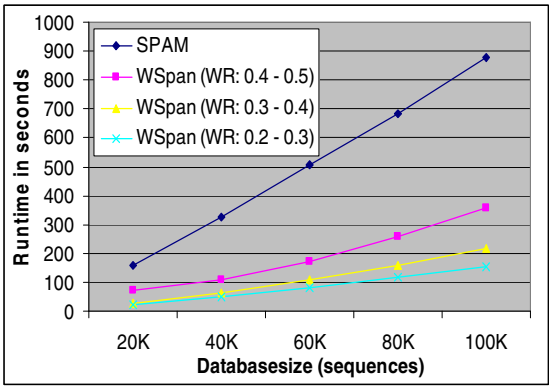

Figure 54. Scalability test (Runtime) in WSpan (Min_sup = 0.4%).

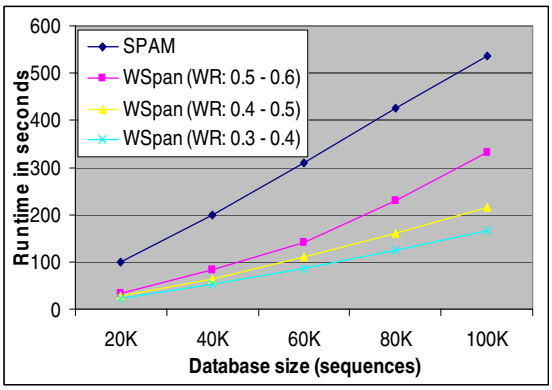

Figure 55. Scalability test (Runtime) in WSpan (Min_sup = 0.5%).

DxC2.5T5S4I2.5 dataset

The DxC2.5T5S4I2.5 dataset is used to test scalability with the number of sequences in a sequence database. From the performance test, WSpan scales much better than SPAM and becomes better by decreasing a weight range. WSpan and SPAM show linear scalability with the number of sequences from 20K to 100K. However, WSpan is much more scalable than SPAM. In Figure 54 and 55, the difference between WSpan and SPAM becomes clear. We set a minimum support as 0.4% and 0.5% respectively, and used different weight ranges. We can see that WSpan has much better scalability in terms of number of sequences in the database and becomes faster as a weight range is increased. Without the weight constraint, in SPAM, the number of sequential patterns and runtime increase dramatically.

6.2.7. Summary

In summary, WSpan is efficient and scalable in weighted sequential pattern mining. WSpan is faster than SPAM which is a recently suggested sequential pattern mining algorithm. Additionally, it becomes much faster and generates fewer but important sequential patterns even with a very low minimum support for larger databases.

6.3 WIS: Weighted interesting sequential pattern mining with a level of support and/or weight affinity

6.3.1 Overview of WIS

In this approach, we suggest an efficient sequential pattern mining algorithm called WIS (Weighted Interesting Sequential pattern mining) based on a level of support and/or weight affinity. We divide the support and weight constraints more specifically and define the concept of a sequential support/weight affinity pattern that uses new measures, called sequential s-confidence and w-confidence which consider support/weight affinity and prevent the generation of sequential patterns with substantially different support and/or weight levels. The sequential s-confidence measure is used to generate patterns with similar levels of support and the sequential w-confidence measure is utilized to identify strong weight affinity patterns, so more meaningful sequential patterns can be generated. An extensive performance analysis shows that weighted interesting sequential patterns are extremely valuable patterns, since they have strong affinity in terms of a support and/or a weight. Users can adjust the number of sequential patterns or find their interesting sequential patterns by using weighted sequential patterns with support and/or weight affinity, instead of only obtaining frequent patterns by changing the minimum support threshold. To decrease the number of thresholds, sequential s-confidence, w-confidence and weighted support can be used selectively.

The main contributions of this approach are: 1) definition of new measures, a sequential s-confidence, and a sequential w-confidence, 2) classification and

incorporation of two key features, a support and a weight, 3) introduction of the weighted sequential affinity pattern in terms of support and weight, 4) description of weighted interesting sequential pattern mining by using sequential s-confidence and/or w-confidence, 4) implementation of our algorithm, WIS, and 5) execution of an extensive experimental study to compare the performance of our algorithm, WIS with SPAM [1] and WSpan.

6.3.2 Sequential affinity patterns

In this section, we define the sequential s-confidence and w-confidence measures, explain the concept of weighted sequential affinity patterns, and show important properties

6.3.2.1 Sequential support affinity pattern

Definition 6.5 Sequential support-confidence (s-confidence)

Support confidence of a sequential pattern $S = \{s_1, s_2, \ldots, s_m\}$, and s_j is $(x_1x_2\ldots x_k)$, where x_t is an item, denoted as sequential s-confidence, is a measure that reflects the overall support affinity among items within the sequence. It is the ratio of the minimum support of items within this pattern to the maximum support of items within the sequential pattern. That is, this measure is defined as

$$\text{S-conf (P)} = \frac{\text{Min}_{\,1 \le m' \le m,\, 1 \le k' \le k}\{\text{support}\,(\{x_{m'k'} \subseteq s_{m'}\})\}}{\text{Max}_{\,1 \le m'' \le m,\, 1 \le k'' \le k}\{\text{support}\,(\{x_{m''\,k''} \subseteq s_{m''}\})\}}$$

Definition 6.6 Sequential support affinity pattern

A sequential pattern is a sequential support affinity pattern if the s-confidence of the sequential pattern is no less than a minimum s-confidence (min_sconf). In other

words, a sequential pattern S is a sequential support affinity pattern if and only if $|S| > 0$ and s-confidence (S) $\geq$ min_sconf.

Property 6.1 Anti-monotone property of sequential s-confidence

The anti-monotone property of sequential s-confidence is similar to that of the support measure used in frequent pattern mining. If the s-confidence of a sequential pattern S is no less than a min_sconf, sequential s-confidence of every subset of the sequential pattern S is also no less than the min_sconf.

Lemma 6.4 Sequential s-confidence has the anti-monotonic property.

Given a sequential pattern, $S = \langle s_1, s_2, \ldots, s_m \rangle$ and s_j is $(x_1 x_2 \ldots x_k)$,where x_t is an item, $\text{Max}_{(1 \leq m'' \leq m,\ 1 \leq k'' \leq k)} \{\text{support}(\{x_{m'' k''} \subseteq s_{m''}\})\}$ of a sequential pattern S is always greater than or equal to that of a sub-sequence of the sequential pattern S and $\text{Min}_{(1 \leq m' \leq m,\ 1 \leq k' \leq k)} \{\text{support}(\{x_{m' k'} \subseteq s_{m'}\})\}$ of the pattern S is always no less than that of a subset of the sequential pattern S. Therefore, we know that

$$\text{S-conf}(P) = \frac{\text{Min}_{1 \leq m' \leq m,\ 1 \leq k' \leq k} \{\text{support}(\{x_{m'k'} \subseteq i_{m'}\})\}}{\text{Max}_{1 \leq m'' \leq m,\ 1 \leq k'' \leq k} \{\text{support}(\{x_{m'' k''} \subseteq i_{m''}\})\}}$$

$$\leq \frac{\text{Min}_{1 \leq m' \leq m-1,\ 1 \leq k' \leq k} \{\text{support}(\{x_{m' k'} \subseteq s_{m'}\})\}}{\text{Max}_{1 \leq m'' \leq m-1,\ 1 \leq k'' \leq k} \{\text{support}(\{x_{m'' k''} \subseteq s_{m''}\})\}}$$

$$\leq \frac{\text{Min}_{1 \leq m'-2 \leq m,\ 1 \leq k' \leq k} \{\text{support}(\{x_{m' k'} \subseteq s_{m'}\})\}}{\text{Max}_{1 \leq m''-2 \leq m,\ 1 \leq k'' \leq k} \{\text{support}(\{x_{m'' k''} \subseteq s_{m''}\})\}}$$

That is, if the s-confidence of a sequential pattern is greater than or equal to a min_sconf, so is every subset of size m - 1. Therefore, the sequential s-confidence can be used to prune the exponential search space.

Property 6.2 Cross support sequential pattern property

A pattern is called a cross support sequential pattern if the sequential pattern contains items which have different levels of supports. Given a min_sconf as a threshold, if sequential s-confidence has the cross support sequential pattern property, for any cross support sequential pattern S with regard to a min_sconf, the value of the sequential s-confidence is less than the min_sconf.

Lemma 6.5 Sequential s-confidence has cross support property.

Given definition 6.5, assume that there is a cross support pattern S = $\{s_1, s_2, ..., s_m\}$ that contains at least two items X and Y such that support ({X}) / support ({Y}) < t. where 0 < t < 1.

$$\text{S-conf (P)} = \frac{\text{Min}_{\,1 \le m' \le m,\, 1 \le k' \le k}\{\text{support}(\{x_{m'\,k'} \subseteq s_{m'}\})\}}{\text{Max}_{\,1 \le m'' \le m,\, 1 \le k'' \le k}\{\text{support}(\{x_{m''\,k''} \subseteq s_{m''}\})\}}$$

$$\le \frac{\text{Min}_{\,1 \le m' \le m,\, 1 \le k' \le k}\{..., \text{support}(\{X\}), ..., \text{support}(\{Y\}), ...\}}{\text{Max}_{\,1 \le m'' \le m,\, 1 \le k'' \le k}\{..., \text{support}(\{X\}), ..., \text{support}(\{Y\}), ...\}}$$

$$\le \frac{\text{support}(\{X\})}{\text{Max}_{1 \le m'' \le m,\, 1 \le k'' \le k}\{..., \text{support}(\{X\}), ..., \text{support}(\{Y\}), ...\}}$$

$$\le \frac{\text{support}(\{X\})}{\text{support}(\{Y\})} < t$$

Therefore, we know that the value of the sequential s-confidence is less than the min_sconf for any cross support sequential pattern S with regard to a sequential s-confidence threshold, t.

6.3.2.2 Sequential weight affinity pattern

Definition 6.7 Sequential weight-confidence (w-confidence)

Weight confidence of a sequential pattern $S = \{s_1, s_2, \ldots, s_m\}$, and s_j is $(x_1x_2\ldots x_k)$, where x_t is an item, denoted as sequential w-confidence, is a measure that reflects the overall weight affinity among items within the sequential pattern. It is the ratio of the minimum weight of items within this pattern to the maximum weight of items within the sequential pattern. That is, this measure is defined as

$$\text{W-conf (P)} = \frac{\text{Min}_{\,1 \le m' \le m,\, 1 \le k' \le k}\{\text{weight}\,(\{x_{m'\,k'} \subseteq s_{m'}\})\}}{\text{Max}_{\,1 \le m'' \le m,\, 1 \le k'' \le k}\{\text{weight}\,(\{x_{m''\,k''} \subseteq s_{m''}\})\}}$$

Definition 6.8 Sequential weight affinity pattern

A sequential pattern is a sequential weight affinity pattern if the w-confidence of the sequential pattern is no less than a minimum weight confidence (min_wconf). In other words, a sequential pattern S is a sequential weight affinity pattern if and only if $|S| > 0$ and w-confidence (S) $\geq$ min_sconf.

Property 6.3 Anti-monotone property of sequential w-confidence

The anti-monotone property of weight confidence is similar to that of the support measure used in frequent pattern mining. If w-confidence of a sequential pattern S is no less than min_wconf, sequential w-confidence of every subset of sequential pattern S is also no less than the min_wconf.

Lemma 6.6 Sequential w-confidence has the anti-monotonic property.

From definition 6.7, we can see that $\text{Max}_{\,(1 \le m'' \le m,\, 1 \le k'' \le k)}\{\text{weight}\,(\{x_{m''\,k''} \subseteq s_{m''}\})\}$ of a sequential pattern S is always greater than or equal to that of a sub-sequence

of the sequential pattern S and Min $_{(1 \le m' \le m,\ 1 \le k' \le k)}$ {weight ({$x_{m'\,k'} \subseteq s_{m'}$})} of the pattern S is always no less than that of a subset of the sequential pattern S. Therefore, we know that

$$\text{W-conf (P)} = \frac{\text{Min}_{\,1 \le m' \le m,\ 1 \le k' \le k}\{\text{weight}\,(\{x_{m'\,k'} \subseteq s_{m'}\})\}}{\text{Max}_{\,1 \le m'' \le m,\ 1 \le k'' \le k}\{\text{weight}\,(\{x_{m''\,k''} \subseteq s_{m''}\})\}}$$

$$\le \frac{\text{Min}_{\,1 \le m' \le m-1,\ 1 \le k' \le k}\{\text{weight}\,(\{x_{m'\,k'} \subseteq s_{m'}\})\}}{\text{Max}_{\,1 \le m'' \le m-1,\ 1 \le k'' \le k}\{\text{weight}\,(\{x_{m''\,k''} \subseteq s_{m''}\})\}}$$

$$\le \frac{\text{Min}_{\,1 \le m'-2 \le m,\ 1 \le k' \le k}\{\text{weight}\,(\{x_{m'\,k'} \subseteq s_{m'}\})\}}{\text{Max}_{\,1 \le m''-2 \le m,\ 1 \le k'' \le k}\{\text{weight}\,(\{x_{m''\,k''} \subseteq s_{m''}\})\}}$$

In other words, if w-confidence of a sequential pattern S is no less than a min_wconf, so is every subset of size m - 1. Therefore, the sequential w-confidence satisfies the downward closure property and prunes weak weight affinity patterns.

Property 6.4 Cross weight sequential pattern property

A sequential pattern is called a cross weight pattern if the pattern contains items which have weak affinity. Given a min_wconf as a threshold, if sequential w-confidence has the cross weight property, for any cross weight sequential pattern S with regard to a min_wconf, the value of the sequential w-confidence is less than the min_wconf.

Lemma 6.7 Sequential w-confidence has cross weight property.

Given definition 6.7, assume that there is a cross weight sequential pattern S = {s_1, s_2, ..., s_m} that contains at least two items Z and W such that weight ({Z}) / weight ({W}) < t. where 0 < t < 1.

$$\text{W-conf (P)} = \frac{\text{Min}_{\,1 \le m' \le m,\, 1 \le k' \le k}\{\text{weight}\,(\{x_{m'\,k'} \subseteq s_{m'}\})\}}{\text{Max}_{\,1 \le m'' \le m,\, 1 \le k'' \le k}\{\text{weight}\,(\{x_{m''\,k''} \subseteq s_{m''}\})\}}$$

$$\le \frac{\text{Min}_{\,1 \le m' \le m,\, 1 \le k' \le k}\{..., \text{weight}\,(\{Z\}), ..., \text{weight}\,(\{W\}), ...\}}{\text{Max}_{\,1 \le m'' \le m,\, 1 \le k'' \le k}\{..., \text{weight}\,(\{Z\}), ..., \text{weight}\,(\{W\}), ...\}}$$

$$\le \frac{\text{weight}\,(\{Z\})}{\text{Max}_{1 \le m'' \le m,\, 1 \le k'' \le k}\{..., \text{weight}\,(\{Z\}), ..., \text{weight}\,(\{W\}), ...\}}$$

$$\le \frac{\text{weight}\,(\{Z\})}{\text{weight}\,(\{W\})} < t$$

Therefore, we know that the value of the w-confidence is less than the min_wconf for any cross weight sequential pattern S with regard to a sequential w-confidence threshold, t.

6.3.3 Weighted interesting sequential patterns

In this section, we define weighted interesting sequential pattern mining and show pruning approaches.

Definition 6.9 Weight range (WR)

A weight of an item is a non-negative real number that shows the importance of the item. The weight of each item is assigned to reflect the importance of each item within sequences in the sequence database. A weight is given to an item within a weight range, $W_{min} \le WR \le W_{max}$.

Lemma 6.8 Sequential w-confidence can be applied irrespective of different weight ranges.

WIS uses the weight range to maintain the downward closure property. For example, WR_k of a sequential pattern K = {<A>, <A, B>, <A, B, C>} is from 1 to 3 and

$WR_{k`}$ of a sequential pattern K` = {<D>, <D, E>, <D, E, F>} is from 0.1 to 0.3. Assume that weight ({A}) = 1, weight ({B}) = 2, weight ({C}) = 3, weight ({D}) = 0.1, weight ({E}) = 0.2, and weight ({F}) = 0.3, where weight is the weight value of a sequential pattern. Then, sequential w-confidence (K) = 0.33 and sequential w-confidence (K`) = 0.33. Using $WR_{k`}$ rather than WR_k generates fewer sequential patterns. However, the w-confidences (0.33) of sequential patterns K and K` are the same in spite of different weight ranges. We know that sequential w-confidence is defined as the ratio of the minimum weight of items within this sequential pattern to the maximum weight of items within the sequential pattern. Therefore, if ratios of the minimum weight to the maximum weight of different weight ranges are the same, the effect is the same. In other words, the w-confidence of a sequential pattern is only decided by a level of weight affinity between items of a sequential pattern, not by a weight range.

Definition 6.10 Weighted Interesting Sequential pattern (WIS)

A sequence is a weighted interesting sequential pattern if the following pruning conditions are satisfied. Note that these pruning conditions can be applied selectively. For example, pruning conditions (1, 2), (1, 3) and (1, 2, 3) can be used as desired. Sequential s-confidence and w-confidence can also be used independently.

Pruning condition 1: (support * MaxW ≥ min_sup)

Pruning condition 2: (s-confidence ≥ min_sconf)

The sequential pattern is a sequential support affinity pattern if the s-confidence of a pattern is greater than or equal to a minimum s-confidence.

Pruning condition 3: (w-confidence ≥ min_wconf)

The sequential pattern is a sequential weight affinity pattern if the sequential w-confidence is no less than a min_wconf.

6.3.3.1 Sequential w-confidence VS. s-confidence

Sequential s-confidence is a support measure which is used to identify strong affinity patterns in terms of support and sequential w-confidence is a weight measure that considers the weight affinity of items within a sequential pattern. The previous use of a weight constraint in WSpan can generate many spurious patterns containing different levels of weights or miss interesting low weight patterns. Both measures satisfy the anti-monotone property. Therefore, these measures can be effectively used to prune weak affinity patterns.

6.3.3.2 Sequential w-confidence VS. weighted support constraint

Although weighted support constraint considers weight and support, it can generate weak affinity patterns in terms of weight. Sequential w-confidence uses only weights of items within patterns. Patterns with a high support and a high weight satisfy the weighted support constraint but the w-confidences of these patterns can not satisfy the minimum w-confidence if they are patterns with different levels of weights.

6.3.3.3 Sequential s-confidence VS. support constraint

Sequential s-confidence and support constraint both use a support measure. Support constraint generates weak affinity patterns. Although the sequential patterns with a high support satisfy the support constraint, these sequential patterns can not

satisfy the sequential s-confidence when they are sequential patterns including items with different levels of support.

6.3.4 Mining weighted interesting sequential patterns with support and/or weight affinity

In this section, we show how to mine sequential affinity patterns by using a prefix-based projection approach that computes local frequent sequential patterns of a prefix by scanning its projected database. The projection is based on a frequent prefix. We use the sequence database SDB in Table 15, Assume that min_sup is 2, min_wconf is 0.7, min_sconf is 0.7 and a weight range is $0.4 \leq WR \leq 0.8$ in which the weight list is <a:0.6, b:0.8, c:0.5, d:0.6, e:0.4, f:0.8, g:0.5, h:0.6>. In the WIS, mining weighted interesting sequential patterns is performed as follows.

Table 15. A sequence database as a running example in WIS

Sequence ID	Sequence
10	⟨a (abc) (ac) d (cf)⟩
20	⟨(ad) abc (bcd) (ae) bcde⟩
30	⟨a(ef) b (ab) c (df) ac⟩
40	⟨ac (bc) eg (af) acb (ch) (ef)⟩
50	⟨ba (ab) (cd) eg (hf)⟩
60	⟨a (abd) bc (he)⟩

Step 1: Find length-1 weighted sequential patterns.

Scan the sequence database once, count the support of each item, check the weight of each item and find all the weighted frequent items in sequences. The weighted infrequent items are found and removed according to pruning condition 1 in definition 6.10. After the first scan of the sequence database, length-1 frequent sequential patterns are <a> : 6, <b> : 6, <c> : 6, <d> : 5, <e> : 5, <f> : 4, <h> : 3 and <g> : 2. The weight list is <a:0.6, b:0.8, c:0.5, d:0.6, e: 0.4, f:0.8, h:0.6, g:0.5> and the maximum weight (MaxW) is 0.8. From pruning by weighted support constraint, item "g" is removed because the value (1.6) of multiplying the support (2) of item "g" with a MaxW (0.8) is less than a min_sup (2). Note that an item with a lower support can be a weighted frequent item if it has a higher weight. After the projected database is generated from the sequence database, WIS mines weighted interesting sequential patterns from the projected databases recursively and the weighted interesting patterns are generated by adding items one by one.

Step 2: Divide search space.

The complete set of weighted sequential patterns can be partitioned into the following seven subsets having prefix: (1) <a>; (2) <b>, (3) <c>, (4) <d>, (5) <e>, (6) <f>, and (7) <h>.

Step 3: Find subsets of sequential patterns.

The subsets of sequential patterns can be mined by constructing the corresponding set of projected databases and mining them recursively. The process of weighted sequential pattern mining in WIS is explained as follows.

1. Find weighted sequential patterns with prefix <a>

We only collect the sequences which have <a>. Additionally, in a sequence containing <a>, only the subsequence prefixed with the first occurrence of <a> should be considered. For example, in sequence <a (abc) (ac) d (cf)>, only the subsequence <(abc) (ac) d (cf)> is considered and in sequence <(ad) abc (bcd) (ae) bcde>, only the sequence <(~d) abc (bcd) (ae) bcde> is collected. The sequences in sequence database SDB containing <a> are projected with regards to <a> to form the **<a>-projected database**, which consists of six suffix sequences: <(abc) (ac) d (cf)>, <(~d) abc (bcd) (ae) bcde>, <(ef) b (ab) c (df) ac>, <c (bc) eg (af) acb (ch) (ef)>, <(ab) (cd) eg (hf)> and <(abd) bc (he)>. By scanning the <a> projected database once, its locally frequent items are a:6, b:6, c:6, d:d, e:5, f:4, h:3, g: 2 (~b):4, (~c):1, (~d):1, (~e):1 and (~f):1. In WIS, the pruning conditions in definition 6.10 are used. The locally frequent item "g" which has less than two as a support is removed by weighted support constraint because the value (1.6) of multiplying the support of the sequences with a maximum weight (0.8) is less than a minimum support (2). Therefore, in WIS, (~c):1, (~d):1, (~e) and (~f):1 are also removed. In addition, a local item "e:5" is pruned by sequential w-confidence. The candidate pattern, from a local item "e:5" and a conditional prefix "a" is <ae:5> and the sequential w-confidence (0.68) of the candidate sequential pattern <ac:5> is less than the minimum w-confidence (0.7). Moreover, the candidate pattern <ah:3> is pruned by sequential s-confidence because the s-confidence of the sequential pattern is 0.5 which is less than the minimum s-confidence (0.6). All the length-2 sequential patterns prefixed with <a> are: **<aa>:6, <ab>:6, <ac>:6, <ad>:5 <af>:4 and <(ab)>:4**. Note that

previous sequential pattern mining algorithms only consider a support as a measure in each projected database so sequences <(ac):1> <(ad):1> and <(ae:1> are only pruned because they are not frequent. The recently developed WSpan algorithm uses weighted support constraint. However, in WIS, before constructing the next projected database, sequential w-confidence and s-confidence are applied to prune sequential weak affinity patterns. The final <a>-projected database is generated as follows: **<(abc) (ac) d (cf)>, <(~d) abc (bcd) a bcd>, <fb (ab) c (df) ac>, <c (bc) (af) acbcf>, <(ab) (cd) f> and <(abd) bc>.** Recursively, all the sequential patterns with prefix <a> can be partitioned into six subsets prefixed with: 1) <aa>, 2) <ab>, 3) <ac>, 4) <ad>, 5) <af> and 6) <(ab)>. These subsets can be mined by constructing respective projected databases and mining each recursively as follows.

1) The <aa> projected database consists of six suffix subsequences prefixed with **<(~bc) (ac) d (cf)>, <bc (bcd) abcd>, > <(~b) c (df) ac>, < (~f) acbcf>, < (~b) (cd) f>, and <(~bd) bc>.** By scanning the <aa> projected database once, its local items are a:4, b:4, c:6, d:5, f:4, (~b):4, and (~c):1. The locally frequent item "(~c):1" is pruned by weighted support constraint. The <aa> projected database returns the following sequential patterns: <aaa:4>, <aab:4>, <aac:6>, <aad:5>, <aaf:4> and <(aab):4>. Sequential s-confidence and w-confidence of these patterns are no less than a minimum s-confidence and w-confidence respectively. Recursively, sequential patterns with prefix <aa> are partitioned and mined.

2) The <ab> projected database consists of six suffix subsequences prefixed with <ab>: **<(~c) (ac) d (cf)>, <c (bcd) abcd>, <(ab) c (df) ac>, <(~c) (af) acbcf>, <(cd) f>**

and <(~d) bc>. By scanning the <ab> projected database once, we obtain its local items: a: 4, b:4, c:6, d:4, f:4, (~c):1, and (~d):1. Items (~c):1, and (~d):1 are pruned by weighted support constraints because the value (0.8) of multiplying support (1) of the items with a MinW (0.8) is less than the minimum support (2). In WIS, the sequential candidate pattern, <abf>:4 can be removed by sequential s-confidence because the sequential s-confidence (0.67) of pattern <abf> is less than a min_sconf (0.7). From sequential w-confidence, sequence candidate <abc>:4 is pruned because the w-confidence (0.625) of the sequence candidate <abc>:4 is less than min_wconf (0.7). The final sequential pattern is <abd>:4. Recursively, sequential patterns with prefix <ab> are partitioned and mined.

3) The <ac> projected database consists of five suffix subsequences prefixed with <ad>: **<(ac) d (cf)>, <(bcd) a bcd>, <(df) ac>, <(bc) (af) acbcf>, and <(~d) f>.** By scanning the <ac> projected database once, its local items are a:4, b:2, c:4, d:3, f:4, (~d):1 and (~f):1. Sequential candidate patterns <a(cd)>:1, <a(cf)>:1 and <acb>:2 are pruned by weighted support constraint. The weighted sequential patterns <aca>: 4, <acc>:4 <acd>:3 and <acf>:4 are generated. Recursively, sequential patterns with prefix <ac> are partitioned and mined.

4) The <ad> projected database consists of six suffix subsequences prefixed with **<(cf)>, <abc (bcd) abcd>, <(~f) ac>, <f> and <bc>.** By scanning the <ad> projected database once, its local items are a:2, b:2, c:4, d:1, f:2, and (~f):1. Among these candidate patterns, the only weighted frequent item is c:4, so <ad> projected database

returns one sequential pattern: <adc>:4. Recursively, sequential patterns with prefix <ad> are partitioned and mined.

5) The <af> projected database consists of five suffix subsequences prefixed with <af>: **<b (ab) c (df) ac>, and <acbcf>.** By scanning the <af> projected database once, its local items are a: 2, b:2, c:2, d:1, and f:2. All local items are pruned because they do not satisfy the pruning conditions in definition 6.10. Recursively, sequential patterns with prefix <af> are partitioned and mined.

6) The <(ab)> projected database consists of four suffix subsequences prefixed with <ad>: **<(~c) (ac) d (cf)>, <c (df) ac>, <(cd) f> and <(~d) bc>.** By scanning the <(ab)> projected database once, its local items are a: 2, b:1, c:4 d:3, f:3, (~c):1 and (~d):1. Local items "b:1" "(~c):1 and "(~d):1" are pruned by the weighted support constraint and sequential candidate pattern <(ab)c>:4 is pruned by sequential w-confidence because the w-confidence of the pattern is 0.625 which is less than the minimum w-confidence (0.7). The candidate pattern "(ab)f" is pruned by sequential s-confidence because it is a weak support affinity pattern. Finally, the sequential pattern generated by the <(ab)> projected database is <(ab)d>:3. Recursively, sequential patterns with prefix <(ab)> are partitioned and mined.

2. Mine remaining weighted sequential patterns. This can be done by constructing the **<b>, <c>, <d>, <e>, <f> and <h>** projected databases and mining them, respectively as shown above.

Step 4: The set of sequential patterns is the collection of patterns found in the above recursive mining process. You can see that in this example, the number of weighted

sequential patterns is fewer than the number of sequential pattern generated in the previous sequential pattern mining algorithms because WIS provides pruning by sequential s-confidence and w-confidence as well as weighted support constraint defined in WSpan.

Table 16 shows examples of pruning candidate patterns by weighted support, sequential s-confidence, and w-confidence. In this example, a minimum support is 2 and a minimum s-confidence and w-confidence are 0.7 respectively. The number of sequential patterns can be adjusted by changing minimum thresholds. However, previous sequential pattern mining algorithms still generate weak affinity patterns even though the minimum support is high. By using two objective measures, sequential s-confidence and w-confidence, these weak affinity patterns can be pruned first when the number of patterns need to be reduced. Note that weighted support, sequential s-confidence and w-confidence may be used selectively.

Table 16. Pruning candidate patterns in WIS

Candidate patterns	Weighted support	Sequential w-confidence	Sequential s-confidence
<ae> : 4	(0.8 * 4)	Pruned 0.67 (0.4/0.6)	0.83 (5/6)
<ah> : 3	(0.8 * 4)	1 (0.6/0.6)	Pruned 0.5 (3/6)
<acb> : 2	Pruned (0.8 * 2)	Pruned 0.625 (0.5/0.8)	1 (6/6)
<adb> : 2	Pruned (0.8 * 2)	0.75 (0.6/0.8)	0.83 (5/6)
<(ab)c> : 4	(0.8 * 4)	Pruned 0.625 (0.5/0.8)	1 (6/6)
<(ab)f> : 3	(0.8 * 3)	0.75 (0.6/0.8)	Pruned 0.67 (4/6)

6.3.5 WIS algorithm

WIS pushes sequential s-confidence and w-confidence into the projection based on pattern growth approach and mines strong affinity patterns in terms of support and/or weight. We now show the weighted sequential pattern mining process and present the mining algorithm.

WIS algorithm: Weighted sequential pattern mining with support and/or weight affinity.

Input: (1) A sequence database: SDB,

(2) A support threshold: min_sup,

(3) A w-confidence threshold: min_wconf

(4) A s-confidence threshold: min_sconf

(5) The weights of the items within weight range: w_i,

Output: The complete set of weighted sequential patterns.

Begin

1. Let WSP be the set of Weighted Sequential Patterns that satisfy the constraints.

Initialize WSP ← {};

2. Scan SDB once, count the support of each item, check the weight of each item and find each weighted frequent item, β, in sequences satisfying the following pruning conditions: β is a weighted sequential item if the following pruning conditions are not satisfied.

Condition 1: (support * MaxW < min_sup)

3. For each weighted frequent item, β, in SDB

Call WIS (WSP, <β>, 0, SDB)

End for

End

Procedure WIS (WSP, α, L, S|α)

Parameter:

(1) α is a weighted sequential pattern that satisfies the above pruning conditions,

(2) L is the length of α,

(3) $S|_{\alpha}$ is the sequence database, SDB if α is null, otherwise, it is the α-projected database.

1. Scan $S|_{\alpha}$ once, count the support of each item, and find each weighted frequent item, β in sequences satisfying the following pruning conditions: β is a weighted sequential item if the following pruning conditions are not satisfied.

Condition 1: (support * MaxW < min_sup)

Condition 2: (w-confidence < min_wconf)

Condition 3: (s-confidence < min_sconf)

(a) β can be assembled to the last element of α to form a sequential pattern or

(b) <β> can be appended to α to form a sequential pattern.

2. For each weighted frequent item β,

Add it to α to form a sequential pattern α`, and output α`;

End for

3. For each α`,

Construct α` projected database S| α`;

Call WIS (α`, L+1, S | α`)

End for

6.3.6 Performance evaluation

In this section, we present our performance study over various datasets. The WIS is the first sequential pattern mining algorithm to consider a level of support and/or weight affinity between items of sequential patterns. WIS gives a balance between the two measures of weight and support, but also considers support affinity and/or weight affinity between items within sequential patterns, so more valuable patterns can be generated. We report our experimental results on the performance of WIS in comparison with recently developed algorithms: SPAM [1] and WSpan. The main purpose of this experiment is to demonstrate how effectively the weighted sequential patterns can be generated by using sequential s-confidence and/or w-confidence. First, we show how the number of weighted sequential patterns can be adjusted through user feedback, the efficiency of the WIS algorithm, and quality of weighted sequential affinity patterns. Second, we illustrate that WIS has good scalability against the number of transactions in the datasets.

6.3.6.1 Test environment and datasets

We used synthetic datasets generated by the IBM dataset generator as shown in section 6.2.6. WIS was written in C++. Experiments were performed on a sparcv9 processor operating at 1062 MHz, with 2048MB of memory. All experiments were

performed on a Unix machine. In our experiments, a random generation function was used to generate weights for each item. When running WIS, minimum s-confidence and/or w-confidence are set up as the cut off values for mining weighted sequential patterns.

6.3.6.2 Experimental results

6.3.6.2.1 Comparison of WIS with SPAM and WSpan

Our experiment shows that, in most cases, WIS outperforms SPAM [1] and WSpan. First, we evaluated the performance on the D1C10T5S8I5 dataset.

The effect of sequential w-confidence pruning

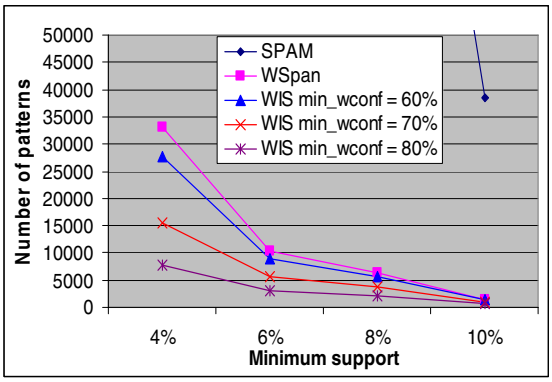

Figure 56. Number of patterns in WIS (sequential w-confidence).

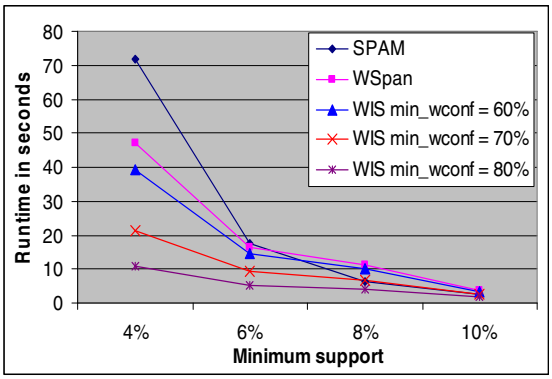

Figure 57. Runtime in WIS (sequential w-confidence).

The effect of sequential s-confidence pruning

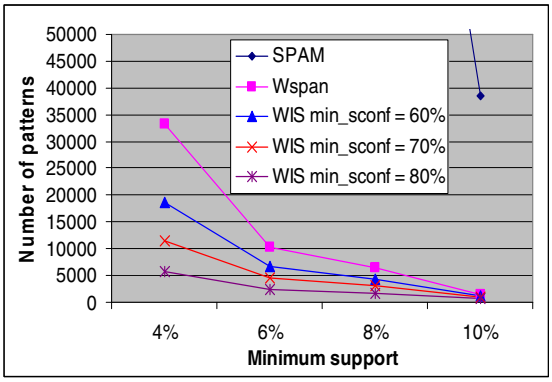

Figure 58. Number of patterns in WIS (sequential s-confidence).

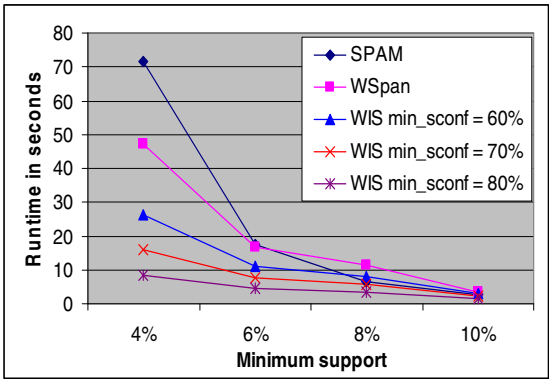

Figure 59. Runtime in WIS (sequential s-confidence).

The effect of combination of sequential s-confidence and w-confidence pruning

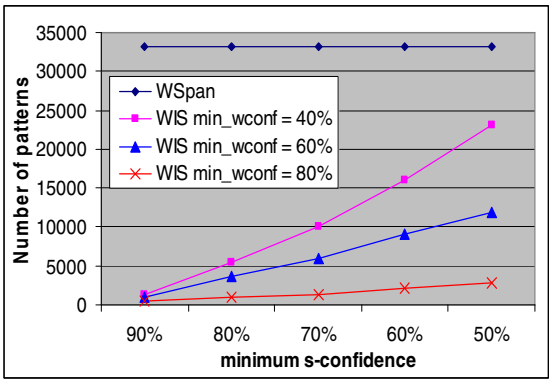

Figure 60. Number of patterns in WIS (Min_sup = 2.0%).

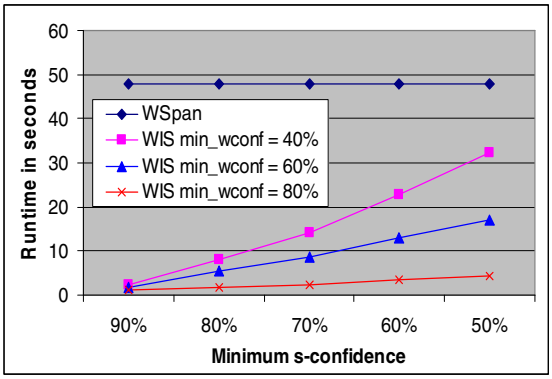

Figure 61. Runtime in WIS (Min_sup = 2.0%).

D1C10T5S8I5 dataset

From Figure 56 to Figure 59, we set up a weight range from 0.3 to 0.6. The number of sequential patterns and runtime of WIS are compared with those of SPAM and WSpan as the minimum support is increased. WIS generates fewer sequential patterns than SPAM and WSpan. Particularly, smaller sequential patterns are generated as the sequential s-confidence in Figure 56, and w-confidence in Figure 58, are increased. In Figure 57, 59 and 61, we can see that WIS is faster than SPAM and

WSpan. In Figure 56 and Figure 57, the sequential s-confidence is only used to prune weighted sequential patterns with weak support affinity. Meanwhile, in Figure 58 and Figure 59, sequential w-confidence is only utilized to remove weighted sequential patterns involving items with different levels of weight. Note that sequential s-confidence and/or w-confidence can be used individually without combination with other measures. Sequential s-confidence and w-confidence in WIS show better performance than SPAM and WSpan. Specifically, the performance gaps increase as the minimum s-confidence and/or the minimum w-confidence increase. SPAM generates a huge number of sequential patterns with a minimum support of less than 10%. For instance, the numbers of patterns of SPAM are 38,615 with a minimum support of 10%, 160,685 with a minimum support of 8%, and 443,639 with a minimum support of 6%. In Figure 60 and Figure 61, a minimum support threshold is fixed at 2% and the performance is evaluated as the minimum s-confidence and w-confidence are changed. The number of sequential patterns discovered by WIS is several orders of magnitude smaller than the number of sequential patterns found by WSpan. From performance results in Figure 60 and Figure 61, we see that sequential s-confidence and w-confidence can be individually used without a minimum support. Note that, in Figure 60 and Figure 61, the number of sequential patterns and runtime is not changed in WSpan because WSpan does not use sequential s-confidence and w-confidence measures. Although the sequential patterns are generated after decreasing a minimum support threshold, the sequential patterns with weak support and/or weight affinity remain because the minimum support threshold alone can not prune weak affinity patterns. Sequential s-

confidence and w-confidence can be effectively used to prune weak affinity patterns and adjust the number of sequential patterns at constant minimum support.

The effect of sequential s-confidence and/or sequential w-confidence pruning

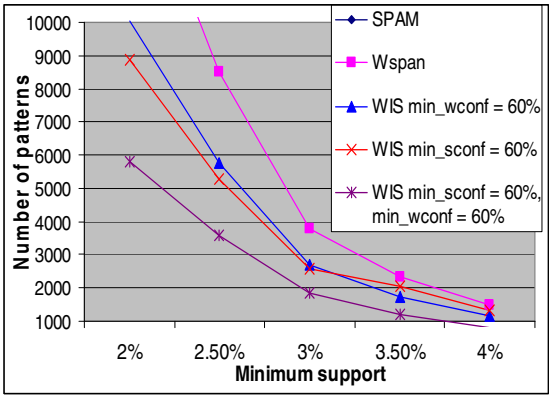

Figure 62. Number of patterns in WIS (min_wconf = 60%).

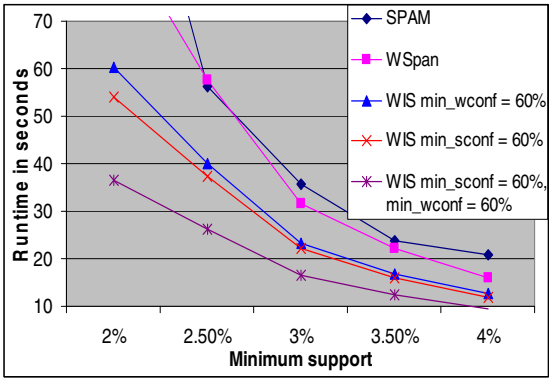

Figure 63. Runtime in WIS (min_wconf = 60%).

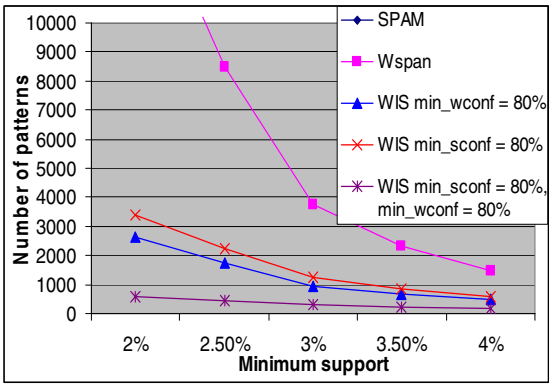

Figure 64. Number of patterns in WIS (min_wconf = 80%).

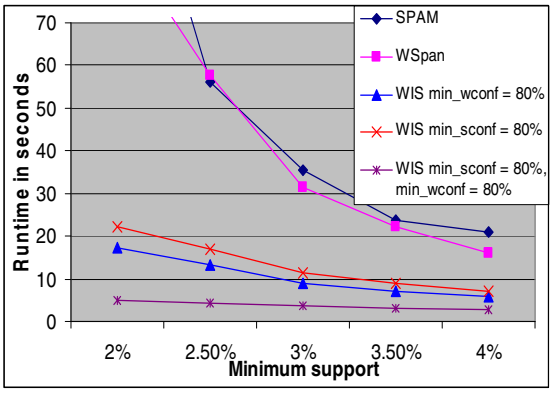

Figure 65. Runtime in WIS (min_wconf = 80%).

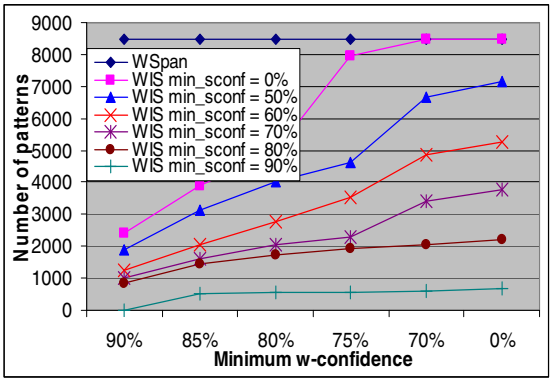

Figure 66. Number of patterns in WIS (Min_sup = 2.5%).

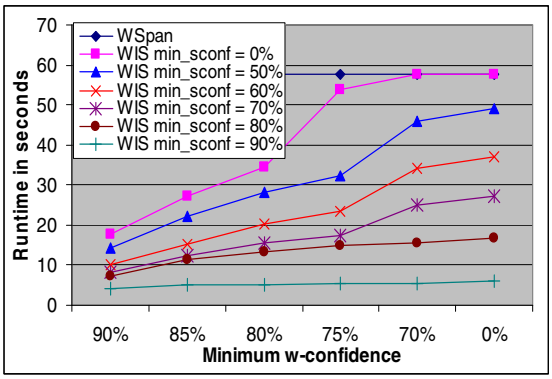

Figure 67. Runtime in WIS (Min_sup = 2.5%).

D7C7T7S7I7 dataset

From Figure 62 to Figure 67, we report the evaluation results for D7C7T7S7I7 dataset. We set up a weight range: from 0.1 to 0.3 for Figure 62 to Figure 65 and from 0.2 to 0.3 for Figure 66 and Figure 67. The main performance difference between WIS and other algorithms such as SPAM and WSpan results from using sequential s-confidence and/or sequential w-confidence. By increasing sequential s-confidence and/or w-confidence thresholds, fewer patterns with a higher level of affinity in terms of support and/or weight can be generated. We can also see that the performance of using both sequential s-confidence and w-confidence is better than using either one alone. In addition, given a minimum s-confidence and w-confidence at 60%, the effect of sequential s-confidence pruning outperforms that of sequential w-confidence. However, at a threshold of 80%, the performance of sequential w-confidence becomes better than that of sequential s-confidence. In most cases, WIS is the fastest among the three algorithms and generates fewer patterns. In Figure 62 and Figure 63 we could not show the number of patterns generated by SPAM because the number of patterns becomes

huge at less than 4%. For example, the number of patterns in SPAM are 170,965 with a minimum support of 4%, 292,161 with a minimum support of 3.5%, 439,953 with a minimum support of 3.0%, 701,760 with a minimum support of 2.5%, and 1,646,818 with a minimum support of 2%.

The effect of combination of sequential s-confidence and w-confidence pruning

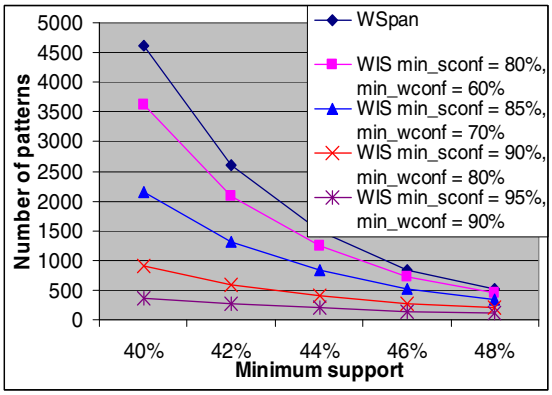

Figure 68. Number of patterns in WIS (WR: 0.4 – 0.8).

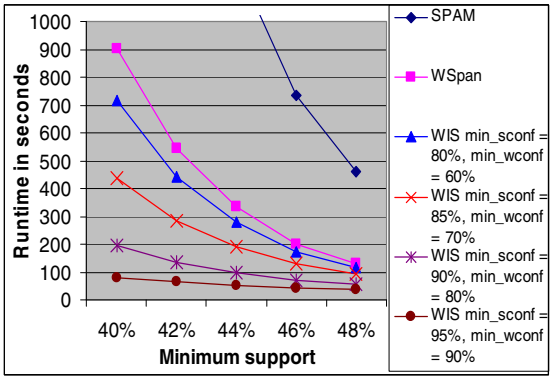

Figure 69. Runtime in WIS (WR: 0.4 – 0.8).

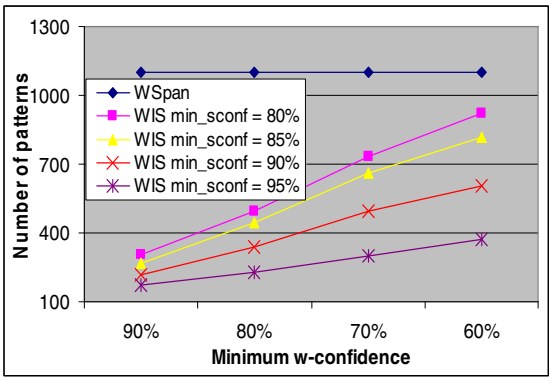

Figure 70. Number of patterns in WIS (Min_sup = 0.4%).

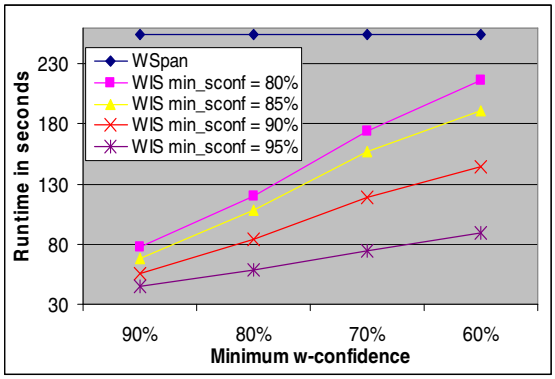

Figure 71. Runtime in WIS (Min_sup = 0.5%).

D15C15T15S15I15 dataset

Figure 68 to Figure 71 demonstrate the results of a performance test using the D15C15T15S15I15 dataset with a weight range from 0.4 to 0.8. WIS outperforms SPAM and WSpan. When the w-confidence threshold is lowered, the performance difference becomes larger. At higher weight confidences, such as 90%, the performance of WIS becomes better. We can see that the number of patterns for WIS is decreased as the sequential s-confidence and w-confidence are increased. Recall that WSpan can also

adjust the number of patterns by resetting the weight range, although we fixed the weight range in these tests. Decreasing a weight range means more priority is given to a support measure. However, WIS prunes the sequential patterns with weak support and/or weight affinity. Users can choose their level of interest and use a sequential s-confidence and/or w-confidence. If users increase the sequential w-confidence threshold, it means they want patterns that involve items with higher weight affinity.

6.3.6.2.2 Quality of patterns in WIS

In previous evaluation, we showed that the sequential s-confidence and w-confidence can be used to prune patterns with weak support and/or weight affinity and the sequential patterns with high affinity sequential patterns can be used in real applications. In all test datasets, items are expressed as integer values so it is difficult to understand the meaning of items and discovered sequential patterns. In this evaluation, the D7C7T5S4I2.5 dataset is used to illustrate the quality of weighted sequential patterns mined by WIS. A minimum support is set to 2.5% and a weight range is set as 0.1 – 0.3. We analyzed the patterns discovered by WIS to show the effectiveness of weighted sequential pattern mining with support and/or weight affinity. We compared the patterns mined by WIS with those of SPAM and WSpan. For instance, sequential patterns <(2) (45) (27, 91) (17, 70)>:12 and <(1, 61, 91) (27) (91) (70)>:12 are mined by SPAM and sequential patterns <(70) (61) (45, 61)>:40 and <(91) (47) (91) (27, 91)>:47 are discovered by WSpan. However, these patterns are all removed by s-confidence (min_sconf = 0.6) and w-confidence (min_wconf = 0.6) respectively. That is, these sequential patterns are patterns with weak affinity patterns.

Although the minimum support threshold is increased, these weak affinity patterns such as <(2) (45) (27, 91) (17, 70)>:12 and <(1, 61, 91) (27) (91) (70)>:12 are found by SPAM. In addition, although the minimum support threshold is increased and/or the weight range is changed, the weak affinity patterns such as <(70) (61) (45, 61)>:40 and <(91) (47) (91) (27, 91)>:47 are still discovered in WSpan. The weak affinity patterns can be effectively pruned by sequential s-confidence and/or w-confidence. The affinity pattern mining approach can be effectively applied to discover unusual patterns which may need special attention (to detect fraudulent sequential patterns used by money laundering and other financial crimes).

6.3.6.2.3 Scalability test

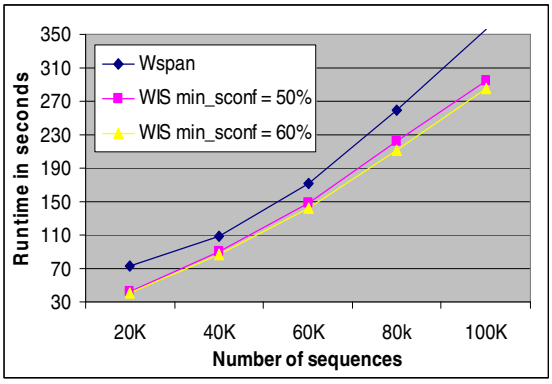

Figure 72. Scalability test in WIS (Min_sup = 0.4%).

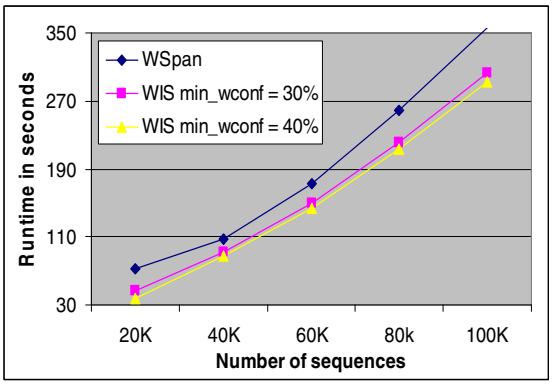

Figure 73. Scalability test in WIS (Min_sup = 0.5%).

DxC2.5T5S4I2.5 dataset

The DxC2.5T5S4I2.5 dataset was used to test scalability with the number of sequences in the database. From the performance test, WIS scales much better than WSpan and becomes better as sequential s-confidence and/or w-confidence are increased. WSpan shows linear scalability with the number of transactions from 20k to 100k. However, WIS is much more scalable than WSpan. In this test, we set a minimum support as 0.4% and 0.5% respectively and a weight range as 0.1 to 0.5. In Figure 72 and Figure 73, we can see that WIS has much better scalability in terms of number of sequences and becomes faster as the sequential s-confidence and w-confidence are increased.

6.3.7 Summary

In summary, WIS is efficient and scalable in weighted interesting pattern mining. Although WSpan uses a weight range and a minimum weight to adjust the number of sequential patterns, these sequential patterns include items with weak support and/or

weight affinity. WIS is faster than WSpan and SPAM and generates smaller but more important sequential patterns.

7. FURTHER EXTENSIONS

In this section, future extensions for weighted pattern mining are discussed. We show examples of applications for weighted frequent pattern mining. In huge datasets, extracting valuable patterns is not easy work. Previous pattern mining algorithms use the same priority for each pattern. The number of frequent patterns becomes huge as the minimum support becomes lower. Therefore, it is difficult for users to find more important patterns. Applying weights is effective and efficient to not only generate more important patterns but also adjust the number of patterns. Specifically, it is more effective to apply weight constraints to pattern mining with lower minimum support. Many opportunities exist to apply weight based pattern mining.

7.1. Applications of weighted pattern mining

7.1.1 Biomedical and DNA data analysis

Weight-based sequential pattern mining can be applied in biomedical data and DNA analysis. The weighed sequential pattern mining approach can be applied in search and comparison among DNA sequences, and identification of co-related gene sequences. For example, most diseases are not triggered by a single gene but by a combination of genes. When identifying specific disease gene sequences, we can give more weight to the important gene sequences and identify co-occurring gene sequences including the disease gene sequences.

7.1.2 Analysis of Web access pattern

In web log mining, we can apply weighted sequential pattern mining to analyze web access patterns. It is more efficient to find important sequential patterns by giving more weights to items within important sequences. For example, in application domain such as financial data analysis, retail industry and telecommunication industry, weighted sequential pattern mining can also be used to detect unusual access such as sequences related to financial crimes, fraudulent telecommunication activities, and purchase of expensive items within a short time. In this case, high weights are given to previously found fraudulent sequences to discover the suspicious sequential patterns.

7.2 Applications of weighted pattern mining with support and/or weight affinity

Weighting applications are common and important in the real world. Particularly, more meaningful patterns can be found and used in the real world by mining patterns with support and/or weight affinity. In other words, weighted sequential pattern mining with support and/or weight affinity can be used to find previous fraudulent users and their usage patterns in crimes such as money laundering, purchase of expensive items within a short time, and use of stolen mobile. We can apply weighted sequential pattern mining approaches in several ways.

First, after giving more weight to previously found fraudulent sequences, similar patterns can be mined and analyzed. Second, patterns with different levels of support may be identifying fraudulent patterns since the usage (transaction) frequency for each user are regular. Additionally, in analyzing customer buying patterns, the level of affinity can help catch fraudulent patterns. For example, customers have purchasing

styles. Sequential patterns with different level of price (weight) may be fraudulent patterns. The techniques of mining patterns with different level of support and/or weight affinity can be applied to detect fraudulent patterns used in money laundering and other financial crimes.

Finally, techniques of mining patterns with support and/or weight affinity can be applied to identify co-occurring gene sequences in biomedical data and DNA analysis. The pattern mining with support and/or weight affinity can help determine the kinds of genes that are likely to co-occur together in target samples.

8. CONCLUSIONS

Many studies on frequent pattern mining have been conducted in the last decade. One of the main limitations of the traditional method for mining frequent patterns is that all items are treated uniformly, while real items have different characteristics. For this reason, weighted frequent pattern mining algorithms have been studied. Mining the complete set of patterns often suffers from generating a very large number of patterns and association rules. Although frequent closed/maximal patterns mining algorithms are suggested, they, in large databases, still generate too many patterns when support is low or the pattern becomes long. Additionally, the previous weighted association rule mining algorithms adopted an Apriori algorithm. However, Apriori-based weighted frequent pattern mining algorithms use candidate set generation and test that is very costly to do that.

In this research, we suggested efficient and scalable frequent pattern mining algorithms with weight constraints. Our main approach is to push the weight constraints into the pattern growth algorithm while maintaining the downward closure property.

First, we developed WFIM which focuses on weighted frequent pattern mining based on a pattern growth algorithm. A weight range and a minimum weight constraint are defined and items are given different weights within the weight range. The extensive performance analysis shows that WFIM is efficient and scalable in weighted frequent pattern mining. Many improved algorithms using divide and conquer methods have been

suggested. In future work, the WFIM can be extended by using a combination of FP-growth based algorithms with better performance.

Second, we developed the WLPMiner algorithm that integrates a weight constraint measure with a length decreasing support constraint measure for mining frequent patterns. The key insights achieved in this approach are the high performance of the WSVE property and the use of a weight range in the weight constraint. We show that combining a weight constraint with a length decreasing support constraint improves performance in terms of the number of patterns and runtime. WLPMiner is efficient and scalable in weighted frequent pattern mining. In future work, the WSVE property will be used with different pruning techniques suggested in other algorithms using length decreasing support constraints.

Third, we defined the problem of mining weighted interesting patterns. We introduced a w-confidence measure and the concept of weighted hyperclique patterns using the w-confidence. Our main goal in this framework is to push a w-confidence and an h-confidence into a weighted frequent pattern mining algorithm based on the pattern growth method and to prune weak affinity patterns. In WIP, the w-confidence and/or the h-confidence are used to avoid generating patterns that involve items with different weight and/or support levels. The extensive performance analysis shows that WIP is efficient and scalable in weighted frequent pattern mining.

Finally, we studied the problem of mining weighted sequential patterns and suggested weighted sequential pattern mining algorithms: WSpan (Weighted Sequential pattern mining with a weight range and a minimum weight) and WIS (Weighted

Interesting Sequential pattern mining with a similar level of support and/or weight affinity). WSpan focused on weighted frequent pattern mining based on the prefix projected sequential pattern growth approach. A weight range and a minimum weight are used to adjust the number of sequential patterns. In WIS, we define sequential s-confidence and w-confidence measures and the concept of weighted interesting sequential patterns by using the two measures. Our main goal in this framework is to push sequential s-confidence and/or w-confidence into the weighted sequential pattern mining algorithm based on the pattern growth method. The sequential s-confidence and/or w-confidence measures can be used to avoid generating spurious sequential patterns that involve items from different support and/or weight levels. The extensive performance analysis shows that WSpan and WIS are efficient and scalable in weighted affinity pattern mining. Many improved techniques such as sequential pattern mining using pseudo projection or bitmap representation have been suggested. In future work, these algorithms can be extended by using a combination of the techniques.

As the reader can see, this area is a very promising research area. Many opportunities exist to improve the performance of weighted frequent pattern mining.

REFERENCES

[1] J. Ayres, J. Gehrke, T. Yiu, and J. Flannick, "Sequential Pattern Mining Using a Bitmap Representation," *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 429-435, July 2002.

[2] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pp. 207-216, June 1993.

[3] R. Agrawal, and R. Srikant, "Fast Algorithms for Mining Association Rules in Large Databases," *Proceedings of the 20th International Conference on Very Large Data Bases, (VLDB'94)*, pp. 487-499, Sep. 1994.

[4] R. Agrawal, and R. Srikant, "Mining Sequential Patterns," *Proceedings of the Eleventh International Conference on Data Engineering*, pp. 3-14, March 1995.

[5] F. Bonchi, and B. Goethals, "FP-Bonsai: The Art of Growing and Pruning Small FP-trees," *Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD`04)*, pp. 155-160, May 2004.

[6] C. Bucila, J. Gehrke, D. Kifer, and W. White, "DualMiner: A Dual-Pruning Algorithm for Itemsets with Constraints," *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 42-51, July 2002.

[7] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, "Exante: Anticipated Data Reduction in Constrained Pattern Mining," *Proceedings of the Seventh European Conference on Principles and Practice of Knowledge Discovery in Databases,* pp. 59-70, Sep. 2003.

[8] F. Bonchi, F. Giannotti, A. Mazzanti, and D. Pedreschi, "ExAMiner: Optimized Level-wise Frequent Pattern Mining with Monotone Constraints," *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, pp. 11-18, Dec. 2003.

[9] F. Bonchi, and C. Lucchese, "On Closed Constrained Frequent Pattern Mining," *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM 2004)*, pp. 35-42, Nov. 2004.

[10] F. Bonchi, and C. Lucchese, "Pushing Tougher Constraints in Frequent Pattern Mining," *Tenth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD`05)*, pp. 114-124, May 2005.

[11] S. Brin, R. Motwani and C. Silverstein, "Beyond Market Baskets: Generalizing Association Rules to Correlations," *Proceedings of the 1997 ACM SIGMOD International Conference on Management of Data*, pp. 255-264, May 1997.

[12] C. H. Cai, A. W. Chee Fu, C. H. Cheng, and W. W. Kwong. "Mining Association Rules with Weighted Items," *Proceedings of the Sixth International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2005)*, July 1998.

[13] M. S. Chen, J. Han, and P. S. Yu. "Data Mining: An Overview from Database Perspective," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, pp. 866-883, Dec. 1996.

[14] H. Cheng, X. Yan, and J. Han, "IncSpan: Incremental Mining of Sequential Patterns in Large Databases," *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 527-532, Aug. 2004.

[15] D. Chiu, Y. Wu, and A. L. Chen, "An Efficient Algorithm for Mining Frequent Sequences by a New Strategy without Support Counting," *Proceedings of the Twentieth International Conference on Data Engineering*, pp. 375-386, March/April 2004.

[16] H. Chung, X. Yan, and J. Han, "SeqIndex: Indexing Sequences by Sequential Pattern Analysis," *Proceedings of the Fifth SIAM International Conference on Data Mining*, pp. 601-605, April 2005.

[17] E. Cohen, M. Datar, and C. Yang, "Fining Interesting Associations without Support Pruning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, pp. 64-78, Jan./Feb. 2001.

[18] S. Cong, J. Han, and D. Padua, "Parallel Mining of Closed Sequential Patterns," *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 562-567, Aug. 2005.

[19] G. Cong, K. L. Tan, Anthony K.H. and Tung, F. Pan, "Mining Frequent Closed Patterns in Microarray Data," *Proceedings of the Fourth IEEE International Conference on Data Mining (ICDM 2004)*, pp. 363-366, Nov. 2004.

[20] G. Cong, A. K.H. Tung, X. Xu, F. Pan, and J. Yang, "FARMER: Finding Interesting Groups in Microarray Datasets," *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pp. 143-154, June 2004.

[21] A. Das, "Rapid Association Rule Mining," *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, pp. 474-481, Nov. 2001.

[22] M. Ester, "A Top-Down Method for Mining Most Specific Frequent Patterns in Biological Sequence Data," *Proceedings of the Fourth SIAM International Conference on Data Mining*, pp. 90-101, April 2004.

[23] B. Fung, K. Wang, and M. Ester, "Hierarchical Document Clustring Using Frequent Itemsets," *Proceedings of the Third SIAM International Conference on Data Mining*, pp. 59-70, May 2003.

[24] K. Gade, J. Wang, and G. Karypis, "Efficient Closed Pattern Mining in the Presence of Tough Block Constraints," *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 138-147, Aug. 2004.

[25] M. Garofalakis, R. Rastogi, and K. Shim. "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints," *Proceedings of the Twenty-fifth International Conference on Very Large Data Bases, (VLDB'99)*, pp. 223-234, Sep. 1999.

[26] B. Goethals, and M. J. Zaki, "FIMI`03: Workshop on Frequent Itemset Mining Implementations," *Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, Nov. 2003.

[27] G. Grahne and J. Zhu. "Efficiently Using Prefix-trees in Mining Frequent Itemsets," *Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations*, Nov. 2003.

[28] M. E. Hajj, and O. R. Zaiane, "Inverted Matrix: Efficient Discovery of Frequent Items in Large Datasets in the Context of Interactive Mining," *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 109-118, Aug. 2003.

[29] J. Han and Y. Fu, "Mining Multiple-Level Association Rules in Large Databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 11, pp. 798-805, Sep/Oct. 1999.

[30] J. Hipp, and U. Guntzer, "Is Pushing Constraints Deeply into the Mining Algorithms Really What We Want?: An Alternative Approach for Association Rule Mining", *ACM SIGKDD Explorations Newsletter***,** vol. 4, pp. 50-55, June 2002.

[31] J. Hipp, U. Guntzer, and G. Nakhaeizadeh, "Algorithms for Association Rule Mining – A General Survey and Comparison", *ACM SIGKDD Explorations Newsletter***,** vol. 2, pp. 58-64, June 2000.

[32] J. Han, and M. Kamber, *Data Mining Concepts and Techniques*, San Francisco: Morgan Kaufmann, 2001.

[33] J. Han, J. Pei, B. M. Asi, Q. Chen, U. Dayal, and M. C. Hsu, "FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining," *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 355-359, Aug. 2000.

[34] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, pp. 1-12, May 2000.

[35] 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.

[36] J. Han, J. Wang, Y. Lu, and P. Tzvetkov, "Mining Top-K Frequent Closed Patterns without Minimum Support," *Proceedings of the Second IEEE International Conference on Data Mining (ICDM 2002)*, pp. 211-218, Dec. 2002.

[37] H. Huang, X. Wu, and R. Relue, "Association Analysis with One Scan of Databases," *Proceedings of the Second IEEE International Conference on Data Mining (ICDM 2002)*, pp. 211-218, Dec. 2002.

[38] Y. Huang, W. Wu and H. Xiong, A Hybrid Approach for Mining Maximal Hyperclique Patterns," *Proceedings of* Sixteenth IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2004), pp. 354-361, Nov. 2004.

[39] L. Jia, R. Pei, and D. Pei, "Tough Constraint-Based Frequent Closed Itemsets Mining," *Proceedings of the 2003 ACM Symposium on Applied Computing (SAC)*, pp. 416-420, March 2003.

[40] W. Y. Kim, Y. K. Lee, and J. Han, "CCMine: Efficient Mining of Confidence-Closed Correlated Patterns," *Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD`04)*, pp. 569-579, May 2004.

[41] C. Kim, J. H. Lim, R. Ng, and K. Shim, "SQUIRE: Sequential Pattern Mining with Quantities," *Proceedings of the Twentieth International Conference on Data Engineering*, pp. 827-827, March/April 2004.

[42] H. C. Kum, J. Pei, W. Wang, and D. Duncan, "ApproxMAP: Approximate Mining of Consensus Sequential Patterns," in *Proceedings of the Third SIAM International Conference on Data Mining*, pp. 311-315, May 2003.

[43] L. V.S. Lakshmanan, R. T. Ng, J. Han, and A. Pang, "Optimization of Constrained Frequent Set Queries with 2-variable Constraints," *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pp. 157-168, June 1999.

[44] Y. K. Lee, W. Y. Kim, Y. D. Cai, and J. Han, "CoMine: Efficient Mining of Correlated Patterns," *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, pp. 581-584, Dec. 2003.

[45] C. K. Leung, L. V.S. Lakshmanan, and R. T. Ng, "Exploiting Succinct Constraints Using FP-trees," *ACM SIGKDD Explorations Newsletter***,** vol. 2, pp. 40-49, June 2002.

[46] J. Liu, "Mining Frequent Item Sets by Opportunistic Projection," *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 229-238, July 2002.

[47] B. Liu, W. Hsu, and Y. Ma, "Mining Association Rules with Multiple Minimum Supports," *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 337-341, Aug. 1999.

[48] G. Liu, H. Lu, W. Lou, and J. X. Yu, "On Computing, Storing and Querying Frequent Patterns," *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 607-612, Aug. 2003.

[49] G. Liu, H. Lu, Y. Xu, and J. X. Yu, "Ascending Frequency Ordered Prefix-tree: Efficient Mining of Frequent Patterns," *Eighth International Conference on Database Systems for Advanced Applications (DASFAA '03)*, pp. 65-72, March 2003.

[50] H. Albert-Lorincz, and J. F. Boulicaut, "Mining Frequent Sequential Patterns under Regular Expressions: A Highly Adaptive Strategy for Pushing Constraints," *Proceedings of the Third SIAM International Conference on Data Mining*, pp. 316-320, May 2003.

[51] S. Ma and J. L. Hellerstein, "Mining Mutually Dependent Patterns," *Proceedings of the First IEEE International Conference on Data Mining (ICDM 2001)*, pp. 409-416, Nov/Dev. 2001.

[52] R. T. Ng, L. V.S. Lakshmanan, A. Pang, and J. Han, "Exploratory Mining and Pruning Optimizations of Constrained Associations Rules," *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, pp. 13-24, June 1998.

[53] E. R. Omiecinski, "Alternative Interest Measures for Mining Associations in Databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, pp. 866-883, Dec. 1996.

[54] F. Pan, G. Cong, Anthony K. H. Tung, J. Yang, and M. J. Zaki, "CARPENTER: Finding Closed Patterns in Long Biological Datasets," *Proceedings of the Ninth ACM*

SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 637-642, Aug. 2003.

[55] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules," *Proceedings of the Seventh International Conference on Database Theory (ICDT),* pp. 398-416, Jan. 1999.

[56] J. Pei, and J. Han, "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets," *Proceedings of the Fifth ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pp. 21-30, May 2000.

[57] J. Pei, and J. Han, "Can We Push More Constraints into Frequent Pattern Mining?" *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 350-354, Aug. 2000.

[58] J. Pei, and J. Han, "H-Mine: Hyper-Structure Mining of Frequent Patterns in Large Databases," *Proceedings of the First IEEE International Conference on Data Mining (ICDM 2001)*, pp. 441-448, Nov/Dec. 2001.

[59] J. Pei, J. Han, B. M. Asi, and H. Pino, PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, *Proceedings of the Seventeenth International Conference on Data Engineering*, pp. 215-224, April 2001.

[60] J. Pei, J. Han, B. M. Asi, J. Wang, and Q. Chen, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 1424-1440, Oct. 2004.

[61] J. Pei, J. Han, and L. V.S. Lakshmanan, "Mining Frequent Itemsets with Convertible Constraints," *Proceedings of the Seventeenth International Conference on Data Engineering*, pp. 433-442, April 2001.

[62] J. Pei, J. Han, and W. Wang, "Mining Sequential Patterns with Constraints in Large Databases," *Proceedings of the 2002 ACM CIKM International Conference on Information and Knowledge Management*, pp. 18-25, Nov. 2002.

[63] H. Pinto, J. Han, J. Pei, and K. Wang, "Multi-dimensional Sequence Pattern Mining," *Proceedings of the 2001 ACM CIKM International Conference on Information and Knowledge Management*, pp. 474-481, Nov. 2001.

[64] V. Pudi, and J. R. Haritsa, "Generalized Closed Itemsets for Association Rule Mining," *Proceedings of the Nineteenth International Conference on Data Engineering*, pp. 714-716, March 2003.

[65] A. Savasere, "An Efficient Algorithm for Mining Association Rules in Large Databases," *Proceedings of the Twenty-First International Conference on Very Large Data Bases, (VLDB'95)*, pp. 432-444, Sep. 1995.

[66] M. Seno, and G. Karypis, "LPMiner: An Algorithm for Finding Frequent Itemsets Using Length-Decreasing Support Constraint," *Proceedings of the First IEEE International Conference on Data Mining (ICDM 2001)*, pp. 505-512, Nov/Dec. 2001.

[67] M. Seno, and G. Karypis, "SLPMiner: An Algorithm for Finding Frequent Sequential Patterns Using Length-Decreasing Support Constraints," *Proceedings of the Second IEEE International Conference on Data Mining (ICDM 2002)*, pp. 418-425, Dec. 2002.

[68] M. Seno, and G. Karypis, "Finding Frequent Patterns Using Length Decreasing Support Constraints," University of Minnesota, Minneapolis, Technical Report 03-003, 2003.

[69] R. Srikant, and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proceedings of the Fifth International Conference on Extending Database Technology*, pp. 3-17, March 1996.

[70] R. Srikant, and R. Agrawal, "Mining Association Rules with Item Constraints," *Proceedings of the Third ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 67-73, Aug. 1997.

[71] M. Steinbach, P. N. Tan, H. Xiong and V. Kumar, "Generalizing the Notion of Support," *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 689-694, Aug. 2004.

[72] F. Tao, "Weighted Association Rule Mining Using Weighted Support and Significant Framework," *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 661-666, Aug. 2003.

[73] P. N. Tan, V. Kumar and J. Srivastava, "Selecting the Right Interestingness Measure for Association Patterns," *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 32-41, July 2002.

[74] P. Tzvetkov, X. Yan, and J. Han, "TSP: Mining Top-K Closed Sequential Patterns," *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, pp. 347-354, Dec. 2003.

[75] J. Wang, and J. Han, "BIDE: Efficient Mining of Frequent Closed Sequences," *Proceedings of the Twentieth International Conference on Data Engineering*, pp. 79-90, March/April 2004.

[76] J. Wang, J. Han, Y. Lu, and P. Tzvetkov, "TFP: An Efficient Algorithm for Mining Top-K Frequent Closed Itemsets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, pp. 652-664, May 2005.

[77] J. Wang, J. Han, and J. Pei, "CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets", *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 236-245, Aug. 2003.

[78] K. Wang, Y. He, and J. Han, "Mining Frequent Itemsets Using Support Constraints," *Proceedings of the Twenty-Sixth International Conference on Very Large Data Bases, (VLDB'00)*, pp. 43-52, Sep. 2000.

[79] J. Wang, and G. Karypis, "BAMBOO: Accelerating Closed Itemset Mining by Deeply Pushing the Length-Decreasing Support Constraint," *Proceedings of the Fourth SIAM International Conference on Data Mining*, pp. 432-436, April 2004.

[80] K. Wang, L. Tang, J. Han, and J. Liu, "Top Down FP-Growth for Association Rule Mining", *Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD`02)*, pp. 334-340, May 2002.

[81] K. Wang, Y. Xu, and J. X. Yu, "Scalable Sequential Pattern Mining for Biological Sequences," *Proceedings of the 2004 ACM CIKM International Conference on Information and Knowledge Management*, pp. 178-187, Nov. 2004.

[82] W. Wang, J. Yang, and P. S. Yu, "Efficient Mining of Weighted Association Rules (WAR)," *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 270-274, Aug. 2000.

[83] H. Xiong, X. He, C. Ding, Y. Zhang, V. Kumar, and S. R. Holbrook, "Identification of Functional Modules in Protein Complexes via Hyperclique Pattern Discovery," *Proceedings of the Pacific Symposium on Biocomputing (PSB`05)*, Jan. 2005.

[84] H. Xiong, S. Shekhar, P. N. Tan and V. Kumar, "Exploiting a Support-based Upper Bound of Pearson's Correlation Coefficient for Efficiently Identifying Strongly Correlated Pairs," *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 334-343, Aug. 2004.

[85] H. Xiong, M. Steinbach, P. N. Tan, and V. Kumar, "HICAP: Hierarchical Clustering with Pattern Preservation," *Proceedings of the Fourth SIAM International Conference on Data Mining*, pp. 279-290, April 2004.

[86] H. Xiong, P. N. Tan and V. Kumar, "Mining Strong Affinity Association Patterns in Data Sets with Skewed Support Distribution," *Proceedings of the Third IEEE International Conference on Data Mining (ICDM 2003)*, pp. 387-394, Dec. 2003.

[87] H. Xiong, P. N. Tan and V. Kumar, "Mining Hyperclique Patterns with confidence Pruning," University of Minnesota, Minneapolis, Technical Report 03-006, 2003.

[88] X. Yan, J. Han, and R. Afshar, "CloSpan: Mining Closed Sequential Patterns in Large Datasets," *Proceedings of the Third SIAM International Conference on Data Mining*, pp. 166-177, May 2003.

[89] J. Yang, P. S. Yu, W. Wang, and J. Han, "Mining Long Sequential Patterns in a Noisy Environment," *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 406-417, June 2002.

[90] U. Yun, and J. J. Leggett, "WFIM: Weighted Frequent Itemset Mining with a Weight Range and a Minimum Weight," *Proceedings of the Fourth SIAM International Conference on Data Mining*, pp. 636-640, April 2005.

[91] U. Yun, and J. J. Leggett, "WLPMiner: Weighted Frequent Pattern Mining with Length Decreasing Support Constraints," *Ninth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD`05)*, pp. 555-567, May 2005.

[92] M. J. Zaki, "Generating Non-Redundant Association Rules," *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 34-43, Aug. 2000.

[93] M. J. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences," *Machine Learning*, vol. 42, pp. 31-60, Jan. 2001.

[94] M. J. Zaki, "CHARM: An Efficient Algorithm for Closed Itemset Mining," *Proceedings of the Second SIAM International Conference on Data Mining*, pp. 457-473, April 2002.

[95] M. J. Zaki, "Fast Vertical Mining Using Diffsets," *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 326-335, Aug. 2003.

[96] Z. Zheng, "Real World Performance of Association Rule Algorithms," *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 401-406, Aug. 2001.

VITA

Unil Yun was born in Seoul, Korea on November 25, 1971. He earned his Bachelor of Science degree in computer engineering from Hong Ik University, Seoul, Korea in 1995, and earned his Master of Science degree in computer science from Korea University, Seoul, Korea, in 1997. He worked at Multimedia Laboratory in Korea Telecom, Seoul, Korea, as research staff from February 1997 through June 2002. He participated in several projects: Hanmir Multimedia Information Retrieval systems, XML based Data Management and Information Retrieval systems.

He enrolled in the Department of Computer Science at Texas A&M University, College Station, in the Fall of 2002, and started his research under the direction of Dr. John J. Leggett. He worked with the Center for the Study of Digital Libraries (CSDL) at Texas A&M University from September 2002 through December 2005, under the supervision of Dr. John J. Leggett. He received his Ph.D. in computer science from Texas A&M University in December 2005. His research interests include data mining, database systems, bioinformatics, data management, information retrieval, digital libraries, and hypertext systems. His permanent address is:

25-28 Jongam-1-Dong

Sungbuk-Gu

Seoul, Republic of Korea

The typist for this dissertation was Unil Yun.